Table of Content

Part One:
1. *Describe how a honeypot could be used to detect spam, and aid spam filters on ”real systems”.*
2. *In what way can personalised login screens provide protection? Explain your answer carefully.*

Part Two:
1. *INTRODUCTION*
2. *INSTRUCTIONS*
   - 2.1 Python version
   - 2.2 Running the program
   - 2.3 File storing method
     - 2.3.1 filterPolicy.txt
     - 2.3.2 alertPolicy.txt
     - 2.3.3 log.txt
     - 2.3.4 alert.txt
3. *SOURCES CODE*
   - 3.1 Library - Uses of Library makes programming easier
   - 3.2 main() - Main Menu of the program
   - 3.3 Filter
     - 3.3.1 fil() - Filter Menu for choosing the Filter-related option
     - 3.3.2 show() - Show all Filter Policy
     - 3.3.3 add() - Add new Filter Policy
     - 3.3.4 edit() - Edit existing Filter Policy
     - 3.3.5 delete() - Delete unwanted Filter Policy
     - 3.3.6 Filter() - Decide which Filter option to be used
     - 3.3.7 rangePort() - Handle port in a range
     - 3.3.8 allPort() - Handle all port from port folders
     - 3.3.9 onePort() - Handle only one port
   - 3.4 Analyzer
     - 3.4.1 alert() - Alert Menu for choosing Analyzer-related option
     - 3.4.2 show() - Show all Alert Policy
     - 3.4.3 add() - Add new Alert Policy
     - 3.4.4 edit() - Edit existing Alert Policy
     - 3.4.5 delete() - Delete unwanted Alert Policy
     - 3.4.6 Analyzer() - Analyze log.txt with Alert Policy
4. *SAMPLES OF EXECUTION*
   - 4.1 Show Policy
   - 4.2 Add Policy
5. *CONCLUSION*

Part One:

1. **Describe how a honeypot could be used to detect spam, and aid spam filters on ”real systems”.**

Answer:
System Administrator can create a mailing system or address which is used to receive spam mail from same email address or same IP address, which mean the system or mail address normally won’t receive any email from clients or potential consumer. So that, system administrator or the mailing system itself can detect the numbers of time particular email or IP address emailed to the system, and decide whether to mark the email as spam or block particular IP address.

2. **In what way can personalised login screens provide protection? Explain your answer carefully.**

Answer:
User can easily differentiate the real sites and phishing sites, as phishing sites can’t provide exactly same login screens after the user entered the ID, this technique is widely used in Online Banking System. But if the database compromised or there is APIs to access to the personalised login screens, phishing website can easily fake the personalised login screens by request the data from the database itself.

Part Two:

1. INTRODUCTION

This assessment is about Intrusion Detection System, the author is required to build a simple console program to add/delete/edit/show rules or cases for the two-step verification of the activity might be abnormal or intrusion behaviour. The first-step used Filter to scan the port with rules and generate log.txt, and then the second step Analysis the log.txt with cases and generate alert.txt .

2. INSTRUCTIONS

2.1. Python version

The python version using in this assignment is v2.7(more information: docs.python.org/2.7/). Even though the Python v3.4 is stable release, but most Linux Operating System come with v2.7 by default, that's why in the author chose to code in v2.7 in order to prevent incompatible issues. As there is a few changes which might affect the whole program. For examples, print statement is v3.4 behave as a function print().

2.2. Running the program

For this program, the program itself will auto-compile since the user first time use the program.
To start the menu-driven program, user can input following line in terminal,

"python main.py"

But there is a shortcut to run the Filter or Analyzer,

Filter, ***"python main.py -f <startPort> <endPort>"***
"python main.py -f 4000 4020"

Analyzer, ***"python main.py -a"***

2.3. File storing method

2.3.1. filterPolicy.txt

The file stored at ~resources/filterPolicy.txt,
It represented as, PORT:USER:ACCESS:TRIGGER:OPTION

"4013:any:rwx:5:all"

This file is used to store the filter policy data set by user, user can simply add new, edit or delete the policy based on the user requirement.

2.3.2. alertPolicy.txt

The file stored at ~resources/alertPolicy.txt,
It represented as, PORT/RULE:USER:ACCESS:TRIGGER:OPTION

"3:any:rwx:2:day"

This file is used to store the alert policy data set by user, user can simply add new, edit or delete the policy based on the user requirement

2.3.3. log.txt

The file stored at ~log.txt,
It represented as, RULE,PORT,OPTION|USER,READ,WRITE,EXECUTE: USER, ...:

"3,4000,day|A,1,2,5:B,2,3,4:...Z:0,5,0:"

This file is auto generated by the Filter() function, the logged data can be make use by Analyzer to create useful Alert.

