Diploma of
Software Development

Subject 2:

# BASIC SECURITY LANGUAGES

## Assessment Workbook
## Security Toolkit in C

Academy of Information Technology delivering as Coder Academy

# Diploma of Software Development
# Assessment Workbook
# Security Toolkit in C

---

# Introduction

## Purpose

This assessment tool forms part of the assessment for a learner working toward an ICT50715 - Diploma of Software Development (**https://training.gov.au/Training/Details/ICT50715**).

This assessment tool covers the following units of competency in their entirety:

- **ICTPRG418** - Apply intermediate programming skills in another language

## Glossary of Terms for Assessment Tasks

> **Unit of competency** - a unit of competency (UoC) is a nationally agreed statement of the skills and knowledge required to perform a particular job or function. A UoC contains elements, performance criteria, knowledge evidence, performance evidence, and foundation skills which dictate what must be assessed. Looking at a UoC on training.gov.au may give you assistance in understanding the evidence requirements for a unit.
>
> **Instructions** - each task will have instructions which explain what you are required to do and how you are to submit each task.
>
> **Context** - each task will have information about what is needed to complete the task or any special conditions to commence the task.
>
> **Decision-making rules** - each task will have a list of decision making rules you must follow to receive a satisfactory result in the task.
>
> **Satisfactory** - a task or sub-task you complete will be deemed as satisfactory if evidence you provide you has followed the assessment instructions according to the decision making rules.
>
> **Not Satisfactory** - a task or sub-task you complete will be deemed as not satisfactory if evidence you provide does not followed the assessment instructions according to the decision making rules.

## Assessment Deadline

This task will be made available to learners in Week 6 of Study Period 1.

The task will be due on the Friday of Week 7 by 20:00.

Submission of the task should be made according to the guidelines provided at the end of each task.

## Assessment Instructions

You must complete all tasks in this assessment to a satisfactory level to be deemed competent in the units of competency.

Please review the assessment checklist at the end of each task to ensure that you have submitted evidence for each task, and that you have submitted your evidence in the format requested.

## Overview of Assessment Methods and Tasks

This summarises the separate tasks which form this assessment. Check the submission requirements and checklist for each task to ensure you have submitted all the required evidence.

| Task | Method of Assessment | Assessment Title | Description |
|---|---|---|---|
| 1 | Work Product | Hex Dump Application | You will implement a hex dump application in C. |
| 2 | Work Product | Key Logger Application | You will implement a key logger application in C. |
| 3 | Work Product | Password Strength Checker Application | You will implement a password strength checker application in C. |
| 4 | Observation | Setup of IDE, Automated Builds and Debugging | You will demonstrate the use of an IDE to automate the building of an application, how to reference multiple files, and the use of debugging tools. |
| 5 | Work Product | Testing | You will write tests of code and document the results to ensure the program meets the specification. |
| 6 | Written Questions | Developer Workbook | You must answer written questions to demonstrate your knowledge. |

# Assessment Policies

## Resubmission Policy

### Unsatisfactory Work

All tasks will be assessed on the basis of being satisfactory or not satisfactory. To be deemed competent for the units covered in this assessment task you must achieve a satisfactory result for all tasks and sub-tasks in this assessment.

Should any aspect of your work be deemed not satisfactory by the assessor you will be provided with feedback and an opportunity to resubmit the assessment ONCE.

Your trainer is committed to supporting you to get a satisfactory result. Please speak to your trainer if you have any concerns or questions regarding the task.

## Plagiarism

Plagiarism of any kind is not allowed and may result in the learner being placed on the plagiarism register, having to re-submit work, or being subject to disciplinary processes.

When submitting your assessment you will be required to verify that what you are submitting is all your own work. Submissions will not be accepted unless the learner has completed a declaration stating that the submitted work is their own work.

Please refer to the Academy of Information Technology website (**http://ait.edu.au**) for detailed policies relating to the submission of assessments, resubmissions, illness/misadventure and plagiarism.

## Disclaimer re Ethical Hacking and Cyber Security

This assessment involves learning about tools and techniques that could be used to exploit systems. Use of knowledge, techniques, and tools acquired in the course of your training cannot be used on a system without express written permission of the owner, and may constitute a criminal offence in Australia and other countries.

As an ethical hacker and cyber security professional, you have a legal and professional responsibility to use your understanding of these tools to harden systems against attack, not exploit them.

It is a requirement of your enrolment that you agree not use any of the knowledge or techniques you learn unethically or unlawfully.

## Late Submissions

Tasks submitted after the Assessment Deadline will not be assessed except in the case of illness or misadventure. To apply for illness or misadventure refer to the Academy of Information Technology website for the policies regarding this. See (http://ait.edu.au) and speak to your trainer.

