**Unit 2 – Every Bit of the Internet**

**PROJECT 2.1.6 – A pHishy Fish Tank**

**\*\*Instructions:** Please change the text color of your responses to red text.  Please organize the endings to each page.

**Introduction**

The internet is full of gadgets and it seems *everything*can be connected to the internet, from a device that lets you remotely talk to your dog when it barks to a salt shaker that can monitor your salt usage. But the vast amount of connected devices can present vulnerabilities that threaten the security of the entire internet.

Even the most seemingly benign devices, when connected to the internet, can create a vulnerability for a larger system. One unusual device attack was on a fish tank at a North American casino. The fish tank was connected to the internet and was not as secure as it should have been. Malicious users were able to gain access to it. Once the users had access to the fish tank, they were able to scan other devices on the same network. The malicious users were able to send all sorts of data from this network to a remote server outside of the country before they were caught and denied further access.

In this project, you will apply all of the skills you have learned throughout this lesson to perform data analysis and secure a pHishy fish tank.

In its lobby, a small business in town has a state-of-the-art, internet-enabled fish tank. They have reached out to you as a software engineer and told you their problem.

* *A few days ago, we updated our fish tank monitoring system via email. Yesterday, one of our fish died unexpectedly. The monitoring system reports everything is okay, but we suspect it might not be working properly. Can you please check that the update was valid and that the software is running properly? We are worried about our fish!*

You suspect the fish tank monitoring system has been compromised. You will collect evidence, analyze the system, and fix any problems you encounter.

**Requirements**

* Identify and explain any and all security vulnerabilities that may have resulted in a system breach.
* Document any and all problems you find with the fish tank monitoring system.
* Describe how you plan to fix the problems.
* Record how you fixed and tested the system.
* Make recommendations to further improve the software so that these types of problems do not recur.

**Evidence #1**

Since the customer said that updates are done via email, you decide to research emails first.

Analyze the entries in the file. Identify any and all activities you think could be related to a lack of security and/or a system breach.

| The account camila@great\_company.com does not have two-factor authentication on and their password is weak, which could have led to the password being brute-forced. |
| --- |

**Evidence #2**

After seeing the email evidence, you suspect there has been a system breach and now need to explore more evidence—the *Downloads*folder. It contains files that one of the employees of the business downloaded from the internet.

Analyze the files, review the contents of the extracted files and identify any irregularities that may exist.  Test the software by attempting to run it. Can you? Why or why not?

| The software cannot run, as all files are imported into the fishtank.py, but an instance of the FishTank class is never made and thus no code is ever run, as all code is inside of the two methods part of the FishTank class. |
| --- |

**Evidence #3**

You suspect the software may be incomplete. You decide to continue exploring the evidence in the *Downloads* folder to see if you can discover the missing information.

You spot a file whose content looks suspicious, not in plain text. Convert the contents of the file one line at a time into something readable.  NOTE: If you try to convert more than one line at a time, the process will fail.  This process will take a few minutes, so begin your documentation.

67ee2dbe7b114fed94c8d6ba4a420b3a.pngInclude a screenshot of all evidence related to the breach.

|  |
| --- |

Once you can read the contents of the file, copy the readable contents to a new *Python®* file in your *designated*folder.  Share a copy of the py file here.

| # start\_monitoring.py  import tkinter as tk  import fishtank as tank  my\_tank = tank.FishTank()  my\_tank.mainloop() |
| --- |

Document any and all problems you find with the fish tank monitoring system.

 Include screenshots of the running program, capturing output on the screen and elsewhere.

Record the filenames and specific lines of code that are causing the problem(s).

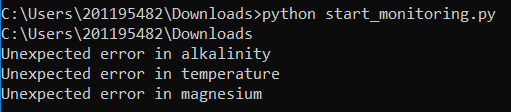
|  |
| --- |

Describe how you plan to fix the problems

| Since every file has a try except that prints “Unexpected error,” I will replace this with the file that is erroring, which will allow me to find the problems in the files.  After finding the location of the errors, I will remove the try except block to allow the program to print out the error report, which will make it easier to fix the problem. |
| --- |

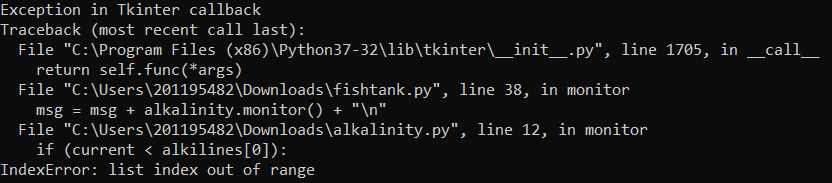
Describe how each modification will or will not affect other parts of the monitoring software.

| These modifications will cause the entire monitoring software to crash at first, as an error in one file that is not caught in a try except block will stop the entire program. |
| --- |

Record how you fixed and tested the system: 

First, I changed the print(“Unexpected error”) in all of the files to say the file that the error occurred in, and then found that the errors were in alkalinity.py, temperature.py, and magnesium.py.

