# HEALTHNET

Group 3

Andrew Hill
Catilina Self
Amy Stewart
Lance Fisher
Patrick Taylor

Revisions:

| Revision | Date | Author(s) | Remarks |
|---|---|---|---|
| 1.0 | 25 April 2016 | Lance | Initial Writing |
| 2.0 | 29 April 2016 | Cat | Edited and Finalized for Submission |
| 2.5 | 29 April 2016 | Andrew | Finalized Testing Cases |
| | | | |
| | | | |

**TABLE OF CONTENTS**

# 1 INTRODUCTION

## 1.1 System Overview

The product to be delivered entails a secure means of storing the medical records of patients for the client hospital. In addition to covering the listed client specified requirements, we will also be addressing a number of security concerns that could arise based on the features desired by the client. This program will assist in the storage of files and information associated with patients, nurses, and doctors. It will allow for tracking of all relevant medical data as well as maintain Health Insurance Portability and Accountability Act (HIPAA) standards on personal information.

The following use and abuse cases will detail specifications given by the prospective client based on the supplied HealthIT.gov website. This project will cover the Stage 1 Core Objectives and the Menu Objective requirements.

## 1.2 Test Approach

The product is a standalone application without remote access, therefore many vectors of attack are not applicable. This system is therefore going to be tested with an input validation and safe coding approach. With this approach common mistakes in coding will be scanned for using open source tools from a reliable and trusted source, CERT. The 'rosecheckers' application is able to scan source code files and spot common mistakes in buffer management, integer use, and the use of unsafe functions.

# 2 TEST PLAN

## 2.1 Features to be Tested

Authentication checks
Availability of unauthorized information
Malformed input / No input
Accessibility of elevated permissions to unauthorized users

## 2.2 Features Not to be Tested

Local file based password deduction
Testing Tools and Environment:
Rosecheckers 0.7, a distribution developed by CERT
Debian 4
gcc/g++ 4.4.5
gdb 7.0.1
CERT rose 0.9.5a-15163
Eclipse Indigo
Sublime Text
gitHub

# 3 TEST CASES

## 3.1 Case 1

### 3.1.1 Purpose

This case tests whether an incorrect password may be used to gain entry to the system.

### 3.1.2 Inputs

The user name to a known doctor is inputted to the application. An incorrect password is also supplied. This is next attempted with an incorrect user and correct password.

### 3.1.3 Expected Outputs & Pass/Fail Criteria

The system should respond with an error. This informs the user that the login credentials used were incorrect. The system will not divulge whether the user name or the password or both were incorrect. This case will fail if the system grants access to the user.

### 3.1.4 Test Procedure

The application will be started. At the login screen, the user will present the username of the doctor user to the application. An incorrect password is supplied. The system will give a response.

The application will be started. At the login screen, the user will present the misspelled username of the doctor user to the application. The correct password is supplied. The system will give a response.

## 3.2 Case 2

### 3.2.1 Purpose

This case tests whether an unauthorized user can access data owned by other users.

### 3.2.2 Inputs

A user without access to individual medical records will attempt to view a patient's records.

### 3.2.3 Expected Outputs & Pass/Fail Criteria

The system should respond with an error. This informs the user that the function is not available to them. The system will not divulge whether the patient records exist. This case will fail if the system grants access to the user.

### 3.2.4 Test Procedure

The application will be started. At the login screen, the user will present the username of the 'patient' user to the application. A correct password is supplied. The 'patient' user will attempt to

access patient records that are not their own. The system will give a response.

## 3.3 Case 3

### 3.3.1 Purpose

This case tests whether malformed or missing input will cause the application to crash or divulge sensitive data.

### 3.3.2 Inputs

Inputs of varying length are used at different places in the application.

### 3.3.4 Expected Outputs & Pass/Fail Criteria

The system should respond with an error. This informs the user that the input was not understood. The previous prompt should be displayed. No further action should be taken by the application, otherwise the case is deemed a failure.

### 3.3.5 Test Procedure

The application will be started. At the login screen, the user will present a username that consists of a very long string. Afterwards, a blank password will be entered.

This procedure will be repeated throughout the application, with several types of input attempted. The rosecheckers application will be used to scan the sources for mistakes in input buffer management.

## 3.4 Case 4

### 3.4.1 Purpose

This case tests whether an unauthorized user can perform administrative tasks.

