Final Document:

Risk Assessment Tool That Calculates The Type, Cost And Likelihood of Breaches.

Institute of Technology Carlow

Department of Computing & Networking

Student Name: Eimhin Lane

Student Number: C00240680

Student Name: Eimhin Lane
Student Number: C00240680

Table of Contents

Introduction:

This document's purpose is that of underlining the process, decision and compromises made during development of the quantitative risk assessment tool. Outlining how the research phase would alter and impact development of the proper tool, which was an ultimately expected result. Additional descriptions on development problems, process and learning experiences are to be described in this document.

Final Product Description:

The risk assessment tool developed is in the style of a point-based questionnaire in order to provide a concise and easy to use tool for those that are not knowledgeable with security matters. The tool was designed around this idea as to allow everyone an opportunity to not only learn of the risks they face but how to tackle them.

Development was done in two stages; the first stage was a proof of concept for the tool in Excel developed using macros. Alongside this is the formula for the Monte Carlo function with its standard 5000 iterations, this is used to determine the monetary losses incurred based on various aspects of a company's financial standing.

The second stage was taking to Python to develop this point-based Risk Assessment tool. This process was a learning experience from start to finish due to my unfamiliarity with Python.

Finding the various intricacies of Python and learning how to adapt them to the style of interface I wanted took much trial and error, from small quirks to failed builds. The overall development of the project experienced multiple upheavals due to the nature of learning Python.

Tools Used:

The primary development language used for development was Python. I found various tools in Python that would be useful for development such as Tkinter which helps in developing user interfaces and Python Imaging Library (PIL) which is used to open, manipulate and save images. I would find PIL useful going forward for the various images used in the programs.

Early Development Hurdles:

When taking on Python as the primary programming language I soon found I would need to create a foundation to work towards. Due to Python being a language that I had otherwise no experience in I took to creating a mock up interface and quiz style layout in excel utilising macros. Equally to familiarise myself with Python as a programming language I would work on various smaller programs to improve myself with the various tools I now had at my disposal.

I attempted to begin the project in Python initially, utilising tkinter and PIL I would develop a sidebar application that was based on concepts images made earlier during research. While the sidebar application functioned, it would present some issues. These issues are that the sidebar would extend further than necessary when the mouse scrolled over it and wouldn't fully retract after the mouse scrolls away. The application would also prove to be problematic with placing the intended questions in the centre of the screen.

With this issue leeching more time I would instead move over to Excel and begin making developing the proof of concept.

Developing the Proof of Concept:

Development Direction Decision-

To start I knew I would need to utilise either a regressive machine learning model or the monte carlo function when determining the loss of money in the event of an attack. I found that FAIR's tool, FAIR-U was developed to utilise the Monte Carlo function as its basis for risk assessment. Additionally, NIST describes their use of the monte carlo function in conjunction to using FAIR as a basis. These two institutions are at the forefront of most discussion involving Risk Assessment tools and as such I elected to follow their lead, adopting the monte carlo function to determine monetary losses.

FAIR-U's interface was used to determine where I'd like to take the interface of my project due to its complex layout for the uninformed. Seeing that the tool only asked about monetary income and losses I chose only to take on its Monte Carlo function and instead deviate to the

point system for all other risk assessment questions I had planned. This way I would be able to create a cleaner and easier to follow interface for the uninformed.

Excel Development-

The proof of concept was created utilising Excel macros to design a question interface where you can add or remove questions from the list and even add images if necessary. Following the question worksheet is the test worksheet where users can enter their answers to multiple choice questions. Finally, a worksheet to store inputted answers is needed.

Starting with setting up the question interface I made it so that the range for answers would extend down the rows "D" and "E". All questions from the "D" and "E" we take up 5 rows for answers and for later setting up an array for determining the users selected answer.

```
Option Explicit

Sub AddQuestion()
Dim FirstAvailRow As Long
Dim QuestionTemplateRange As Range
With Sheet1
    Set QuestionTemplateRange = .Range("AA1:AH5")
    FirstAvailRow = .Range("D999").End(xlUp).Row + 5
    QuestionTemplateRange.Copy
    .Range("D" & FirstAvailRow).PasteSpecial xlPasteAll
    .Range("D" & FirstAvailRow).Value = "=D" & FirstAvailRow - 5 & "+1"
    .Range("E" & FirstAvailRow).Select
End With
End Sub
```

