Improved Security Evaluation of the Software by using PSSS based Security Analyzer

Surkhab Shelly
Department of CSE
Guru Nanak Dev University

Anil Kumar
Department of CSE
Guru Nanak Dev University

ABSTRACT
After analyse the three security processes (CLASP, SDL AND PSSS) it has been selected that the PSSS as security approach to develop a secure project since of its advantages over the other two security processes. The most important objective of PSSS security process is to improve the effectiveness of software security projects. The overall objective of this paper is to evaluate the security analysis of the given software and return a security report which allows programmers to take certain action based upon the outcomes. The main objective of this dissertation is to develop a secure application using PSSS process and the other objectives are- To integrate the each activity of each phase of PSSS in each phase of software development. This paper initiated security process by establishment of a security Engineering approach consisting of security activities forming a process to support the development of more secure software. The validation of the security model has been done by approach by developing a security report through analysis. Thus, one can make its product more secured by rewrite and replacing some security threats in secure manner.

Keywords
PSSS MODEL, SECURITY AND SDLC, CLASP, SDL, ISO, SSE-CCM

1. INTRODUCTION
The application of a efficient, well-organized, quantifiable approach to the development, operation, and continuation of software; that is, the application of engineering to software or Software engineering is the branch of systems engineering concerned with the development of huge and difficult software intensive systems[1]. It has focused on the real-world goal for services provide by, and constraints on such systems; the accurate requirement of system construction and behaviour, and the implementation of these condition; the activities requisite in order to develop an assurance that the specifications and real world goals have been met; the advancement of such systems over time and across system families. It involves the elicitation of the systems requirements, the specification of the system, its architectural and detailed design. In addition, the system wants to be verified and validated, a set of activities that usually take more than 50% of all improvement resources. Testing techniques and tools at different levels (unit, integration, and system) are essential. Software development being a human intensive process, management and quality control techniques are also required to run successful projects and construct quality systems. In generally systems, including telecommunication systems, software is the overruling component in terms of cost and complexity. Good software engineering practice and tools can therefore make a substantial distinction, even to the amount that they may be the driving force of the project success.

1.1 Security in SDLC
When defining security in the SDLC, two areas must be address. The first area is the SDLC process itself. The second area is application operational security. One should understand the SDLC process and related security activities and the specific application and operational security controls that are available to the application designer. Security reflects the systems capability to protect itself from accidental or planned exterior attack. The primary goals of software security are the protection of the confidentiality, integrity, and availability (CIA) of the information assets and resources that the software creates, stores, processes, or transmit together with the executing programs themselves. Preserving confidentiality is about preventing unauthorized disclosure; preserving integrity is about preventing unauthorized alteration; and preserving availability is about preventing unauthorized destruction or denial of access or service. One more method of looking at security in computer systems is that we attempt to protect the services and data it offers aligned with security threats. There are four types of security threats to consider: Interception refers to the condition that an unauthorized party has gained access to a service or data. Interception happen when data are illegally copied, for example, once breaking into a person's private directory in a file system. An example of interruption is when a file is infected or missing. In general, interruption refers to the condition in which services or data become unavailable, broken, damaged, and so on. Modifications involve unauthorized altering of data or tamper with a service so that it no longer adheres to its original specifications. Examples of modifications include intercept and then changing transmitted data, tampering with database entries, and altering a program so that it secretly logs the activities of its user. Fabrication refers to the situation in which added data or activity are generated that would usually not exist. For example, an intruder may effort to add an entry into a password file or database. Similarly, it is sometimes possible to break into a system by replay before sent messages.

1.2 SDLC Security Issues
The goal of a good SDLC process is to capture, verify, and implement all the requirements needed to make the application of use to the organization. These requirements consist of security needs defined around confidentiality, integrity, and availability of the information system. If security requirements are accurately identified and the appropriate security controls added are to the application to meet these requirements, the result is a secure application. But in reality, developing applications involve tradeoffs to meet budget, resource, and time constraints placed on the project. In several cases, security is the first necessity to be dropped [2].

