**Final Report**

Submitted to

Computer Science Department

College of Science

Kuwait University

**Course Name & No.**

CS 561 - System and Network Security

**Instructor**

Dr. Hussain Al-Mohri

Mr. Mohammad Souod

**Project Name**

C Vulnerability Scanner

**Group Members**

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

[Ahmad Bader Humoud] [218122042] {**Member**}

**DATE OF SUBMISSION**

27/9/2020

**Table of contents**

[1. Introduction and Background](#_Toc415125542)

[2. System Architecture](#_Toc415125544)

[3. Sofware Development Model](#_Toc415125544)

[4. Implementation Framework and Details](#_Toc415125547)

[5. Tools and Component Reuse](#_Toc415125549)

[6. Conclusions and Lessons Learned](#_Toc415125550)

7. [Screenshots](#_Toc415125552)

8. [GIT link](#_Toc415125552)

8. [References](#_Toc415125552)

# Introduction and Background

*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.*

*the motivations behind developing the C scanner program involves the process of contribution to release a much secure and less vulnerable C program as much as possible to highlight the vulnerabilities and bring the attention of developers to different programming habits that can result in exploits that could be utilized by an attacker to perform malicious action through that C application.*

*The deliverables that were submitted to the GIT respiratory through* [*https://github.com/mwalidshaarawi*](https://github.com/mwalidshaarawi) *involves the incremental deliverables of different Scanner attributes that includes different tracking processes which tries to find and detect different vulnerabilities in the fed C code input.*

*scope of the work includes the set of all C programs that utilized different system and function calls that when call in an inappropriate way result in different system vulnerabilities that may be exploited by an attacker to perform a malicious act.*

# System Architecture

*The system architecture model includes encapsulates a set of different processes that start with capturing a C program file that is pre-compiled and is scanned for all the suspected use of vulnerable function calls until end of the code file is reached. The basic semantics of the C scanner program is based on three main components/phases that represent the high level design of the C scanner system. Initially, the first phase includes preparing the input for the C scanner program that is essential for performing the necessary checks for the fed C program. Once this phase is done, the next phase includes performing an instruction by instruction tracing and analyze different function call and its corresponding components that may contribute in the vulnerability of that instruction to the C program. Each tracked instruction is then marked with the vulnerability exploited and the proposed changes that are recommended to avoid this vulnerability in that specific code segment. Once this done, the C scanner program accumulates the list of dependent calls and declaration in the program and compares whether they were handles correctly in an efficient and secure manner or not. The complete output is then generated once the C scanner program terminates highlighting all the vulnerabilities detected and the exact location they were detected at for efficient tracing and debugging. The below flow chart below represent the high level logical flow of the C scanner program based on different conditions encountered for each instruction scanned through the C scanner program.*

Diagram

Description automatically generated

# Software Developemnt 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.*

# 4.Implementation Framework and Details

*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 algorithm of our project will include scanner using dictionary of defined list of function that would cause vulnerabilities if used in an improper way.*

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

# 5.Tools and Component Reuse

*The components that were used in this project involved using pre-defined libraries that aid the C scanner program to search for the desired vulnerabilities in the scanned C code as an input. Some of the proposed libraries components that were proposed in the project proposal were not usable since most of them were found to be pre-compiled packages and executables that are ready to be executed and not open source code that can be used and is portable among different projects.*

# 6.Conclusions and Lessons Learned

*The proposed C scanner involves using pre-defined knowledge dictionary of a list of known vulnerable functions and system calls that result in different issues within the program.*

*The lessons learned from this project involves the implantation of the different tracking techniques for each function call and the parameters passed and the predicted vulnerability outcome that may result from this call. In addition, the scanning for different dependencies in different segments of the code that involve multiple data pointer references and manipulation during the program execution.*

# 7. [Screenshots OF YOUR RUNS.](#_Toc415125552)

Text

Description automatically generated

Text

Description automatically generatedText

Description automatically generated

# 8. [GIT link](#_Toc415125552)

*The C-scanner project respiratory can be found in the link below:*

[*https://github.com/mwalidshaarawi*](https://github.com/mwalidshaarawi)

# 9. References

CERN (Ed.). Common vulnerabilities guide for C programmers. Retrieved September 26, 2020, from <https://security.web.cern.ch/recommendations/en/codetools/c.shtml>

S.Sonar (Ed.). vulnerabilities for C. Retrieved September 26, 2020, from *https://rules.sonarsource.com/c/type/Vulnerability*

NVD (Ed.). NATIONAL VULNERABILITY DATABASE. Retrieved September 26, 2020, from [*https://nvd.nist.gov/vuln/detail/CVE-2019-11365*](https://nvd.nist.gov/vuln/detail/CVE-2019-11365)

Carnegie Mellon University (Ed.). Software Engineering Institute. Retrieved September 26, 2020, from *https://wiki.sei.cmu.edu/confluence/collector/pages.action?key=c*