For deleting a question, you open a confirmation box that asks the user if they are certain, as opposed to the initial system which immediately deleted questions on click which risked an individual deleting questions unintentionally. If the user selects no, we exit the submenu and return to the standard interface. In the event the user selects yes it begins clearing the 5 rows used by the question and renames the now moved up questions from below to the new value.

```
Sub DeleteQuestion()
Dim SelQuestRow As Long, SelQuestNumb As Long, LastQuestNumb As Long, LastRow As Long, QuestNumb As Long,
Dim PicShape As Shape
With Sheet1
    If MsgBox("Are you sure you want to delete this question?", vbYesNo, "Delete Question") = vbNo Then Ex
    If .Range("B5").Value = Empty Then Exit Sub 'exit sub if no selected row
    SelQuestRow = .Range("B5").Value 'Question Row
    SelQuestNumb = .Range("D" & SelQuestRow).Value 'Question #
    On Error Resume Next
    .Shapes("QuestPic" & SelQuestNumb).Delete 'Delete any existing question Picture
    On Error GoTo 0
    .Shapes("DelQuestBtn").Visible = msoFalse ' Hide Delete button
    .Shapes("AddPicBtn").Visible = msoFalse 'Hide Add Pic button
```

The message box is created and delete functions take place.

Following this we now create a separate macro in visual basic for creating questions pictures and placing the previous macros add or remove buttons to the questions. We open the user storage to find any of the standard image file types from ".jpg" to ".bmp".

We then assign the picture path to the space in the question list for use when running the assessment tool. Additionally, if any picture existed for the question before hand I needed to ensure it was deleted otherwise it would cause errors such as placing two images or only keeping the first one which was not intended. I chose to delete the previous images path the moment a new image was selected. The new images path was assigned a name based on the question number and placed in the appropriate question's row.

Following this we assign the interactable add and remove buttons to their respective rows on the first sheet. They were incremented towards the top left corner of their questions columns, "h" and "j" respectively, and made visible above any text entered by the user if hovered over.

Finally, was the most important macro, the one that ran the tool. I created the start-up function first, which presents the user with an interface to enter a name and ID and makes it so that when the start button is pressed it becomes hidden.

```
Sub StartQuiz()
With Sheet2
    .Shapes("StartQuizBtn").Visible = msoFalse 'Hide Start button
    .Range("B2").Value = 1 'Set first question
    Sheet3.Range("B2").Value = .Range("G2").Value 'Name
    Sheet3.Range("C2").Value = .Range("G4").Value 'ID
    .Range("G2,G4").ClearContents 'Clear Name & Ide
    LoadQuestion
    .Range("B1").Value = Now 'Set Start Time & Date
End With
End Sub
```

Taking in the inputs of name and ID and moving to the first question.

Loading the questions came next which took the longest to develop due to it needing to keep track of multiple variables and worksheet positions. The time it took to create this function did help me familiarise myself with visual basic considerably due to the depth of content being handled. This trial and error would benefit my work speed greatly in the next functions. Loading up a question requires clearing the screen of its previous contents and bringing in the first worksheets details, these being question number, wording, choices, style of question and images where necessary. Checking the style of questions such as multiple choice, checkbox or text was necessary to presenting the proper answer system. When loading the final question, the "finish" button would be made visible rather than the previous "next " button.

```
eneral)                                                    (Declarations)
Sub LoadQuestion()
Dim QuestNumb As Long, AnswerRow As Long, LastQuestRow As Long, LastQuestNumb, ShapeNumb As Long
Dim PicShape As Shape
Dim QuestType As String
LastQuestRow = Sheet1.Range("D999").End(xlUp).Row
LastQuestNumb = Sheet1.Range("D" & LastQuestRow)
With Sheet2
    .Range("F24:G25,D5:D9").ClearContents 'Clear Previous Results
    'Clear Previous Checkboxes & Options
    For ShapeNumb = 1 To 5
        .Shapes("CheckBox" & ShapeNumb).Visible = msoFalse
        .Shapes("Option" & ShapeNumb).Visible = msoFalse
    Next ShapeNumb
    QuestNumb = .Range("B2").Value 'Question #
    'Set Question Type
    If .Range("D2").Value = "Checkboxes" Then
    QuestType = "CheckBox"
    ElseIf .Range("D2").Value = "Multiple Choice" Then
    QuestType = "Option"
    End If
    If QuestType = "" Then QuestType = "Text"
    On Error Resume Next
    .Shapes("QuestPic" & QuestNumb - 1).Delete
    .Shapes("QuestPic" & QuestNumb).Delete
    On Error GoTo 0
       On Error Resume Next
        Set PicShape = Sheet1.Shapes("QuestPic" & QuestNumb)
        On Error GoTo 0
        If Not PicShape Is Nothing Then 'Picture Exists
                 .Range(4 & ":" & 17).EntireRow.Hidden = False
                PicShape.Copy
                Sheet2.Range("F3").Select
                On Error Resume Next
                Sheet2.PasteSpecial
                Selection.Name = "QuestPic" & QuestNumb
```