An additional security SDLC issue is the lack of security training and knowledge along with developers and system designers. Poor design decisions are prepared when developers are not aware of existing security risks. As a result of these SDLC security shortcoming, security is frequently an addition, and security controls are implemented as add-ons after the project is complete and security issues come to light. Applications built this way become overly complex, expensive, and hard to maintain. This ensures that security is further

compromised and the application system suffers from continual security problems. Even if a system is designed and developed with security in mind, systems modify over time. New equipment, software, and functionality are added to systems repeatedly over time. These changes must be authorized and track and security issues require to be evaluating as part of a configuration management process.

2. LITERATURE REVIEW

This section deals with the research issues interrelated to software security. The purpose of this paper is to help you recognize the important role that security plays in the SDLC. A number of small and large organizations and departments have devoted their time to build up policies, strategies, guidelines to create a secure application.

Prem kumar and Stuart , 2000 [3] has discussed that almost every software controlled v system faces threats since potential adversaries, from Internet-aware client applications running on PCs, to difficult telecommunications and power systems accessible above the Internet, to commodity software with copy protection mechanisms. Software engineers must be cognizant of these threats and engineer systems with credible defences, whereas still delivers value to customers. Shreyas 2000 [4] has discussed that since the creation of distributed systems, security of software systems has been an issue of massive concern. Traditionally, security is integrated in a software system after all the functional requirements have been addressed. It has focused on utilize of Software Architecture to solve certain problems that are faced in the engineering of secure systems.

Mike, 2003 [5] has presented Systems Security Engineering Capability Maturity Model (SSE-CMM). This model provides industry best practice guidance not including being specific as to how security solutions are implemented. The SSE-CMM provides a broad list of “base practices” from which the security engineer can profit when defining the objectives of the security implementation. Louise 2004 [6] has presented a structure for developing security requirements of information systems. In this method, qualitative metrics are used to give up experimental information that can be used to develop the evaluation process specially risk assessment, vulnerability assessment, protection profiles, and test coverage which are significant aspects of systems specification.

Nithin 2007[1] has discussed that Security is an important issue in the development of information systems; presently the common approach towards the inclusion of security within a system is to classify security requirements after the definition of a system. Conversely, as pointed earlier, this approach leads many times to problems and systems full of security vulnerabilities. It should be possible to remove such problems through the integration of security concern at every phase of the system development.

Bart et al, 2009 [7] has discussed the three high-profile processes for the development of secure software namely OWASP’s CLASP, Microsoft’s SDL and McGraw’s Touch points are evaluate and compare in detail. Development processes for software construction are common knowledge and mainstream practice in most development organizations. Unfortunately, these processes suggest little support in order to meet security requirements.

Eric et al, 2010 [8] has surveyed and contrasts the security models of existing programming languages and platforms. As it identified, there is an inbuilt trade-off between simplicity and flexibility. Simple models tend to provide stronger security guarantees and are less likely to provoke implementation bugs.

Francisco et al, 2010 [9] has proposed a security engineering approach to support software security through a specialized process that helps develop more secure software, entitled Process to Support Software Security (PSSS). PSSS can be seen as a heavy process as well as useless because there are other security-based software processes. PSSS offers 37 positive security activities to be adapted and specialized based on corporate software development preference. Pavel et al, 2010 [10] has discussed the significance of the implementation of IT best practices in enterprises and to identify the explanation challenges managers are facing when creating a standardized IT control framework in order to achieve alignment of best practices to business requirements. Gefei et al, 2012 [11] has proposed ST-Editor which is a supporting tool for the creation and the protection of STs according to ISOIIEC 15408. By using the ST-Editor, users can create and maintain STs easily, quickly, and safely because ST-Editor can provide a helpful and protected editing and maintain environment to assist users to describe and edit STs. Danielito, 2012 [12] has discussed that Software security is one of those legacy problems that will not be solved overnight. By addressing all the phases of the Software Development Life Cycle with the principles of secure and resilient software you are well on your way to recovering the overall software security problem and thus further develop the situation for the organizations, community and business sector. Mandal and Pal, 2012 [13] has explain several SDLC models but it rarely followed by organizations for the real project implementations as it lack suitability. On the way to examine the reasons for non-suitability of these models, it is exposed that there are insufficient parameters and metric for judging the characteristics of any SDLC model. James, 2013 has explained that security plays an important role in the Software Development Life Cycle (SDLC). The paper defines security as it applies to the SDLC and discusses overall SDLC security issues. It then covers each phase of the SDLC and specific security controls and issues for each phase. If companies follow the SDLC phases and integrate the suitable security activities the system in this manner, the security shortcomings of the system will be discovered.

