**MINOR PROJECT**

**SYNOPSIS**

**ON**

**SECURE CODE ANALYSER**

**Submitted By**

|  |  |  |  |
| --- | --- | --- | --- |
| BINEEK RAJA | VIKASH ANAND | MOHIT PANDEY | VANDANA SHARMA |
| 500037964 | 500037924 | 500040630 | 500041079 |
|  |  |  |  |

***Under the guidance of***

**Mr. JATIN SETHI**

Assistant Professor

Department of Computer Science & Engineering,



**Department of Computer Science and Engineering,**

**Centre for Information Technology,**

**UNIVERSITY OF PETROLEUM AND ENERGY STUDIES**

**Dehradun-248007**

**2016**

****

**Centre for Information Technology**

**University of Petroleum & Energy Studies, Dehradun**

**Project Proposal Approval Form (2016-17)**

**2**2

**Minor**

**Project Title: SECURE CODE ANALYSER**

**Abstract:**

Secure code analyser, also known as Static Application security testing tool. We use it to find out security flaws. There are many guidelines that a programmer can implement to aid in the prevention of common security bugs in applications which we will follow. There are many guildelines which we can use in all languages but there are also some guidelines that can only work with a specific language. In this secure code Analyser we only work with C , C++ and Java languages . Analysers allow a programmer to remove much of the common vulnerabilities found in code.

**Introduction:**

Secure code analyser, also referred as Static Application Security Testing (SAST) Tools. It is designed to analyze source code and compiled versions of code to help find security flaws. designed to analyze source code or compiled versions of code. There are many guidelines that a programmer can implement to aid in the prevention of common security bugs in applications. Many of these can be applied to any programming language, but some are specific to only one language. In this project we are mainly working on programming languages like C, C++ and Java. There are numerous guidelines and tips that a programmer can implement to aid in the prevention of common security bugs in applications. Analyzers allow a programmer to remove much of the common vulnerabilities found in code. So with the help of Secure code Analyzer we can find out all the flaws in our programs and remove them.

**Problem Statement:**

* To analyse and find vulnerability of our code.
* Provide solutions for the vulnerability of the code that is being analysed.

**Literature Review:**

Static source code analyzers attempt to ﬁnd code sequences that, when executed, could result in buffer overﬂows, resource leaks, or many other security and reliability problems. Source code analyzers are effective at locating a signiﬁcant class of ﬂaws that are not detected by compilers . Most static source code analyzers use the same type of compiler front end that is used to compile code. The use of a compiler front end is only natural because the analyzer takes advantage of preexisting compiler dataﬂow algorithms to perform its bug-ﬁnding mission. The analyzer looks for many types of ﬂaws. It looks for bugs that would normally compile without error or warning. The following is a list of some of the more common errors that a modern static source code analyzer will detect the following:

* Potential NULL pointer dereferences
* Access beyond an allocated area , otherwise known as a buffer overﬂow
* Writes to potentially read-only memory
* Reads of potentially uninitialized objects
* Resource leaks (e.g., memory leaks and ﬁle descriptor leaks)
* Use of memory that has already been deallocated
* Out-of-scope memory usage (e.g., returning the address of an automatic variable from a subroutine)
* Failure to set a return value from a subroutine

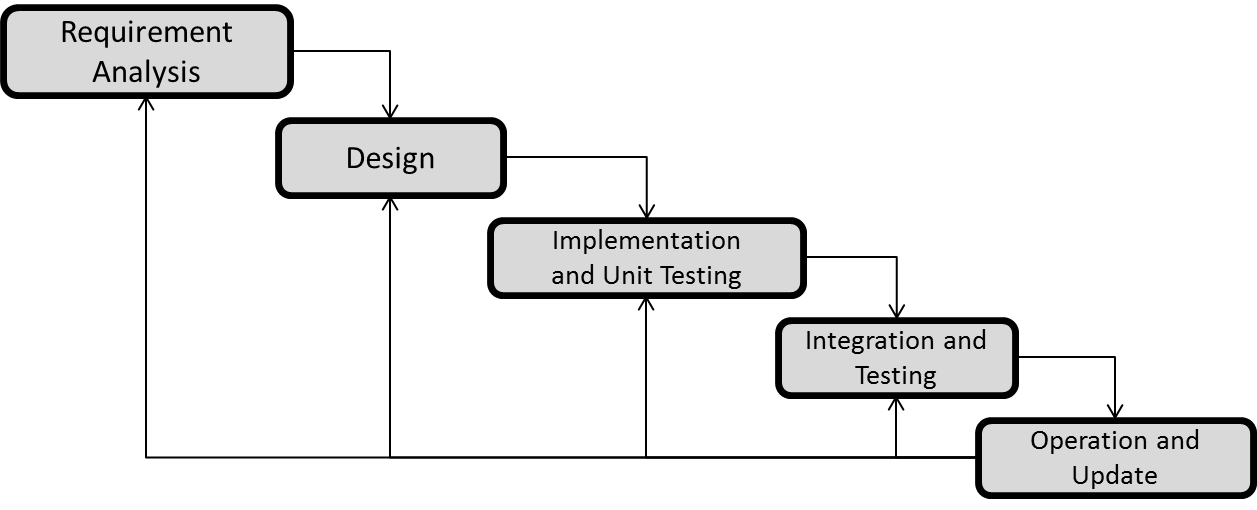
**Objectives:**

* To implement a static application that will Analyse the vulnerability of the code in C, C++ and Java.

**Methodology:**

We are using ITRERATIVE WATERFALL MODEL due to large data size and complexity of our project.

**ITERATIVE WATERFALL MODEL**

****

1. **Requirement Analysis:**

Data modelling i.e. collection of data which will be used as inputs and outputs.

1. **System Design:**An algorithm will be developed on the basis of formulas and data that are collected and will these data will be sampled as input in further phase. The algorithm will decrease the time complexity and will finally give an automated system for all the calculations under a single dashboard.
2. **Implementation and Unit Testing:**

We will develop the application using the algorithm developed during design phase in C language. The application will be developed in two modules and will be tested separately.

1. **Integration and Testing:**

The modules developed will be integrated to develop an application and tested by using the sample data collected during the analysis phase.

1. **Operation and Update:**

The final application developed will be deployed and updated according to the bugs which we will encounter.

**System Requirements:**

* **SOFTWARE REQUIREMENTS**

1. Language – C / C++ , Java
2. Compiler – GCC
3. Operating System – Ubuntu 14.04

* **HARDWARE REQUIREMENTS**
  1. 2 GHz dual core processor or better
  2. 2 GB system memory
  3. 25 GB of free hard drive space

**Schedule: (PERT Chart)**



**References:**

1. The Use and Limitations of Static-Analysis Tools to Improve Software Quality." Anderson, Paul, CrossTalk, Vol. 21, No. 6, (June 2008), pp. 18-21
2. Kratkiewicz Kendra, and Lippmann, workshop on the evaluation on software defect detection tools

**Approved By**

**Mr. JATIN SETHI Mr. S. CHRISTALIN NELSON**

**Project Guide Program Head**