Part of the question loading process and clean-up.

The next function finally utilised the answer sheet for storing user input to later be parsed into the two risks being determined in the proof of concept. Creating multiple variables to track users answer, correct answer, question numbers and more. Comparing the users input to the correct choice in either multiple choice questions or checkbox questions was done by checking if the correct answer was the same as the user's selection. When answers are compared, we run the load question function to move onto the next question.

```
    If QuestType = "Multiple Choice" Then
        On Error Resume Next
        CorrectAnswer = .Range("B11").Value
        UserAnswer = .Range("B12").Value
         If CorrectAnswer = UserAnswer Then Sheet3.Range("E" & AnswerRow).Value = True Else: Sheet3.Range
    End If

    If QuestType = "Checkboxes" Then
        For AnswerCount = 5 To 9
            If .Range("C" & AnswerCount).Value = "ü" Then CorrectAnswer = CorrectAnswer & .Range("B" & Ans
            If .Range("D" & AnswerCount).Value = True Then UserAnswer = UserAnswer & .Range("B" & AnswerCo
        Next AnswerCount
         If CorrectAnswer = UserAnswer Then Sheet3.Range("E" & AnswerRow).Value = True Else: Sheet3.Range
    End If

     If QuestType = "Text" Then
        CorrectAnswer = .Range("C5").Value 'Correct Answer
        UserAnswer = .Range("F24").Value 'User Answer
        If InStr(CorrectAnswer, UserAnswer) > 0 Then Sheet3.Range("E" & AnswerRow).Value = True Else: Shee
     End If

    Sheet3.Range("C" & AnswerRow).Value = CorrectAnswer 'Update Results Sheet with Correct Answer
    Sheet3.Range("D" & AnswerRow).Value = UserAnswer 'Update Results Sheet with UserAnswer

    .Range("B2").Value = .Range("B2").Value + 1 'Increment Question #
    QuestNumb = .Range("B2").Value 'question #
    'Display Finish or Next buttons
    If LastQuestNumb = QuestNumb Then 'on Final question
        .Shapes("NextBtn").Visible = msoFalse
        .Shapes("FinishBtn").Visible = msoCTrue
```

Storing answer types and determining if the next question is the final one.

Finishing the quiz and reset the quiz are two functions that work closely together as once the user is done with their results, they will reset to the entering the username. To show results we wipe the previous layout and gather all information from the answers sheet for use in a formula to present the users results. When you hit reset, you then wipe the answer sheet and hide the finished quiz interface.

While not a macro the equations required for the Monte Carlo function, which as mentioned before is frequently used in Risk Assessment, can be complex. I utilised various formulas found online to determine how to best replicate the methods utilised by FAIR and NIST. The Monte Carlo function is what makes this risk assessment tool truly Quantitative due to it providing multiple results in a certain percentile of error based on the inputted information. FAIR-U utilises the upper percentiles to project losses in the event of a risk in 90% as "most likely" and "99%" as worst-case scenario. This is somewhat different from NIST who use multiple percentiles, from 10%, 50% or 90% and 50, 90% or 99%[1] as their projected losses. Using both in this spreadsheet allows the user to see both the FAIR and NIST approach.

The core values used are the yearly value, risk free rate, expected volatility and number of days worked in the year, which is the perspective of all questions in this tool. Monte Carlo requires a seed value to work which is determined through a formula that takes the risk-free rate, the expect volatility which is usually at a standard 10% and the number of workdays all multiplied and divided amongst each other and finally given a randomisation for the uncertainty aspect of risk assessment.