### 3.4.2 Inputs

A user without administrative permissions will attempt to delete a user. This user will also attempt to add a user.

### 3.4.3 Expected Outputs & Pass/Fail Criteria

The system should respond with an error. This informs the user that the function is not available to them. The system will not divulge whether the user exists or not. This case will fail if the system performs the deletion of addition of users.

### 3.4.4 Test Procedure

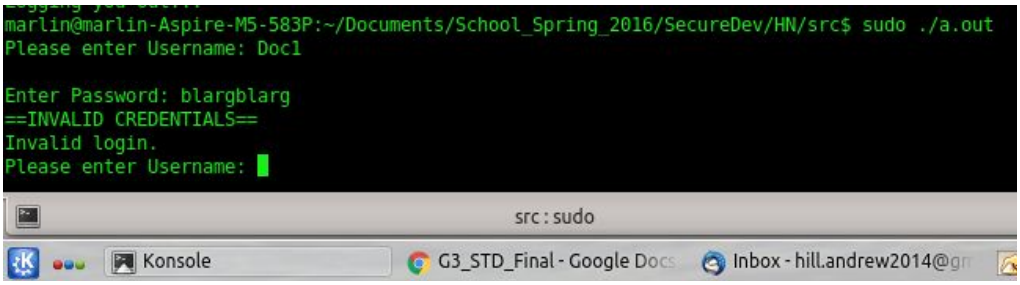
The application will be started. At the login screen, the user will present the username of the ‘patient’ user to the application. A correct password is supplied. The ‘patient’ user will attempt to delete another user of the system. The system will give a response.

# 4 ADDITIONAL MATERIAL

## 4.1 Log for Case 1

### 4.1.1 Test Results

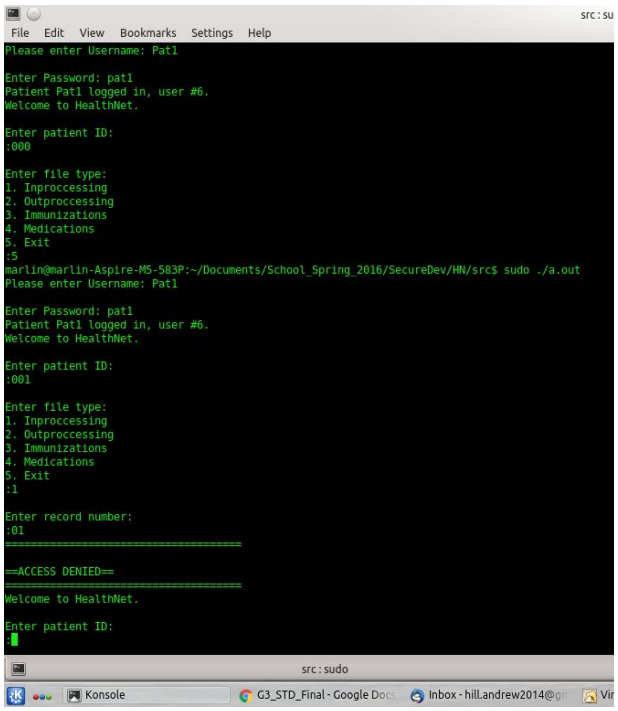
Test Case 1 was a basic verification to check that our password system was functioning properly. The test entailed using a valid username, with an invalid password. The test passed, and access was denied with an invalid password.




## 4.2 Log for Case 2

### 4.1.1 Test Results

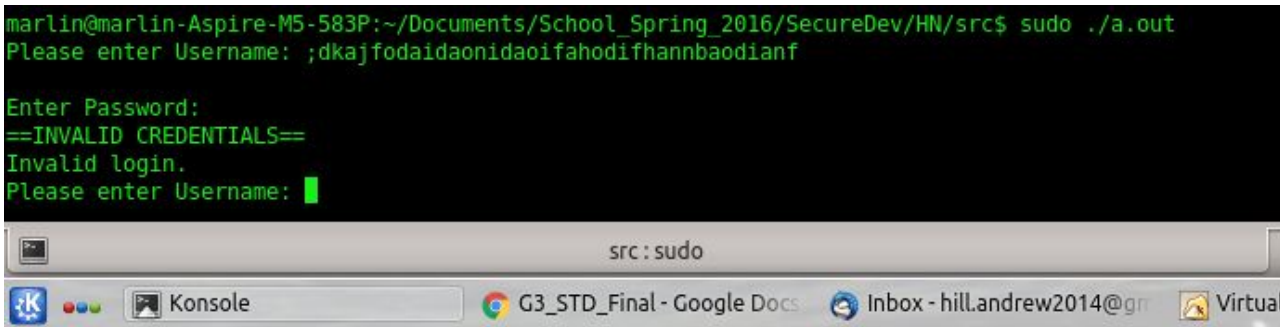
Test Case 2 verified that "patient" users only had access to files they were authorized to access. The tested logged in as a test patient and attempted to access the files of another patient. The Test passed, and access was denied. The program uses checks to ensure that the only valid response from the user, is a filename that includes the patient ID that matches the one associated with their profile.