2.3.4. alert.txt

The file stored at ~alert.txt,
It represented as, USER:ACTION:TRIGGER:PORT:DAY

"user:X, action: write exceed 3 on port 4017 in day 7"

This file is generated by Analyzer() function from the log.txt file, it is used to store the abnormal activity in the list of port over 10 days.

3. SOURCES CODE

3.1. Library - Uses of Library makes programming easier

```
import sys #import standard library for direct contact with command line, passing arguments and terminate program
from time import sleep #to pause system for awhile
```

3.2. main() - Main Menu of the program

```
def main():
    #when there is a system argument, and the argument is -f go to Filter Menu,
    #if the arugment is -a then go to Analyzer menu
    if len(sys.argv) >=2:
        if sys.argv[1] == '-f':
            Filter(sys.argv[2], sys.argv[3])
        elif sys.argv[1] == '-a':
            Analyzer()
        else:
            print "Invalid arguments, SYSTEM TERMINATED!"
            sys.exit();
    #if no default argument, then show the menu
    #the menu is conducted in infinite loop,
    #user can either close it or input valid data to proceed
    while True:
        #show the label of the menu, in this case Main Menu
        print "##############################################"
        print "##\t\tMain Menu\t\t##"
        print "##############################################"
        #show the option of the menu
        print "1. Filter Menu"
        print "2. Analyzer Menu"
        print "0. Exit"
        #require input from user to select option
        option = raw_input("Option: ")
        if option == '1':
            fil() #go to Filter menu if selected 1
        elif option == '2':
            alert() #go to Analyzer menu if selected 2
        elif option == '0':
            sys.exit() #terminated the program when user input 0
        else:
            #any other input consider as wrong input, ask user to re-input
            print "Wrong input, please try again."
```

3.3. Filter

3.3.1. fil() - Filter Menu for choosing the Filter-related option

```
def fil():
    menu = True #temporary turn on the menu
    while menu:
        #for formating purpose only
        print "####################################"
        print "##\t\tFilter\t\t##"
        print "####################################"
        #show menu option
        print "1. Show Filter Policy"
        print "2. Add Filter Policy"
        print "3. Edit Filter Policy"
        print "4. Delete Filter Policy"
        print "5. Run Filter"
        print "9. Back to Main Menu"
        print "0. Exit"

        option = raw_input("Option: ") #get input to choose option
        if option == '1':
            show() #display all Filter policy information
        elif option == '2':
            add()   #add new Filter policy
        elif option == '3':
            edit()   #edit current Filter policy
        elif option == '4':
            delete()        #delete Filter policy
        elif option == '5':
            fr = raw_input("From port: ")
            to = raw_input("To port: ")
```

```
                Filter(fr, to) #call Filter to filter the normal data of port file
                menu = False
        elif option == '9':
                menu = False #set Menu to False, so it will back to Main()
        elif option == '0':
                sys.exit() #terminated program
        else:
                print "Wrong input, please try again."
```

3.3.2. show() - Show all Filter Policy

```
def show():
        print "\nShow Filter Policy,"
        #open policy file as read, and store it into policy[]
        read = open("resources/filterPolicy.txt", "r")
        policy = []
        for line in read:
                policy.append(line.strip().split(':'))
        #print out the policy[] one by one
        for row in xrange(len(policy)):
                print "Rule", row+1
                print "Port:", policy[row][0]
                print "User:", policy[row][1]
                print "Action:", policy[row][2]
                print "Threshold:", policy[row][3], '\n'
        read.close() #close file
```

3.3.3. add() - Add new Filter Policy

```
def add():
        #request user to input required data one by one
        print "Add new Filter Policy,"
        print "input '1' for all port\nPort Number for specific port, e.g. 4013\nRange port use '-',
e.g. 4010-4012"
        port = raw_input("Port: ")
        print "Enter specific username or 'any' for all user"
        user = raw_input("User: ")
        print "r:read\nw:write\nx:execute"
        action = raw_input("Action: ")
        print "At least how many time the action occur for specific user"
        threshold = raw_input("Threshold: ")

        #open policy file and save append the data into it
        with open("resources/filterPolicy.txt", "a") as fil:
                fil.write( '\n' + port + ':' + user + ':' + action + ':' + threshold)
        sleep(0.2) #system pause
        print "The policy is successfully added." #print success
```