Formula for determining Seed Value:
"=EXP(($C$5-0.5*$C$6^2)*$C$7/261+$C$6*SQRT($C$7/261)*NORM.S.INV(RAND()))"

To combat the uncertainty inherent in all risk assessment tools it is standard to run multiple iterations. The average number of iterations for monte carlo are often upwards of 5,000. As such the formula is run 5,000 additional times utilising the same value as the seed.

I used these iterations to find the absolute loss within the percentiles for FAIR and NIST. Determining the percentage of loss is important to finding the value of absolute loss which relies on the percent and yearly value entered. Even with 5,000 iterations these calculations are not exact and can vary as seen by running the calculations again in Excel, but they offer a guide to the amount you should weigh the cost of protection to possible losses during an attack.

Percent Loss Formula:
=1 -PERCENTILE.INC(I5:I5004,1-C10)
I5:I5004=Iterations|C10=Percentage

Python Development:

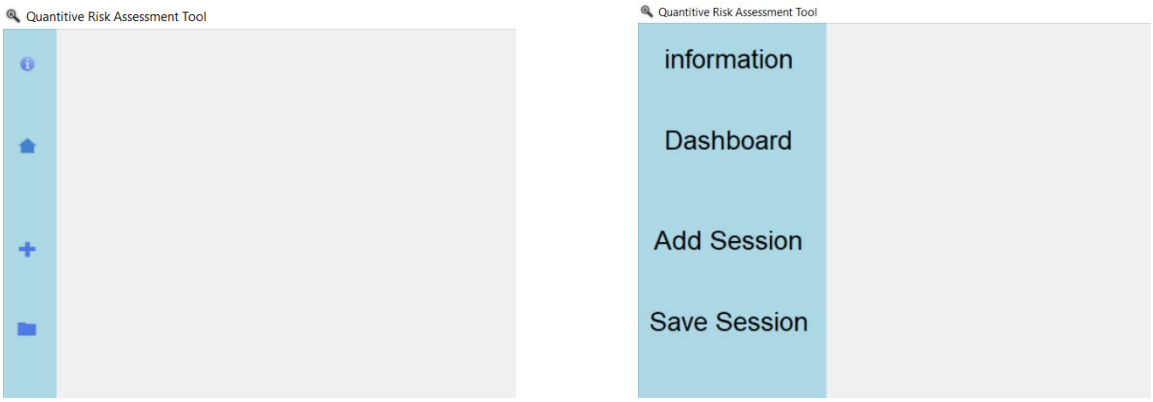
When starting development with Python I adapted to the modular nature quickly due to my previous experience with other modular programming languages such as Ruby on rails. With Ruby on Rails, you would be able to use gems to help your projects develop. I found this process like utilising the libraries of Tkinter, PIL and PyInstaller. Though ultimately, I would only fully utilise Tkinter the additional experience gained by attempted to adapt with these additional libraries of PIL and PyInstaller would help me adjust to the comparative oddities and uniqueness that Python provided.

The first attempts in Python development occurred before the Excel proof of concept. The attempt at developing with Tkinter was simple enough and would revolve around reading, writing, and saving text documents. My original concept designs showed many similarities to excel spreadsheets and with this being developed before the proof of concept I decided to try familiarise myself with these basic interface functions. The tests were done utilising .txt files in place of a .docx or a .xlsm. This was a simplistic process of opening a file within the gui directory that were of the .txt file type with read privileges. You would then project the contents of this file into an interactable dialog box for the user to type the contents they wish. When saving the file, it overwrites initial file with write privileges to change its contents.

```python
from tkinter import *
from tkinter import filedialog

root = Tk()
root.title('Risk Assessment')
root.iconbitmap('C:/gui/RAexamplelogo.png')
root.geometry("500x450")

def open_docx():
    #Read only is 'r', Read and Write 'r+', Write Only 'w', Write and Read 'w+', Append Only 'a', Append and Read 'a+'
    text_file = filedialog.askopenfilename(initialdir= "C:/gui/txt", title= "Open txt File", filetypes=(("Docx Files", "*.txt"), ))
    text_file = open(text_file, 'r')
    #read the file once its open
    text= text_file.read()

    my_text.insert(END, text)
    text_file.close()

def save_docx():
    text_file = filedialog.askopenfilename(initialdir= "C:/gui/docx", title= "Open docx File", filetypes=(("Docx Files", "*.txt"), ))
    text_file = open(text_file, 'w')
    text_file.write(my_text.get(1.0, END))

my_text = Text(root, width=40, height=10, font=("Helvetica", "16"))
my_text.pack(pady=20)

open_text = Button(root, text="Open your docx file", command=open_docx)
open_text.pack(pady=20)

save_button = Button(root, text="Save File", command= save_docx)
save_button.pack(pady=20)
root.mainloop()
```