2.1 Security Processes

Process

The IEEE defines a process as" a sequence of steps performed for a given purpose". A secure software process can be defined as the set of activities performed to develop, maintain, and deliver a secure software solution. Activities may not necessarily be sequential; they could be concurrent or iterative.

2.1.1 CLASP

CLASP is a pre-defined set of documented processes and tools that can be integrated into any software development process. It is designed to be both easy to adopt and effective. The Open Web Application Security Project (OWASP) is an open community dedicated to enabling organizations to conceive, develop, acquire, operate, and maintain applications that can be trusted. One of their more prominent projects is called the Comprehensive, Lightweight Application Security Process, or CLASP [7].

CLASP is rich with an extensive collection of freely available and open source security resources that make implementing those activities practical and achievable. It includes a set of 24 top-level activities and additional resources, which can be tailored to the development process in use. The primary goal of CLASP is to support the construction of software in which security takes a central role. CLASP provides an extensive set of

security resources that facilitate and support the implementation of the activities.

2.1.2 SDL

Bart defined the SDL to address the security issues they frequently faced in many of their products. SDL comprises a set of activities, which complement Microsoft's development process and which are particularly aimed at addressing security issues. SDL can be characterized as follows [7]:

- Security as a supporting quality: The primary goal of SDL is to increase the quality of functionality driven software by improving its security posture. Security activities are most often related to functionality-based construction activities
- Well-defined process: The SDL process is well organized and related activities are grouped in stages. Although these stages are security specific, it is straight forward to map them to standard software development phases.
- Good guidance: SDL does a good job at specifying the method that must be used to execute activities, which, on average, are concrete and often somewhat pragmatic. For instance, attack surface reduction is guided by a flow chart and threat modelling is described as a more detailed sub process. As a result, the execution of an activity is quite achievable, even for less experienced people.

There are always disadvantages with everything:

- End-user does not see the solution until the system is almost complete.
- Users get a system that meets the need as understood by the developers. There may be a loss in translation.
- Documentation is expensive and time-consuming to create. It is also difficult to keep current.
- Users cannot easily review intermediate products and evaluate whether a particular product (e.g., dataflow diagram) meets their business requirements

2.1.3 PSSS (Process to Support Software Security)

PSSS was designed to follow the iterative and incremental life cycle approach which facilitates the coordination between the PSSS and any particular corporate development process. In order to use the PSSS with other life cycles, this would need validation. There is no need to use all the activities of the PSSS. They can be adapted to function effectively within the organizational development process. It is an important aspect to have each activity as integrated as possible into the life cycle phases and one approach to reach this integration is to apply each activity in parallel with the phases [9].The main advantages gained by applying and following the PSSS are:

- Assurance that security was considered during the system development through elaboration of security activities and artifacts, such as attack trees and abuse case, and those potential security vulnerabilities, threats and risks would be treated.
- Identification and definition of security requirements based on a set of security assessments to protect the system against security problems.
- Assurance that the limited project resources were effectively applied based on security assessments and according to the major negative security impacts.