3.3.4. edit() - Edit existing Filter Policy

```
def edit():
        #open policy file and save the data into policy[]
        print "Edit Filter Policy,"
        policy = []
        with open("resources/filterPolicy.txt", "r") as fil:
                for line in fil:
                        policy.append(line.strip().split(':'))

        #request user to input the rules number for edit
        edit = 0 #set for validate purpose only
        while not edit in xrange(1, len(policy)+1):
                edit = raw_input("Rules: ")
                edit = int(edit)
        edit -=1 #list start with 0, not 1
        print "input '1' for all port\nPort Number for specific port, e.g. 4013\nRange port use '-',
e.g. 4010-4012"
        policy[edit][0] = raw_input("Port: ")
        print "Enter specific username or 'any' for all user"
        policy[edit][1] = raw_input("User: ")
        print "r:read\nw:write\nx:execute"
        policy[edit][2] = raw_input("Action: ")
        print "At least how many time the action occur for specific user"
        policy[edit][3] = raw_input("Threshold: ")

        #open file to rewrite all policy into it, include the edited policy
        with open("resources/filterPolicy.txt", "w") as fil:
                for row in xrange(len(policy)):
                        fil.write(policy[row][0] + ':' + policy[row][1] + ':' + policy[row][2] + ':' +
policy[row][3])
                        if row < len(policy)-1:
                                fil.write('\n')
        sleep(0.2) #system pause
        print "The policy is successfully updated." #print success
```

3.3.5. delete() - Delete unwanted Filter Policy

```
def delete():
        print "Delete Filter Policy,"
        #read all policy into policy[]
        policy = []
        with open("resources/filterPolicy.txt", "r") as fil:
                for line in fil:
                        policy.append(line.strip().split(':'))

        #request user to input the policy number wanted to be delete
        delete = 0
        while not delete in xrange(1, len(policy)+1):
                delete = raw_input("Rules: ")
                delete = int(delete)
        delete-=1 #list start with 0, not 1

        #open file and rewrite all policy into it except the policy which is decided to be delete
        with open("resources/filterPolicy.txt", "w") as fil:
                for row in xrange(len(policy)):
                        if row != delete:
                                if row != 0:
                                        fil.write('\n')
                                fil.write(policy[row][0] + ':' + policy[row][1] + ':' + policy[row][2] + ':'
+ policy[row][3])
        sleep(0.2)
        print "The policy is successfully updated."
```

3.3.6. Filter() - Decide which Filter option to be used

```
def Filter(fr = "4000", to = "4020"):
        #if the user input is smaller than 4000 and/or larger than 4020, set it to correct value
        if fr < "4000":
                fr = "4000"
        if to > "4020":
                to = "4020"
        #open the filter policy filter and get the policy as policy
        print "Filter port activity with rules,"
        policy = []
        with open("resources/filterPolicy.txt", "r") as fil:
                for line in fil:
                        policy.append(line.strip().split(':'))

        open("log.txt", "w").close() #clear data in log.txt

        #depend on the policy rule port type, call respective function
        for rule in xrange(len(policy)):
                if policy[rule][0] == '1':
                        allPort(policy[rule], fr, to, rule)
                elif '-' in policy[rule][0]:
                        rangePort(policy[rule], fr, to, rule)
                elif fr <= policy[rule][0] <= to:
                        onePort(policy[rule], rule)
```

```
		else: #if the port less than 4000 or greater than 4020, rule of the port invalid
			print "Rule", rule+1, ", port not in range."
```

3.3.7. rangePort() - Handle port in a range

```
def rangePort(policy, fr, to, rule):
	port= policy[0].split('-') #spilt the word in policy[0]
	#check whether the port is valid
	valid = True
	if port[0] > port[1]: #if the range of the port is greater to lower, swap it
		port[0],port[1] = port[1],port[0]
	if port[0] >= to: #if the lower port greater than greatest port in list, valid False
		valid = False
	if port[1] <= fr: #if the greater port smaller than lowest port in list, valid False
		valid = False

	if valid == True: #set the fr to lower port, to to greater port and pass to allPort()
		allPort(policy, port[0], port[1], rule)
	else: #rules invalid
		print "Rule", rule+1, ", port not in range."
```