## Reasonable Adjustment

Where possible, reasonable adjustments to assessment tasks can be made, however, an adjustment cannot affect the integrity of the task. Please discuss with your trainer if you require an adjustment.

# Tasks

## Assessment Overview

As a security professional, you must have a solid grounding in basic tools and concepts using the C language.

In **Tasks 1 - 3** you will design and develop applications that will do the following:

- **Task 1:** The hex-dump application will develop your understanding of both text files and binary files as byte streams, and introduce you to file operations in C.
- **Task 2:** Building a keylogger application will demonstrate a simple type of malware that can communicate with a remote server, employing the string handling facilities of C and aggregate data structures.
- **Task 3:** The password strength checker will introduce you to dictionary lookups as well as pass-by-reference and memory management in C (memory allocation, deallocation, and dynamic arrays).
- **Task 4** will enable you to demonstrate that you are able to setup and utilise an IDE for coding in and testing code using C
- **Task 5** will allow you to demonstrate that you can test the code you have produced
- **Task 6** will require you to demostrate your knowledge of C and application development by answering written questions.

**Before commencing this task, read the design document in Appendix A to see the requirements of the application.**

# Task 1 - Hex Dump Application

## Context

This assessment will require the use of a computer, and access to the internet, and an IDE for developing C applications.

## Instructions

Create a command-line application which creates a hex dump in C according to the design document in **Appendix A - Part A**.

All **variable delcarations**, **structs** and **control structures** must have documentation in the form of block or inline comments in the code.

## Decision Making Rules

- all features in the design document in **Appendix A - Part A** must be implemented using C
- all code submitted must be formatted according to the "One True Brace Style" convention for indentation and braces
- all **variable delcarations**, **structs** and **control structures** must be documented.

# Task 2 - Key Logging Application

## Context

This assessment will require the use of a computer, and access to the internet, and an IDE for developing C applications.

## Instructions

Create a command-line application which logs key press events according to the design document in **Appendix A - Part B**.

All **variable delcarations**, **structs** and **control structures** must have documentation in the form of block or inline comments in the code.

## Decision Making Rules

- all features in the design document in **Appendix A - Part B** must be implemented using C
- all code submitted must be formatted according to the "One True Brace Style" convention for indentation and braces
- all **variable delcarations**, **structs** and **control structures** must be documented.

# Task 3 - Password Strength Checker Application

## Context

This assessment will require the use of a computer, and access to the internet, and an IDE for developing C applications.

## Instructions

Create a command-line application which checks the strength of a password according to the design document in **Appendix A - Part C**.

All **variable delcarations**, **structs** and **control structures** must have documentation in the form of block or inline comments in the code.

## Decision Making Rules

- all features in the design document in **Appendix A - Part C** must be implemented using C
- all code submitted must be formatted according to the "One True Brace Style" convention for indentation and braces
- all **variable declarations**, **structs** and **control structures** must be documented.

# Task 4 - Setup of IDE, Automated Builds and Debugging

## Context

This assessment will require the use of a computer, and access to the internet, and an integrated development environment (IDE) to use for the coding in this assessment. The recommended IDE to use is CLion (https://www.jetbrains.com/clion/).

## Instructions

Arrange a time with your trainer to demonstrate:

1. the use of a `makefile` to automate the building of **ONE** command-line application using an integrated development environment.
2. how multiple source files can be accessed in a single file.
3. the use of a debugging tool to trace the code execution by placing at least **TWO** breakpoints using **ONE** of the applications developed as part of Tasks 1, 2 or 3.
4. the use of a debugging tool to inspect the contents of an array OR other variable in **ONE** of the applications developed as part of Tasks 1, 2, or 3.

### Decision Making Rules

- all items on the observation checklist must be answered in the affirmative by the assessor to be found satisfactory for Task 1

## Observation Checklist

The following checklist will be used by your trainer to assess whether you have satisfactorily completed **Part A**:

| No. | The learner demonstrated... | Yes | No | Comment (optional) |
|---|---|---|---|---|
| 1 | the use of an IDE to automated building of ONE command-line application using an integrated development environment | | | |
| 2 | how multiple source files can be accessed in a single file by adding two or more imports of other source files to the top of the file | | | |
| 3 | the use of a debugging tool to trace code by setting at least **TWO** breakpoints using **ONE** of the applications developed as part of Tasks 1, 2 or 3. | | | |
| 4 | the use of a debugging tool to inspect the contents of an array OR other variable in **ONE** of the applications developed as part of Tasks 1, 2, or 3. | | | |

| Part A | Satisfactory | Not Satisfactory | Comment |
|---|---|---|---|
| The learner has satisfactorily completed Task 1? | | | |

**Name of Assessor:** _____

**Date/Time of Observation:** _____

# Task 5 - Testing a C Application

## Context

This assessment will require the use of a computer, and access to the internet, and an integrated development environment (IDE) to use for the coding in this assessment. The recommended IDE to use is CLion (https://www.jetbrains.com/clion/).

