SYST44288

Assignment 4 - Security Code Audit

Start date: December 22, 2017

Due date: January 4, 2018

Members: Joshua Mallari, Jaekuk Song

**1. Overall discussion of John The Ripper, WHAT it does**

This report will make an attempt to explore an information security utility called **John The Ripper**. **John The Ripper**, or simply known as the program "**john**", is a password cracking program made to crack encrypted hashes into their clear-text forms.

**2. HOW does the program work/HOW does it function, and what is the core functionality - system calls, concurrency, features**

**3. Overview: Describe key source codes, what do each do, how is it organized i.e key parts of cracker.c, john.c, math.c, etc**

key source codes -

**john.c** - contains the main source code i.e. the int main(). Also contains the code for OS process creation, and concurrency.

john.c has 2 functions that are used to deal with processes. john\_fork is the function used to create child processes

john\_wait is the function used for parent processes waiting on children processes to terminate.

First in the main function, it calls john\_init(), john\_run(), and john\_done() sequentially. Starting from john\_init(), this function initiates the entire process of the program. The function calls a function john\_load(), which deals with database queue, and as it checks for necessary requirements. Once done it will call the john\_fork() function to create a child process. By default the parent process john\_main\_process is set to 1, and when john\_fork() is called by creation of the child process, john\_main\_process is set to 0, while the pid is set to how many processes are needed and then given to john\_child\_pids. Once that is all done john\_init() exits and enters the next function john\_run(). This function is the function that performs the expected decryption operations on hashes by using the flags corresponding to the options of options.c class. This depends on what the user enters for the option methods. After this john\_run() calls the function john\_wait() to wait to terminate the children processes, and giving control back to the parent process. john\_wait() uses the variable john\_child\_count to determine how many child processes are being waited to be terminated.

**options.c** - This c file deals with defining the major options available, by the means of using structs and its data types as flags such as --single, --wordlist, --external, and more when using john the ripper. The opt\_init() function takes the name of the option from the terminal as the parameter, and performs checks (i.e. FLG\_CRACKING\_CHK) in if/else methods. From the if/else statements the john\_fork() function is called and the flag "fork" is raised and returned to the main code.

format.c -

**AFS\_fmt.c**, **BSDI\_fmt.c**, **DES\_fmt.c**, **MD5\_fmt.c** - These are few of many c files that contain code for password cracking algorithm. Each c file is written for a specific hash format and has different methods of processing hashes. For example, each hash format has different characters for comparison when validating if the inputted hash is indeed the correct hash format the user stated as. For MD5 the hash string is compared to the characters "$1$", and "$apr1$", and for AFS DES the hash string is compared to the characters "$K4$".

These c files are called and used by files, such as john.c and format.c, in order to test the format.

**cracker.c** - This c file deals with providing salts and hashing of the passwords. This code comes into play when the program requires anything related to salts and hashing such as using salt to hash a password, cracking a hashed passwords by using a hash table values, updating a cracked password in a database, and more. This c file and its functions are connected to the encryption, decryption parts of the program such as AFS\_fmt.c, and DES\_fmt.c.

**\*Make sure to include screenshots of code when appropriate**