3.3.8. allPort() - Handle all port from port folders

```
def allPort(policy, fr, to, rule):
	#for each port in the port list(fr, to), run the onePort() function
	for port in range(int(fr), int(to)+1):
		policy[0] = str(port) #set policy[0] to the port number instead of 1
		onePort(policy, rule)
```

3.3.9. onePort() - Handle only one port

```
def onePort(policy, rule):
	#open specificied port and read all the information into details
	fileName = "resources/port/Port" +policy[0]+ ".txt"
	details = []
	with open(fileName, "r") as read:
		for line in read:
			details.append(line.strip())

	ID = 'ABCDEFGHIJKLMXZ' #for username

	for row in xrange(len(details)):
		user = list(ID) #create an array based on ID
		#for each name in array, add Read,Write,Execute counter
		for name in xrange(len(user)):
			user[name] = [user[name], 0, 0, 0]

		#record the Read,Write,Execute in a day of a port
```

```
                for col in xrange(len(details[row])):
                        for name in xrange(len(user)):
                                if details[row][col] == user[name][0]:
                                        if details[row][col+1] == 'r' and 'r' in policy[2]:
                                                user[name][1]+=1
                                        elif details[row][col+1] == 'w' and 'w' in policy[2]:
                                                user[name][2]+=1
                                        elif details[row][col+1] == 'x' and 'x' in policy[2]:
                                                user[name][3]+=1

                #check the total read, write and execute time in a day of a port
                #if the total less than it suppose to be, set it to 0
                for name in xrange(len(user)):
                        if user[name][1] < int(policy[3]):
                                user[name][1] = 0
                        if user[name][2] < int(policy[3]):
                                user[name][2] = 0
                        if user[name][3] < int(policy[3]):
                                user[name][3] = 0

                #open log.txt file to store the logging information
                with open("log.txt", "a") as write:
                        #write the rules details into the file
                        word = "%s,%s,%s" %(rule+1,policy[0],row+1)
                        write.write(word)
                        if policy[1] == "any":
                                for line in user:
                                        #write all user information into it
                                        word = ",%s,%s,%s,%s" %(line[0],line[1],line[2],line[3])
                                        write.write(word)
                                write.write('\n')
                        else:
                                for name in xrange(len(user)):
                                        if user[name][0] == policy[1]:
                                                #write only related user information into it
                                                word = ",%s,%s,%s,%s"
%(user[name][0],user[name][1],user[name][2],user[name][3])
                                                write.write(word)
                                write.write('\n')
```

3.4. Analyzer

3.4.1. alert() - Alert Menu for choosing Analyzer-related option

```
def alert():
        menu = True #temporart turn on the menu
        while menu:
```

```
        #formatting purpose only
        print "#####################################"
        print "##\t\tAlert\t\t##"
        print "#####################################"
        #show menu option
        print "1. Show Alert Policy"
        print "2. Add Alert Policy"
        print "3. Edit Alert Policy"
        print "4. Delete Alert Policy"
        print "5. Run Analyzer"
        print "9. Back to Main Menu"
        print "0. Exit"
        #get input from user and goto respective function
        option = raw_input("Option: ")
        if option == '1':
            show() #display all alert policy information
        elif option == '2':
            add() #add new alert policy
        elif option == '3':
            edit() #edit current alert policy
        elif option == '4':
            delete() #delete alert policy
        elif option == '5':
            Analyzer() #call analyzer to analysis log.txt file
            menu = False
        elif option == '9':
            menu = False
        elif option == '0':
            sys.exit()
        else:
            print "Wrong input, please try again."
```

3.4.2. show() - Show all Alert Policy

```
def show():
    print "\nShow Alert Policy,"
    # open policy file and store the data into policy[]
    read = open("resources/alertPolicy.txt", "r")
    policy = []
    for line in read:
        policy.append(line.strip().split(':'))
    # print out the policy[] info one by one
    for row in xrange(len(policy)):
        print "Alert", row+1
        print "Port/Rule:", policy[row][0]
```

```
            print "User:", policy[row][1]
            print "Action:", policy[row][2]
            print "Threshold:", policy[row][3], "\n"
    read.close() #close file
```