## Instructions

Write a unit test for the hexdump application you have developed in Task 1.

1. **Design** a test to verify that the hexdump is working correctly.
   Your test must:
   - output the results of your test to a text file

   You must run your test with the following inputs:
   - **ONE text file** that is larger than 400 bytes and smaller than 4KB
   - **ONE binary file** such as an image or sound file that is larger than 400 bytes and smaller than 4KB
2. **Analyse** the output from each test and **explain** how you verified the correctness of the results for each file tested (150 - 200 words per file tested).

## Submission Instructions

- The test written in C and text files generated for this task should be submitted with the source control repository for Tasks 1-3.
- Files relating to the test should be in their own folder named `Task-5-Testing`.
- The written parts of this task must be submitted in markdown or as a PDF with the source control repository for Tasks 1-3 in the `Task-5-Testing` folder.

## Decision Making Rules

To achieve a satisfactory result for this task you must:

- follow all instructions and provide the required evidence according to the submission instructions
- meet any word requirements specified in the question

# Task 6 - Developer Workbook

## Context

This assessment will require the use of a computer and access to the internet.

## Decision Making Rules

- You must provide satisfactory answers to all questions
- You must meet any word requirements specified in the question

## Questions

| Question | Answer | S / NS* | Comment |
|---|---|---|---|
| 1. **Describe** the properties and function of dynamic variables in C. | | | |
| 2. **Describe** medium-size application development in detail with regard to:<br>- dividing source code into multiple files<br>- the use of header files in C<br>- the use of C libraries | | | |
| 3. **Describe** the properties and function of arrays in C. | | | |
| 4. **Describe** the properties and function of file handling in C. | | | |
| 5. **Describe** the properties and function of user-defined data structures in C. | | | |
| 6. **Describe** TWO development methodologies appropriate for the development of small tools such as the ones implemented in this assessment task. | | | |
| 7. **Outline** the develoment of the C language, making reference to the types of software C is intended for. | | | |

## Submission Instructions

The Developer Handbook must be:

- a PDF document that conforms to the requirments in the **Submission of Written Tasks** in the Assessment Policies section of this document.
- submitted via the Canvas LMS platform

## Assessment Checklist

- [ ] I have completed all questions
- [ ] I have met the word requirements for all questions
- [ ] I have submitted my developer workbook to the trainer

# Appendix A - Simple Design Document

## Part A - Hex Dump Application

A hex dump application must:

- run in the command line.
- accept a path as a command-line argument using `argv[1]`.
- use the `fopen()` and `fclose()` functions to open and close a file.
- use `fread()` to read one chunk of the file at a time into a 2-dimensional array.
- loop over the array and display hexadecimal values on the left of the screen and ASCII on the right side of the screen.
- when the command prompt is full, the printing of values should pause, and then display another page of values on pressing the `enter` or `return` key.
- use output redirection to capture the output of the application into files called, respectively, ‘textOutput.txt’ and ‘binaryOutput.txt’.
- have an appropriate `Makefile` to build an executable called `hexdump`.

## Part B - Keylogging Application

The keylogger must:

- use a `struct`, to store an event that contains an ASCII keycode and a `struct timeval` from `sys/time.h`.
- use an array to store 30 keypresses.
- assemble all 30 keystrokes and their respective times into a single command string using `sprintf()`, once they have all been collected.
- execute the command string using `system()` to send the log to a server.
- have an appropriate `Makefile` to build an executable called `keylogger`.

## Part C - Password Strength Checker

Obtain a dictionary file from the trainer.

The password strength checker must:

### General

- run in the command line.

### Inputs

- accept a path to a dictionary as a command-line argument using `argv[1]`.
- prompt the user to enter a password to be checked using `getline()` once the dictionary has been loaded

### File Access and Parsing

- use `fopen()` and `fclose()` to open and close the dictionary file.
- read the file using `fgetc()` and establish the number of words it contains.
- use `malloc()` to allocate an array to hold all the words in the dictionary
- use `getline()` to parse the dictionary and insert each line into the array

### Sorting

- have a comparison function to use with `qsort()` to sort the array

### Evaluation of Password and Feedback

The evaluation of the password must:

- read the password from the user using `getline()`
- use a function in a separate file which accepts a password, a dictionary, and the length of the dictionary
- use an algorithm to evaluate the password's strength and display an appropriate message, within this function
- use binary search to ensure the password is not a dictionary word, within this function
- use `free()` to release the memory at the end of the evaluation
- prompt the user to enter another password for evaluation
- if an empty password is supplied the program should terminate