2.2 COMPARATIVE ANALYSIS OF SECURITY PROCESSES

Table 1: Comparative Analysis of security Processes

Processes Parameters	CLASP	SDL	PSSS
Full Form	Comprehensive, Lightweight Application Security Process.	Security Development Life cycle.	Process to Support Software Security.
Functionality	It is a pre-defined set of documented processes and tools that can be integrated into any software development process.	Microsoft defined it to address the security issues it frequently faced in many of their products.	Structured to provide visibility towards Information security
Nature	It is light weight and more affordable for small organizations.	It is heavyweight and rigorous and suitable for large organizations& time consuming.	It is suitable for both small and large organizations.
Composition	It includes set of 24 top level activities which can be used to develop the software process	SDL comprises a set of activities, which are particularly aimed at addressing security issues.	The PSSS is formed by 37 activities grouped in a set of 11 sub processes.
Implementation	It does not perform manual code inspection.	It performs manual code inspection.	It also performs manual code inspection.
Security design Principles	It determines implementation strategies for security of software.	It determines implementation strategies for security of software.	It determines implementation strategies for security of software.
Limitations	Weakness Analysis cannot be performed in this process. This process cannot build and execute process improvement program.	Weakness analysis cannot be performed in this process. It cannot build & execute process improvement program.	Weakness analysis can be performed in this process. This process can build and execute process improvement program.

3. EXPERIMENTAL SETUP

This chapter explains experimental setup of proposed work. The activities or sub processes are performed for the security analysis of a project by following PSSS security process. For performing these activities a complete and accurate knowledge about security Vulnerability should be must. All these activities are performed only for three security vulnerabilities that are File handling, Pointers and Conditional operators.

3.1 Plan Security

The information regarding security of a project is defined in terms of three security vulnerabilities that are use of Pointers, File Handling Operations and Conditional Operators. These three aspects are to be taken to plan the security for developing a secure project. Programming Environment C++ is taken and Mat lab tool is used for developing a project. For each aspect, a set of ten programs of C++ is considered in which occurrence and density of these three security vulnerabilities will be evaluated.

3.2 Assess security vulnerability

In this sub process, vulnerabilities assessment methods like how many times a pointers, file handling operations and conditional operators are referenced and how many times number of pointers , file handling operations and conditional operators are used in the form of pointer density, file handling density, conditional operators density respectively are planned for identification. For analyzation of identified security vulnerabilities, density values for these security vulnerabilities are to be taken. For example there is various memory errors related to C++ programming like attempting to free memory already freed, this problem comes under memory leaks.

Attempting free memory already freed

```
#include <stdlib.h>
#include <string.h>
...
char *oldString = "Old String";
char newStrig = strdup(oldString);
if(newString == ENOMEM) ... // Fail!!!!
...
free(newString);
```

This example illustrates that any routine which is supplied by the C libraries or ones written within an application which allocate memory must have the memory freed.

Freeing memory that was not allocated

```
#include <stdlib.h>
char *textString = malloc(128*sizeof(char));
if(textString == ENOMEM) ... // Fail!!!!
...
free(textString);// Don't free if allocation failed here
```

3.3 Model Security Threat

This sub process identifies the security threats for these three security vulnerabilities. For example in case of pointers there are some security suspects which cannot be ignored like always initialize the pointer when declared, allocate memory from the heap and return it to the heap at the same level to avoid memory leaks, Catch an exception to delete memory when necessary, always zero a pointer variable once the pointer is no longer valid. Another possibility which may have negative impact and can cause the major damage to software is programming mistakes related to memory corruption. Memory when altered without an explicit assignment due to the unintentional altering of data held in memory or the altering of a pointer to a specific place in memory. For example: - Buffer overflow (Overwrite beyond allocated length-overflow) it is overflow by one byte.