3.4.3. add() - Add new Alert Policy

```
def add():
    # request user to input required data
    print "Add new Alert Policy,"
    print "Enter rule number to check by rule, e.g. 1\nEnter port number to check by port, e.g.
4013"
    port = raw_input("Port/Rule: ")
    print "Enter specific username or 'any' for all user"
    user = raw_input("User: ")
    print "r:read\nw:write\nx:execute"
    action = raw_input("Action: ")
    print "At least how many time the action occur for specific user"
    threshold = raw_input("Threshold: ")

    #open policy file and write appended data into it
    with open("resources/alertPolicy.txt", "a") as alert:
        alert.write( '\n' + port + ':' + user + ':' + action + ':' + threshold)
    sleep(0.2) #pause system
    print "The policy is successfully added." #print success
```

3.4.4. edit() - Edit existing Alert Policy

```
def edit():
    #open policy file and save the data into policy[]
    print "Edit Alert Policy,"
    policy = []
    with open("resources/alertPolicy.txt", "r") as fil:
        for line in fil:
            policy.append(line.strip().split(':'))

    #request user to input the rules number for edit
    edit = 0 #set for validate purpose only
    while not edit in xrange(1, len(policy)+1):
        edit = raw_input("Rules: ")
        edit = int(edit)
    edit-=1 #list start with 0, not 1
    print "Enter rule number to check by rule, e.g. 1\nEnter port number to check by port, e.g.
4013"
    policy[edit][0] = raw_input("Port/Rule: ")
    print "Enter specific username or 'any' for all user"
```

```
        policy[edit][1] = raw_input("User: ")
        print "r:read\nw:write\nx:execute"
        policy[edit][2] = raw_input("Action: ")
        print "At least how many time the action occur for specific user"
        policy[edit][3] = raw_input("Threshold: ")

        #open file to rewrite all policy into it, include the edited policy
        with open("resources/alertPolicy.txt", "w") as fil:
                for row in xrange(len(policy)):
                        fil.write(policy[row][0] + ':' + policy[row][1] + ':' + policy[row][2] + ':' +
policy[row][3])
                        if row < len(policy)-1:
                                fil.write('\n')
        sleep(0.2) #system pause
        print "The policy is successfully updated." #print success
```

3.4.5. delete() - Delete unwanted Alert Policy

```
def delete():
        print "Delete Alert Policy,"
        #read all policy into policy[]
        policy = []
        with open("resources/alertPolicy.txt", "r") as fil:
                for line in fil:
                        policy.append(line.strip().split(':'))

        #request user to input the policy number wanted to be delete
        delete = 0
        while not delete in xrange(1, len(policy)+1):
                delete = raw_input("Rules: ")
                delete = int(delete)
        delete-=1 #list start with 0, not 1

        #open file and rewrite all policy into it except the policy which is decided to be delete
        with open("resources/alertPolicy.txt", "w") as fil:
                for row in xrange(len(policy)):
                        if row != delete:
                                if row != 0:
                                        fil.write('\n')
                                fil.write(policy[row][0] + ':' + policy[row][1] + ':' + policy[row][2] + ':'
+ policy[row][3])
        sleep(0.2)
        print "The policy is successfully updated."
```