Following the read/write program I would attempt to create a user interface with a collapsible sidebar in Python. This being the initial introduction to Tkinter when developing a complex interface, it would take some adjusting but would prove to be an overall smooth process. The versatility of Tkinter made learning certain aspects of python significantly easier, as such creating a window for the application to run in was as simple as giving it a title, dimensions and even a logo. Initially a boot up screen would open while the main program loaded but due to the speed at which python applications open this only proved to add time onto the booting up process.

The sidebar utilised labels and buttons to present options along the side of the screen. Creating a expand function to manage the sidebar extending outwards from the left-hand side of the screen involved increasing the width of the sidebar and repeating the expansion every 5 milliseconds unt fully expanded. Then we check if the frame is fully expanded, if it is we stop repeating the increments.



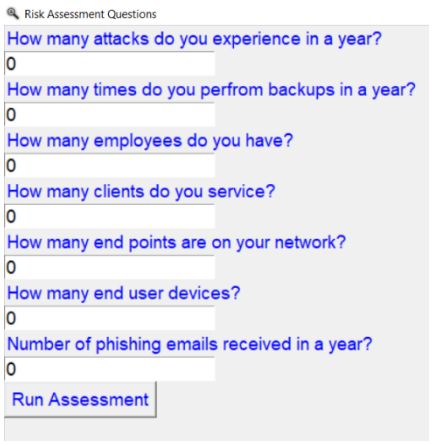
The sidebar retracted and extended.

The contract function is functionally the opposite simply retracting where expand grew. Drawing the sidebar back towards the left side of the screen. The fill function is run when either of the previous functions finished, it replaces the images used with interactable buttons and when no longer expanded returns them to normal. Binding these buttons and images to the frame of the sidebar was done to ensure dynamic reactions when retracting and expanding.

```
def expand():
    global cur_width, expanded
    cur_width += 10 #increase width by 10
    rep = root.after(5,expand) #repeat ever 5 ms
    frame.config(width=cur_width)#changes width to the new size
    if cur_width >= max_width:
        expanded = True #frame is expanded
        root.after_cancel(rep) #stop repeating the function
        fill()

def contract():
    global cur_width, expanded
    cur_width -= 10 #reduce width by 10
    rep = root.after(5,contract) #repeat ever 5 ms
    frame.config(width=cur_width)#changes width to the new size
    if cur_width <= max_width:
        expanded = False #frame is contracted
        root.after_cancel(rep) #stop repeating the function
        fill()

def fill():
    if expanded: #If the frame is expanded show a text and remove the image
        info_b.config(text='information',image = '',font=(0,21))
        home_b.config(text='Dashboard',image = '',font=(0,21))
        add_b.config(text='Add Session',image = '',font=(0,21))
        save_b.config(text='Save Session',image = '',font=(0,21))

    else: #bring the images back
        info_b.config(image=info, font=(0,21))
        home_b.config(image=home, font=(0,21))
        add_b.config(image=AddSession, font=(0,21))
        save_b.config(image=SaveSession, font=(0,21))
```

The final failed build was the questions program, this is where I would find that utilising a point-based system would best benefit the development of this system. This system was not fleshed out by the time I finished attempts on this build. The program is simple in nature, it outputs the questions with an interactable text box that saves users answers. These answers however could only be saved as integer values which at the time I didn't want to utilise. The intent was to utilise strings in their place when the issues were ironed out. The assessment was run by taking points every time a risky answer was inputted. If the user was above a certain risk score it would output if you were at low, medium, or high risk.

```
question6 =Label(window,text='Number of phishing emails received in a year?', fg='blue', font=('Arial',14))
question6.grid(row=12,column=0,padx=0,pady=0, sticky=W)


#Answers:

ans1=IntVar()
ans2=IntVar()
ans3=IntVar()
ans4=IntVar()
ans5=IntVar()
ans6=IntVar()
ans7=IntVar()

answers1=Entry(window, textvariable=ans1, fg='black', font=('Arial',14))
answers1.grid(row=1, column=0, sticky=W)
```