Buffer overflow

```
char *a = malloc(128*sizeof(char));
memcpy(a, data, dataLen);// Error if dataLen too long.
```

Example 2:- Index of array out of bounds (array index overflow-index too large/underflow-negative index)

Index of array out of bounds

```
ptr = (char *) malloc(strlen(string_A));// Should be //(string_A + 1) to account for null termination.

strcpy(ptr, string_A); // Copies memory from string_A//which is one byte longer than its destination ptr.
```

3.4 Assess Security Impact

This sub process reviews the critical activities for security and prioritizes these according to their impact. Out of these three identified security vulnerabilities which vulnerability is considered as prior? For example efficient use of pointers can be taken as prior as compared to file handling operations because memory allocation is directly linked with effective use of pointers. Once the pointer variable has served its purpose, then it is a waste of memory to keep it, and therefore, it is a good practice to deallocate it when no longer needed. In case of Handshaking, file handling is more prior than other two. So we can say priority of these security vulnerabilities depends upon the nature of the developer that what he want in his application.

3.5 Assess Security Risk

It refers to the risk associated with identified security vulnerability. In other words if there is a great use of pointers, file handling operations and conditional operators then what are the risks, which can be evaluated are assessed. For example, if Exception handling is not used then in some exceptional conditions, processed data is not able to store with in a data structure. For example, a floating point divide by zero exception allow to the program to be resumed, by default while out of memory condition might not be resolvable transparently.

3.6 Security Needs

Specify security needs means specification of security needs of the system according to stakeholder as well as customer. Like allocation of memory and deallocation of the memory depends upon various situations. Improper use of dynamic memory allocation can include security bugs or program crashes. Under this process, an agreement is prepared about the security requirements.

3.7 Verify and Validate security

This sub process defines the security verification and validation approach and Perform security verification and security validation of above mentioned three security vulnerabilities. Here we verify these vulnerabilities by finding the occurrences of pointers, file handling operations and conditional operators in C++ programs. It will give an idea of the existence of these security vulnerabilities in a particular program.

3.8 Monitor Security behaviour

The security monitoring tends to measure the effect of the given security vulnerabilities on a given function or modules. For example: - Pointer persistence, in this problem function returning a pointer from stack which can get overwritten by calling function.

Pointer persistence

```
int *get_ii()
{
int ii;// Local stack variable
ii = 2;
return &ii;
}
main()
{
int *ii;
ii = get_ii();//After this call the stack is given up by routine
// get_ii() and its values are no longer safe.
... Do stuff
..ii may be corrupt by this point.
}
```

Pointers are a special problem for persistent data types because there may be more than one pointer to the same object in a data structure. If this was dumped in a naive way, there would be two identical copies of the object in the dump, rather than one object and two pointers to it. It would also be impossible to dump a structure with back pointers because the dump mechanism would get stuck in an infinite recursion.

4. PERFORMANACE EVALUATION

Simulation has been designed and implemented using MATLAB tool. Ten different programs are selected for experimental purpose. Subsequent section contains various results of the given programs.

Figure 1 has shown security analysis of given three security vulnerabilities. Program 1 has been selected for experimental purpose in which blue colour denotes file handling operations, green shows pointers and red colour denotes conditional operations. Y-axis shows the occurrence of the given three security vulnerabilities.

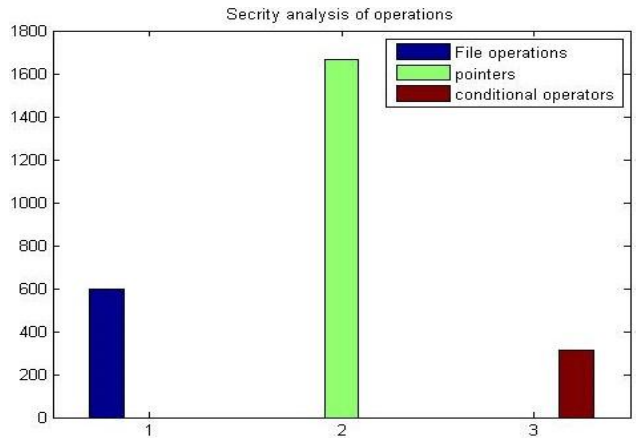

Fig 1: security analysis of operations of First program

Figure 2 has shown the analysis of the of security ratio of program 1. It has been clearly shown that the ratio of pointers is very high than other operations.