3.4.6. Analyzer() - Analyze log.txt with Alert Policy

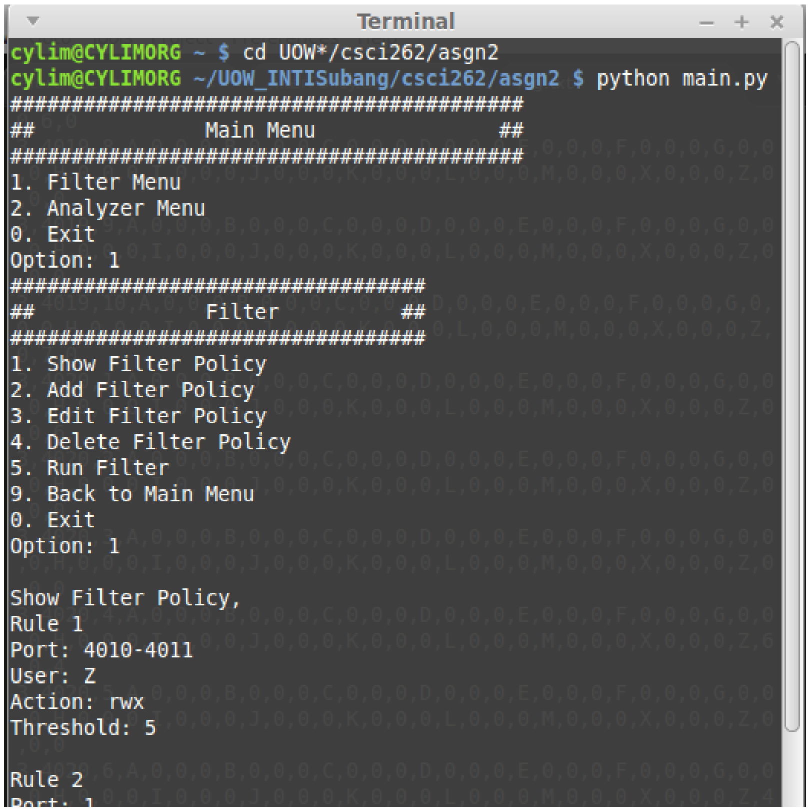
```
def Analyzer():
```

```
print "Analysis log.txt activity with alert policy,"
#read log.txt into log[]
log = []
with open("log.txt", "r") as fil:
        for line in fil:
                log.append(line.strip().split(','))
#read alert policy into policy[]
policy = []
with open("resources/alertPolicy.txt", "r") as fil:
        for line in fil:
                policy.append(line.strip().split(':'))

ID = 'ABCDEFGHIJKLMXZ' #for username
open("alert.txt", "w").close() #clear everything inside the file
fil = open("alert.txt", "a") #append mode

#For each policy, perform below operation
for row in xrange(len(policy)):
        #Save Alert Number into a string, and then write into file
        toWrite = "Alert %d,\n" %(row+1)
        fil.write(toWrite)
        #for each log[], perform below operation
        for line in xrange(len(log)):
                if policy[row][0] == log[line][1]: #if the port is same
                        #for each user in the log file, check the read, write and execute
                        for user in xrange(3, len(log[line])-3):
                                if policy[row][1] == "any": #if the username fill is any
                                        if log[line][user] in ID: #when the name matched with ID
                                #if the policy write contain checking in read, write and/or execute,
                                #check the total trigger, if the total trigger higher than the alert, write it into file
                                                if 'r' in policy[row][2]:
                                                        if policy[row][3] <= log[line][user+1]:
                                                                toWrite = "user:%s, action:
read exceed %s on port %s in day %s\n" %(log[line][user], policy[row][3], log[line][1], log[line][2])
                                                                fil.write(toWrite)
                                                if 'w' in policy[row][2]:
                                                        if policy[row][3] <= log[line][user+2]:
                                                                toWrite = "user:%s, action:
write exceed %s on port %s in day %s\n" %(log[line][user], policy[row][3], log[line][1], log[line][2])
                                                                fil.write(toWrite)
                                                if 'x' in policy[row][2]:
                                                        if policy[row][3] <= log[line][user+3]:
```

```
                                    toWrite = "user:%s, action:
execute exceed %s on port %s in day %s\n" %(log[line][user], policy[row][3], log[line][1],
log[line][2])
                                    fil.write(toWrite)

                    #if the username is same as one of the user in the log
                    elif policy[row][1] == log[line][user]:
                    #if the policy write contain checking in read, write and/or execute,
                    #check the total trigger, if the total trigger higher than the alert, write it into file
                        if 'r' in policy[row][2]:
                            if policy[row][3] <= log[line][user+1]:
                                toWrite = "user:%s, action: read
exceed %s on port %s in day %s\n" %(log[line][user], policy[row][3], log[line][1], log[line][2])
                                fil.write(toWrite)
