Project Proposal

Submitted to **Dr. Hussain Al-Mohri** & **Mr. Mohammad Souod**

Computer Science Department

College of Sciences

A close up of a sign

Description automatically generatedKuwait University

Course Info

System and Network Security

Group Members

[Mohammad Walid Charrwi] [218122250] {Group Leader}

[Ahmad Bader Humoud] [218122042] {Member}

25th of August 2020

# **problem description**

The problem description the current project is planning to work on involves a formal approach for detecting vulnerabilities in C programs. Vulnerabilities denote faults that may be introduced unintentionally into programs making them behave incorrectly, or even causes these programs to have their transaction and information stored exploited and exposed to any attacker.

* ***Problem Statement****:* The main statement of this problem is to create a C language program scanner that searches and scans C code syntax to highlight insecure main code segments, specific function and system calls and selective variable declaration practices that result in different security vulnerabilities for C code programs that are coded in a specific pattern that exploits the previously mentioned issues. Many of these issue are not intentionally caused by the programmer to create a security vulnerability (although sometimes these vulnerabilities are intentionally implemented to be exploited later) but are the result of missing of implementation knowledge of these C language instructions libraries or not tracking the actual memory addressing during execution of each instruction without implementing any type of exception handler since the C language is considered as one of those languages which are labelled as not ***type safe languages***. Java language for instance, unlike C language, is designed to enforce type safety by introducing what is known as garbage collector. Therefore, to address these issue, our C scanner program will work on discovering these exploits in any fed C program.
* ***Definitions:***

1. **Security Vulnerability:** A security vulnerability is defined as a path or gateway for any attacker that can exploit it to change the execution sequence of instructions for any program or have an unauthorized access to the information exchanged back and forth in the program’s transactions.
2. **Type Safe:** A type safe language is defined as a programming language that enables variable or any memory allocation instructions to accesses only the memory locations it is authorized to access. It also means that any program implemented using these type of languages do not get into any type of undefined state where no further transitions are possible.
3. **Garbage Collector:** A garbage collector is a sub-module of some programming languages that usually acts as a procedure for handling the memory management during and after execution. The use of a garbage collector becomes crucially important after the execution of a segment of code or program is completed without extra effort being done by the programmer to handle this issue. The main task done by a garbage collector is to free different memory segments allocated by a program during execution time and deallocates them. Additionally, it prevents any dangling allocated variables, heaps or stack to be kept in the memory after program execution is completed.
4. **Buffer Overflow:**A buffer overflow Vulnerability is typically caused when a programmer fails to do memory bounds checking when writing data into a fixed length buffer, or does the bounds checking incorrectly. This cause the buffer to result in over-writing the return address of the stack, which the attacker may exploit and insert a custom address that executes a malicious code.
5. **Integer Vulnerabilities:** Most of the errors and vulnerabilities in the manipulation of integers involve insufficient limits checking of the variables storing data of this type. Although it is possible to check if the declared memory allocation variable is equal to the entered value, there is no checking related to the signal of the parameter value. As *malloc* expects arguments of the type **size\_t**, it converts the size value to a large unsigned number which can results in a false statement if compared to a declared positive integer variable.
6. **String Vulnerabilities:** These types of vulnerabilities are defined as the unauthorized root access of string related C standard library function calls and unprecise combination usage of function arguments.
7. **Static Vulnerability:** Reading characters as input from a C programs usually continues until a newline or end of file character is reached, at which point the buffer is terminated with a null character. This means that the programmer has no way to specify the size of the buffer passed to specific C language procedure that deal with pointer to character-returning procedures.

* ***Scope of the Project:***The main scope and domain of this project will be focused on analyzing a specific C language code and scanning for any suspected security vulnerable characteristics that may be exploited by an attacker to manipulate the flow or divert the program to perform malicious activity that is considered harmful.
* ***Objectives:***The objectives of this project will focus on monitoring and highlighting bad techniques of function parameter passing, incorrect variable type assignment, conflict pointer aliasing, risky function and system calls in a given C language code.

# **solution architecture** **Software Development Model.**

The main Software Development Methodology that will be used while implementing this project will include Agile Software Development approach with Scrum approach. Since the nature of this project is viable to continuous changes and adaptiveness based on the professors’ feedback and dependencies that may arise while developing any dependent scanner function, this approach was found the most suitable among many other Agile methods (such as Lean, Kanban, XP, etc.). The main software development model that this project will work on to construct the solution architecture will be the decomposition of the fed C project file into composable, discrete code segment each related to the variable and memory allocation that was performed within that section. In addition, dependent code segment will amended based on the common variables or pointer they access or utilize to avoid faulty pointer aliasing and invalid memory access. Furthermore, the proposed solution architecture is also planned to include a detection module for whether the code provided by the programmer to the C language scanner includes the process of garbage collector for the pointer and heap allocation before end of execution. An additional proposed approach that we will try to implement will involve the detection of any malicious library code import/include in the code at the beginning of the execution, like ***“#include <malicious.h>”***, etc.

# **Reusable components to be used.**

This project will involve using multiple re-usable components but not limited to due to the continuity of the project development as we are still learning about more useful components. These tools, libraries and components include but not limited to the following:

1. ***cqual***which is a tainting analysis to detect errors in the fed code
2. ***SecTAC*** which is a benchmark tool that was designed to evaluate buffer over­flow detection tools, this would help validate our implementation of detecting buffer overflow issues.
3. **nsure++** which is a memory debugger computer program, used validate the project implementation ability to detect various errors in programs written in C language code.

# **platform and programming language.**

The decided language that will be used to build the C-language scanner is the C++ language. The main reason of this choice was the familiarity of this language to both of us as we want to build the fastest project within this very tight time schedule we have to cope with work and studies. Furthermore, the motivation behind choosing this language is due to the similarity of this syntax with the C language itself with a significantly lower vulnerabilities compared to the C. The addition features that C++ introduces especially in terms of type-safe code format discussed previously above is that it promotes virtual functions and templates to achieve polymorphism without void pointers, safer casting operators such as dynamic cast that performs run-time type checking and the prevention of implicit type conversions.

The main platform/IDE that will be used to develop this scanner will be combination of code::blocks and visual studio code. The operating system support will be multi-platform support.