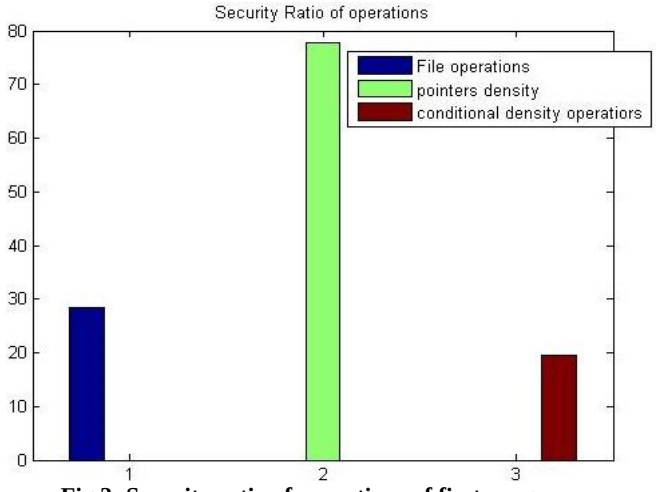

Fig 2: Security ratio of operations of first program

Table 2: Density analysis of given program

NAME	LOC	FILE HANDLING DENSITY	POINTERS DENSITY	CONDITIONAL OPERATORS DENSITY
1	1682	53.3	86.6	6.60
2	278	46.6	6.60	13.3
3	688	73.3	33.3	26.6
4	390	26.6	20.0	6.60
5	388	86.6	46.0	13.3
6	344	46.6	33.3	6.60
7	324	33.3	20.0	6.60
8	337	73.3	73.3	6 .60
9	977	6.60	46.6	20.0
10	406	73.3	66.6	6.60

Table 2 has shown the density of three security vulnerabilities that are File Handling, Pointers, and Conditional operators of different programs which have been taken for Experimental purpose. Here File Handling density shows the effect or impact of file handling operations in different programs of C++. Basically it shows the ratio of occurrence of file handling operations in C++ programs. Similarly Pointers density and Conditional operators density shows the impact of the pointers and conditional operators in each program of C++.

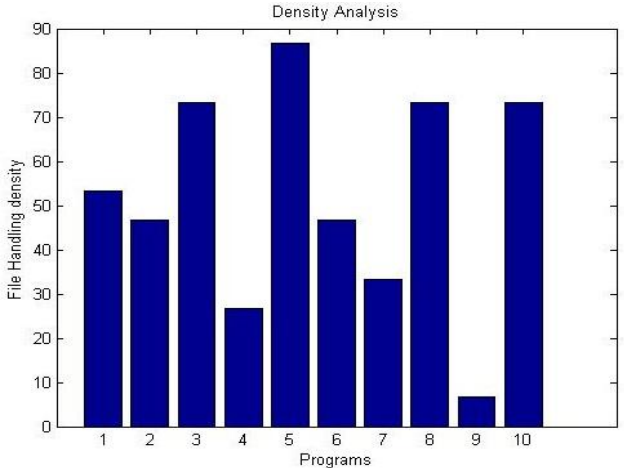

Fig 3: File handling density analysis of each program

Figure 3 has shown the file handling operations density of the ten different programs. It has been shown that the 6th program has highest density so more insecure with respect to handshaking operations. Whereas the program 9 is more secure having lowest density among other programs.