While this program would ultimately be a dead end it helped cement the idea of a point-based system which was more fully explored in the proof of concept that followed. The excel sheets would determine if you selected one option it would be considered a risk for one attack and another for a separate attack, ultimately deciding which of the two attacks you have more points in and are at more of a risk for.

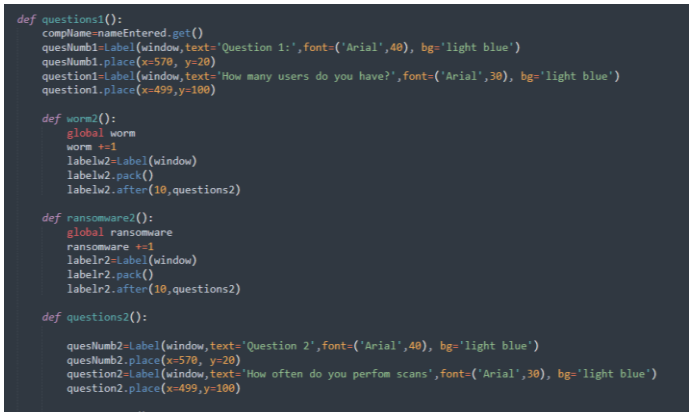
Point system-

As mentioned, the point system was now coming together as the foundation for the assessment. This system would implement multiple different count values to increment depending on which multiple choice answer was interacted with. At the end we then

determine which of the counts are highest for what you are at most risk for. The highest risk is outputted and a brief description on how to counter these attacks is presented to the user.

Final Product:

The final program begins like all the others, I import what's needed from Tkinter, set the title, set the icon, and set the dimensions of the window. I then set the variables I will use to count the points added to each risk. To start with I ask for the company name and a continue button assigned a command to move onto the first question when clicked. From here nesting functions becomes vital to make the system work. Each question sets up its wording and position before you establish the count increments for each risk which can then be assigned to the answers you want. The counts will also link to the next question so when you click a button it increments the count and moves on.



```
def questions1():
    compName=nameEntered.get()
    quesNumb1=Label(window,text='Question 1:',font=('Arial',40), bg='light blue')
    quesNumb1.place(x=570, y=20)
    question1=Label(window,text='How many users do you have?',font=('Arial',30), bg='light blue')
    question1.place(x=499,y=100)

    def worm2():
        global worm
        worm +=1
        labelw2=Label(window)
        labelw2.pack()
        labelw2.after(10,questions2)

    def ransomware2():
        global ransomware
        ransomware +=1
        labelr2=Label(window)
        labelr2.pack()
        labelr2.after(10,questions2)

    def questions2():

        quesNumb2=Label(window,text='Question 2',font=('Arial',40), bg='light blue')
        quesNumb2.place(x=570, y=20)
        question2=Label(window,text='How often do you perfom scans',font=('Arial',30), bg='light blue')
        question2.place(x=499,y=100)
```

The Question is formed, and the attack counts incremented when answered.

```
        choice21=Button(window,height=1,width=10,text='Daily',font=('Arial',30),command=questions3)
        choice21.place(x=400,y=200)
        choice22=Button(window,height=1,width=10,text='Weekly',font=('Arial',30),command=trojan3)
        choice22.place(x=800,y=200)
        choice23=Button(window,height=1,width=10,text='Monthly',font=('Arial',30),command=ransomware3)
        choice23.place(x=400,y=300)
        choice24=Button(window,height=1,width=10,text='Never',font=('Arial',30),command=ransomware3)
        choice24.place(x=800,y=300)

        quesNumb1.destroy()
        question1.destroy()
        choice1.destroy()
        choice2.destroy()
        choice3.destroy()
        choice4.destroy()

    choice1=Button(window,height=1,width=10,text='1000+',font=('Arial',30),command=ransomware2)
    choice1.place(x=400,y=200)
    choice2=Button(window,height=1,width=10,text='500-1000',font=('Arial',30),command=worm2)
    choice2.place(x=800,y=200)
    choice3=Button(window,height=1,width=10,text='250-500',font=('Arial',30),command=worm2)
    choice3.place(x=400,y=300)
    choice4=Button(window,height=1,width=10,text='0-250',font=('Arial',30),command=worm2)
    choice4.place(x=800,y=300)
    title.destroy()
    labelComp.destroy()
    nameEntered.destroy()
    button1.destroy()

button1=Button(window,height=1,width=10,text='Submit',font=('Arial',30),command=questions1)
button1.place(x=600,y=350)

window.mainloop()
```