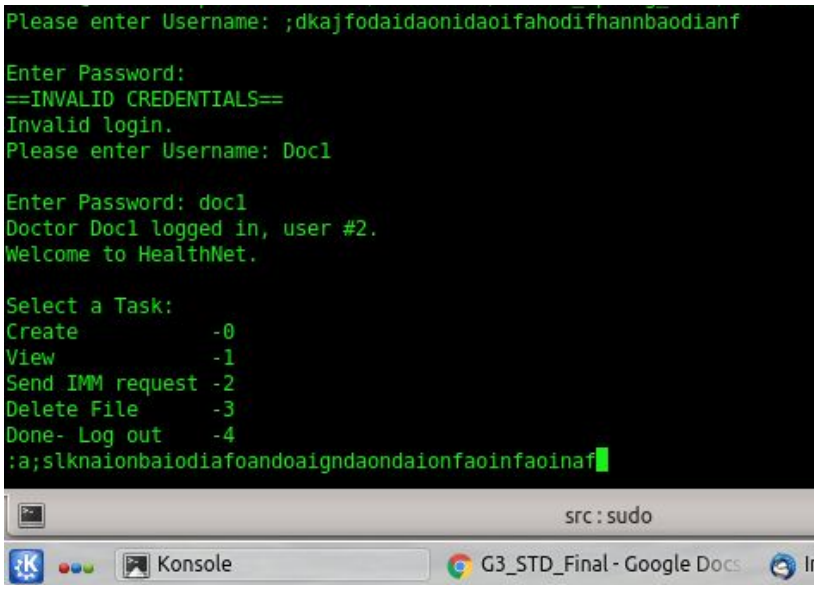
4.3 Log for Case 3

4.1.1 Test Results

Test Case 3 was designed to verify that long strings input into the text fields (such strings common for inserting malicious code) resulted in triggering our safeguards, and prevented further progression of the code, until a valid response was given. When the test procedure was performed on sections expecting string input, the test passed:




However, when the test was performed on input sections that were expecting integers, it failed:



```
File  Edit  View  Bookmarks  Settings  Help
:
Improper Input, type number of selection and press enter...
Enter record number:
:
incorrect fileType
incorrect fileType
NULL filePointer
unable to create ".././patients/000/00000"
unable to create ".././patients/000/000Pv6Pv600"
Improper Input, type number of selection and press enter...
Select a Task:
Create           -0
View             -1
Send IMM request -2
Delete File      -3
Done- Log out    -4
:
Improper Input, type number of selection and press enter...
Enter patient ID:
:
Improper Input, type number of selection and press enter...
Enter file type:
1. Inproccessing
2. Outproccessing
3. Immunizations
4. Medications
:
Improper Input, type number of selection and press enter...
Enter record number:
:
incorrect fileType
incorrect fileType
NULL filePointer
unable to create ".././patients/000/00000"
unable to create ".././patients/000/000Pv6Pv600"
Improper Input, type number of selection and press enter...
Select a Task:
Create           -0
View             -1
Send IMM request -2
Delete File      -3
Done- Log out    -4
^C:
marlin@marlin-Aspire-M5-583P:~/Documents/School_Spring_2016/SecureDev/HN/src$
src : bash
Konsole   G3_STD_Final - Google Docs   Inbox - hill.andrew2014@gr
```

## 4.4 Log for Case 4

### 4.1.1 Test Results

Test Case 4 checked to ensure that users did not have access to admin level privileges, and could not elevate themselves to admin level status. The design and layout of the code was such that the program would only display commands that were associated with their level of access. This was tied to the account itself, and could not be changed. The test passed, as the user was unable to delete, or even access files that were not associated with their account.