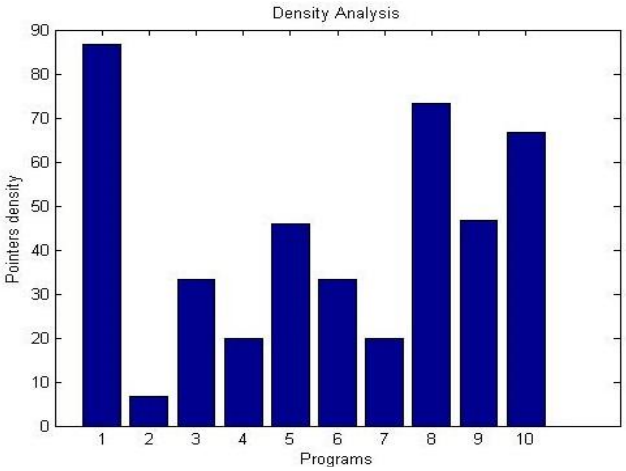

Fig 4: Pointers density analysis of each program

Figure 4 has shown the Pointers density of the ten different programs. It has been shown that the 1st program has highest density so more insecure with respect to memory leakage operations. Whereas the program 2 is more secure having lowest density among other programs.

Figure 5 has shown the Conditional operators density of the ten different programs. It has been shown that the 3rd program has highest density so more insecure with respect to memory leakage operations. Whereas the many programs are more secure having lowest density among other programs.

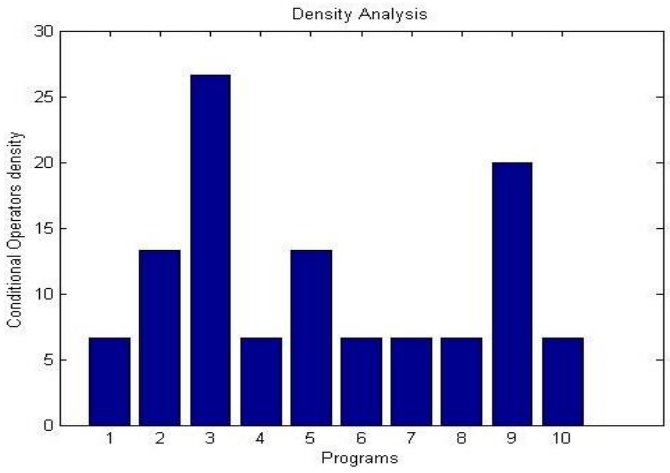

Fig 5: Conditional operator's density analysis

Fig 6 has shown the Density of the ten different programs. It has been shown that the more secured program is 4th one. However the 1, 3, 4, 8 and 10th program are found to be insecure among other programs.

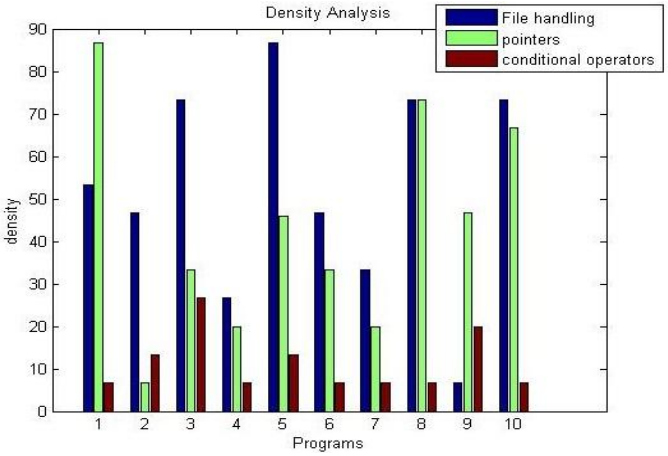

Fig 6: Density analysis

To validate the proposed security model's verification we have run all the program and found that the in real time the security also depend on the relationship of conditional operators density along with either file handling operations density and the pointers density. Figure 7 has shown the run time analysis of program 1. It has been clearly shown that the density has affected the actual calling a lot.