The answers link to the respective attack.

The system repeats from there with varying risks incrementing for each question until the final screen which checks each count against one another for the highest valued risk count. It then shows how to respond to this attack and has a button to close the window when satisfied.

The choice to hard code these questions in place of utilising phpMyAdmin was made due to the interfacing between reading button inputs into the database was not something I could

properly utilise. However, this method would allow more utilisation of dynamic questions that not longer linearly progress for all answers and instead allow multiple choices leading to multiple questions.

Dynamic questions could equally have been hard coded, as they at one point were, but the destroy command for previous question's answers would no longer perform properly, instead either incrementing the incorrect answer from the previous question or leaving the previous buttons up.

Changes made during development:

The decision to prioritize monte carlo over machine learning was a big change in development as I had decided that security is inherently straightforward, as such addressing security concerns shouldn't rely on something like machine learning when you can portray the risk in a more straightforward manner.

Additionally, the portrayal of the questions changed from a more straightforward list to a multiple-choice selection that would increment risk values with each answer. This choice was made out of practicality, with guidance on each answer a person won't need to spend a lot of time thinking about their answer and instead focus on their options, all while getting the same result. It was equally an easier route to program as it results in account for fewer inputted variables and instead focusing on options presented.

Learning Experience:

This project helped me learn a great deal on how risk assessment programs such as FAIR-U work and how to develop in not only one but two programming language, these being Python and Visual basic. The adaption to these languages furthered my overall programming knowledge and help develop my skills in interface design and overall functionality.

When starting the project, I had never used either Visual Basic or Python. While similarities between different programming languages always exist, they will still require a certain amount of learning and adaption in order to establish a knowledge on the various quirks each language has.

Additionally, while my knowledge on the various risks is primarily from my time at I.T Carlow with the course focusing on the various threats we face, how to recognise and counter them. There is still a necessary amount of research necessary due to the nature of risk assessment differencing from risk recognition. You need to be truly capable of assessing how these risks will affect the user within a year. With FAIR's website[2] and book[3] helping me establish my knowledge on how this assessment style I can now better understand the inner workings of risk assessment. NIST equally provided methods to learn through their various resources available on GitHub[4].

Special Thanks:

To Paul Barry, my project supervisor through the gear for the help and support he provides me during development of this project, his guidance was invaluable. Additionally, I'd like to thank James Egan for the project concept and all Cyber Year 4 lecturers for their advice on my project during presentations.

Externally from my research I'd like to thank Excel for Freelancers for their guides to visual basic development, FAIR Institute, NIST, FAIR-U

Conclusion:

This project was a learning experience from start to finish, it tested my abilities and allowed them to expand beyond what they previously were in many ways. The research and development were an enlightening experience with many hurdles but ultimately the finished product provides the service I had wished it to. Overall, this project was a difficult experience albeit an enjoyable one.

References:

1. NIST. 2018. *Risk Assessment Tools*. [online] Available at: https://www.nist.gov/itl/applied-cybersecurity/privacy-engineering/collaboration-space/focus-areas/risk-assessment/tools [Accessed 26 November 2021].

2. Institute, F., 2022. *Establishing Framework Standards for Risk Management*. [online] Fairinstitute.org. Available at: https://www.fairinstitute.org/mission

3. Freund, J. and Jones, J., 2015. *Measuring and managing information risk*. Oxford, UK: Butterworth-Heinemann.

4. Boeckl, K., *GitHub - usnistgov/PrivacyEngCollabSpace: Privacy Engineering Collaboration Space*. [online] GitHub. Available at: https://github.com/usnistgov/PrivacyEngCollabSpace [Accessed 26 November 2021].