                        if 'w' in policy[row][2]:
                            if policy[row][3] <= log[line][user+2]:
                                toWrite = "user:%s, action: write
exceed %s on port %s in day %s\n" %(log[line][user], policy[row][3], log[line][1], log[line][2])
                                fil.write(toWrite)
                        if 'x' in policy[row][2]:
                            if policy[row][3] <= log[line][user+3]:
                                toWrite = "user:%s, action: execute
exceed %s on port %s in day %s\n" %(log[line][user], policy[row][3], log[line][1], log[line][2])
                                fil.write(toWrite)

            elif policy[row][0] == log[line][0]: #if the rules is same
                for user in xrange(3, len(log[line])-3):
                    if policy[row][1] == "any": #if the username fill is any
                        if log[line][user] in ID: #when the name matched with ID
                    #if the policy write contain checking in read, write and/or execute,
                    #check the total trigger, if the total trigger higher than the alert, write it into file
                            if 'r' in policy[row][2]:
                                if policy[row][3] <= log[line][user+1]:
                                    toWrite = "user:%s, action:
read exceed %s on port %s in day %s\n" %(log[line][user], policy[row][3], log[line][1], log[line][2])
                                    fil.write(toWrite)
                            if 'w' in policy[row][2]:
                                if policy[row][3] <= log[line][user+2]:
                                    toWrite = "user:%s, action:
write exceed %s on port %s in day %s\n" %(log[line][user], policy[row][3], log[line][1], log[line][2])
                                    fil.write(toWrite)
                            if 'x' in policy[row][2]:
                                if policy[row][3] <= log[line][user+3]:
                                    toWrite = "user:%s, action:
execute exceed %s on port %s in day %s\n" %(log[line][user], policy[row][3], log[line][1],
log[line][2])
                                    fil.write(toWrite)
                    #if the username is same as one of the user in the log
                    elif policy[row][1] == log[line][user]:
                    #if the policy write contain checking in read, write and/or execute,
                    #check the total trigger, if the total trigger higher than the alert, write it into file
                        if 'r' in policy[row][2]:
                            if policy[row][3] <= log[line][user+1]:
                                toWrite = "user:%s, action: read
exceed %s on port %s in day %s\n" %(log[line][user], policy[row][3], log[line][1], log[line][2])
                                fil.write(toWrite)
                        if 'w' in policy[row][2]:
                            if policy[row][3] <= log[line][user+2]:
                                toWrite = "user:%s, action: write
exceed %s on port %s in day %s\n" %(log[line][user], policy[row][3], log[line][1], log[line][2])
                                fil.write(toWrite)
                        if 'x' in policy[row][2]:
                            if policy[row][3] <= log[line][user+3]:
                    toWrite = "user:%s, action: execute exceed %s on port %s
in day %s\n" %(log[line][user], policy[row][3], log[line][1], log[line][2])
                                fil.write(toWrite)
    fil.close()
```