I first commented out the try catch block in alkalinity.py to try to get the program to print out an error report:

This error is caused by the alkilines list being empty, as the list(range(val1, val2+1)) that was used to create it did not create a list, as, since the step is a positive number and the starting value, val1 = 17, is greater than the end value, val2+1 = 13, this means that the range() method cannot create a list and thus returns an empty list. I fixed this by changing the line of code

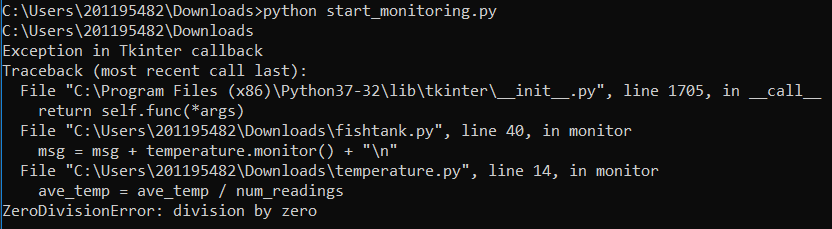
alkilines = list(range(val1, val2+1))

to

alkilines = list(range(val2+1, val1))

which fixed the issue in this file.

I then commented out the try catch block in temperature.py to try to get another error report:



This error is caused by num\_readings being zero, rather than the number of readings, which can be fixed by replacing the line

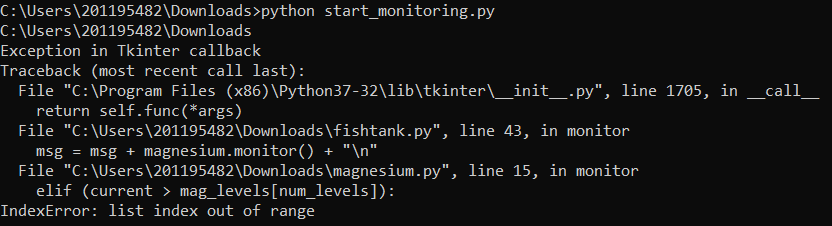
num\_readings = 0

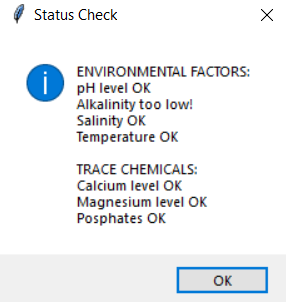
with

num\_readings = len(temp\_readings)

After verifying that the new code worked, I uncommented the try except block and moved onto the last issue.

For the last issue, I commented out the try except block in magnesium.py, and got the error:



This issue is caused by num\_levels being larger than the largest index of the list mag\_levels, as num\_levels is equal to the length of mag\_levels, which is actually 1 larger than the largest index in the list. I fixed this by changing the line of code   
elif (current > mag\_levels[num\_levels]):  
with  
elif (current > mag\_levels[num\_levels - 1]):

After testing the program to verify that all of the known errors were fixed, I added back the try catch block.

Running the fixed program reveals that the alkalinity was too low, which is why the fish died.

Create screenshots of the code you modified, including the test cases.

| Code snippets are shown above |
| --- |

Make recommendations to improve the software so that these types of errors do not recur.

| Testing the software with multiple test cases can help to stop errors from occurring, as they would have revealed these issues. |
| --- |

Before you submit your work, please make sure that you do the following:

Collect and organize all of your documentation:

1. Screenshots and explanations of the evidence that related to the system breach
2. Screenshots of the running program, including all output
3. Screenshots and explanations of the modified code, including the test cases
4. Recommendations to improve the code

Conclusion

How did a lack of security awareness contribute to the problems with the fish tank monitoring software?

| Because Camila’s password was insecure and she didn’t use two factor authentication, an attacker likely was able to log in to her account through brute forcing the password, which caused Camila to receive an incorrect or corrupted version of the fish tank monitoring software, with some parts of the program still encrypted and many parts of it nonfunctional. |
| --- |

How did your debugging and techniques help you fix the problems with the software?

| Reading comments left by the programmer of the program let me understand what the program should have done more easily than if I was just given the code. Additionally, editing print statements allowed me to find where errors occurred in the program. Hand tracing then allowed me to find which line in the program was at fault for the errors, and then testing against test cases allowed me to make sure the program was functional. |
| --- |