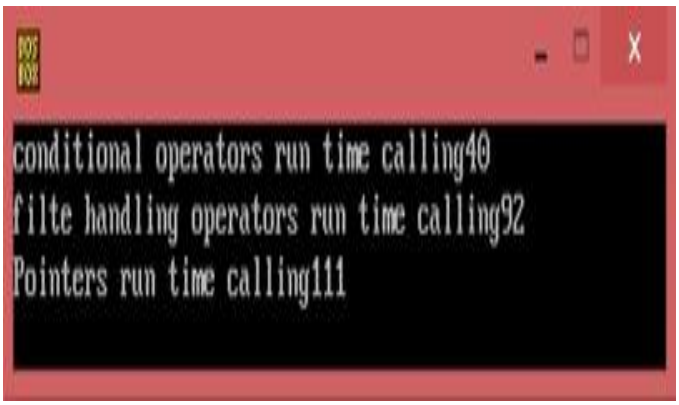

Fig 7: The run time analysis of program 1

5. CONCLUSION & FUTURE WORK

This Paper has evaluated the security analysis of the different projects. The overall objective is to evaluate the various reasons which make the programs more insecure in nature. The review has shown that the conditional operations, pointers and file handling operators can be the security vulnerabilities for any source code or project. To achieve the objectives ten different programs has been considered in this research work. A suitable simulation is done using MATLAB tool to evaluate the effect of different security threads. It has been found that the main security leakage reasons are pointers and file handling operators. But declaration of these operators does not lead us for more density. Because when we run this software in real time the effect of the conditional operator become more critical because it recall each critical operation again and again (file handling and pointers).

In near future some more software will also be considered for experimental purpose. Also one can enhance this model by improving the process of density analysis of each program by using fuzzy set theory.

6. REFRENCES

[1] Nithin Haridas (2007) "Software Engineering – Security as a Process in the SDLC" © SANS Institute Pg No: 1-27

[2] James Purcell (2013) "Defining and Understanding Security in the Software Development Life Cycle"

[3] Premkumar and Stuart (2000) "Software Engineering for Security: a Roadmap" Copyright ACM 1-58113-253-0/00/6, Pg No:227-239

[4] Shreyas (2002) "Software Engineering for Security: Towards Architecting Secure Software" Information and Computer Science Dept. University of California, Irvine CA 92697

[5] Mike Phillips (2003) "Using a Capability Maturity Model to Derive Security Requirements" © SANS Institute 2003

[6] Louise Yngström, Job Asheri Chaula, and Stewart Kowalski (2004) "Security metrics and evaluation of information systems security" Department of Computer and Systems Sciences, Stockholm University/KTH Forum 100, 164 40 Kista, Sweden

[7] Bart De Win, Riccardo, Koen Buyens, Johan Gregoire and Wouter Joosen (2009) "On the secure software development process: CLASP, SDL and Touchpoints compare" see front matter2008 Elsevier B.V. All rights reserved.doi:10.1016/j.infsof.2008.01.010, Pg No.1153-1171

[8] Eric Bodden, Ben Hermann, Johannes Lerch and Mira Mezini (2010) "Reducing human factors in software security architectures"

[9] Francisco José Barreto, Arnaldo Dias Belchior and Adriano Bessa Albuquerque (2010) "Security Engineering Approach to Support Software Security" IEEE 6th World Congress on Services, Pg No. 48-55.

[10] Pavel Nastase, Floarea Nastase and Corina Ionescu (2010) "challenges generated by the implementation of the it standards cobit 4.1, itil v3 and iso/iec 27002 in enterprises"

[11] Gefei Sun, Kenichi Yajima, Junichi Miura, Kai Shi, Yuichi Goto, and Jingde Cheng, (2012) "A Supporting Tool for Creating and Maintaining Security Targets According to ISO/IEC 15408" 978-1-4673-2008-5/12©2012 IEEE Pg No. 745-749

[12] Danielito (2012) dcvizcayno.wordpress.com

[13] Mandal and S. C. Pal (2012) "Investigating and Analyzing the Desired Characteristics of Software Development Lifecycle (SDLC) Models" International journal of software engineering research & practices vol.2, issue 4, ISSN: 2231-2048 e-ISSN: 2231-0320 © RG Education Society (INDIA) Pg No.10-14.