4. *SAMPLES OF EXECUTION*

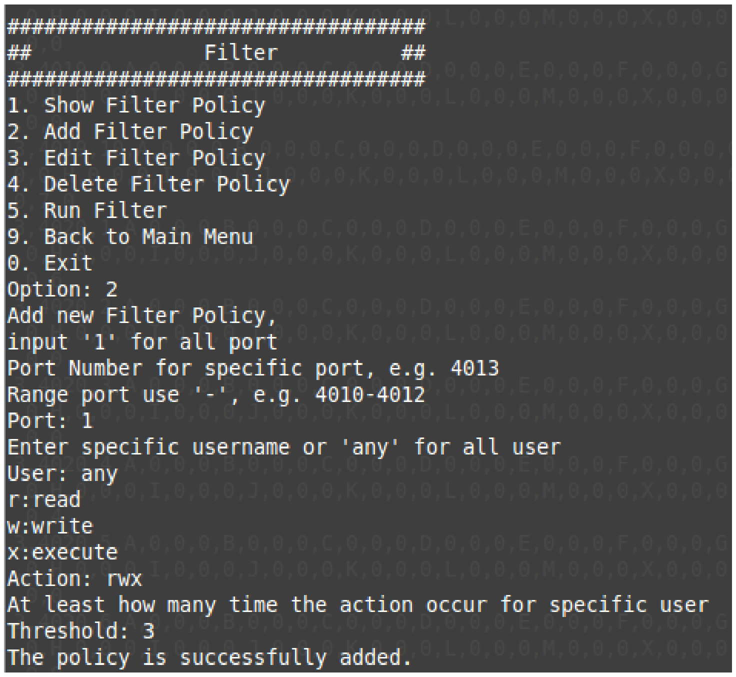
4.1. Show Policy



Terminal

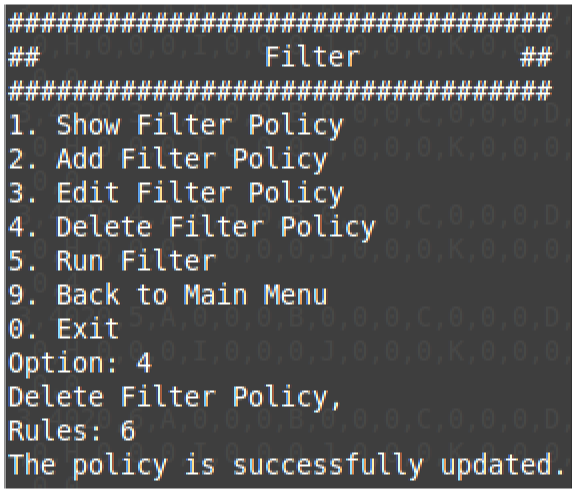
```
cylim@CYLIMORG ~ $ cd UOW*/csci262/asgn2
cylim@CYLIMORG ~/UOW_INTISubang/csci262/asgn2 $ python main.py
##########################################
##              Main Menu               ##
##########################################
1. Filter Menu
2. Analyzer Menu
0. Exit
Option: 1
##################################
##              Filter          ##
##################################
1. Show Filter Policy
2. Add Filter Policy
3. Edit Filter Policy
4. Delete Filter Policy
5. Run Filter
9. Back to Main Menu
0. Exit
Option: 1

Show Filter Policy,
Rule 1
Port: 4010-4011
User: Z
Action: rwx
Threshold: 5

Rule 2
Port: 1
```

4.2. Add Policy



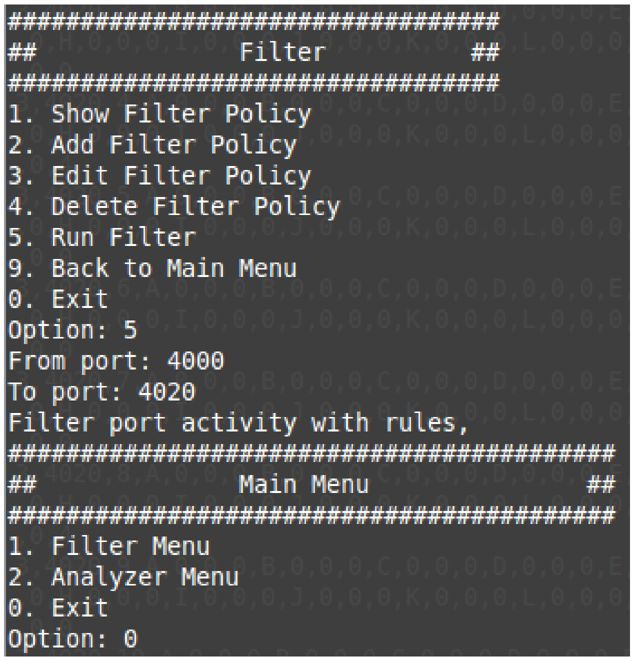
```
##################################
##              Filter          ##
##################################
1. Show Filter Policy
2. Add Filter Policy
3. Edit Filter Policy
4. Delete Filter Policy
5. Run Filter
9. Back to Main Menu
0. Exit
Option: 2
Add new Filter Policy,
input '1' for all port
Port Number for specific port, e.g. 4013
Range port use '-', e.g. 4010-4012
Port: 1
Enter specific username or 'any' for all user
User: any
r:read
w:write
x:execute
Action: rwx
At least how many time the action occur for specific user
Threshold: 3
The policy is successfully added.
```

4.3. Delete Policy



```
####################################
##              Filter            ##
####################################
1. Show Filter Policy
2. Add Filter Policy
3. Edit Filter Policy
4. Delete Filter Policy
5. Run Filter
9. Back to Main Menu
0. Exit
Option: 4
Delete Filter Policy,
Rules: 6
The policy is successfully updated.
```

4.4. Filter



```
##################################
##              Filter          ##
##################################
1. Show Filter Policy
2. Add Filter Policy
3. Edit Filter Policy
4. Delete Filter Policy
5. Run Filter
9. Back to Main Menu
0. Exit
Option: 5
From port: 4000
To port: 4020
Filter port activity with rules,
##########################################
##              Main Menu               ##
##########################################
1. Filter Menu
2. Analyzer Menu
0. Exit
Option: 0
```

5. *CONCLUSION*

The author appreciated that the lecturer used this method of teaching, and not to specific the programming language to be used for coding the console program, as the purpose of the assessment is to enhance students understanding in certain topics and how Intrusion Detection System works. Beside that, the author also have better understanding of how to built a simple Intrusion Detection System by his own.

The author also expressed that this is a better way of learning system security, besides studying all those theory, the author have the chances to actual try to built a console program with the theory which lead the author to have a better understanding of related theories with actual skills.