Template based on the Centers for Medicare & Medicaid Services, Information Security & Privacy Management's Assessment

Security Assessment Report

Version N.0

April 26, 2023

Table of Contents

1. Summary

The goal of this documentation is to assess the security of Samantha Walsh's rendition of "The Oregon Trail" game written in Python code. The greatest security risk is the lack of organization.

1. Assessment Scope

Tools used for testing included PyCharm, GitHub, and IDLE. Major limitations include time management and large size of the program.

2. Summary of Findings

Of the findings discovered during our assessment, 3 were considered High risks, 4 Moderate risks, 3 Low, and 0 Informational risks. The SWOT used for planning the assessment are broken down as shown in Figure 1.

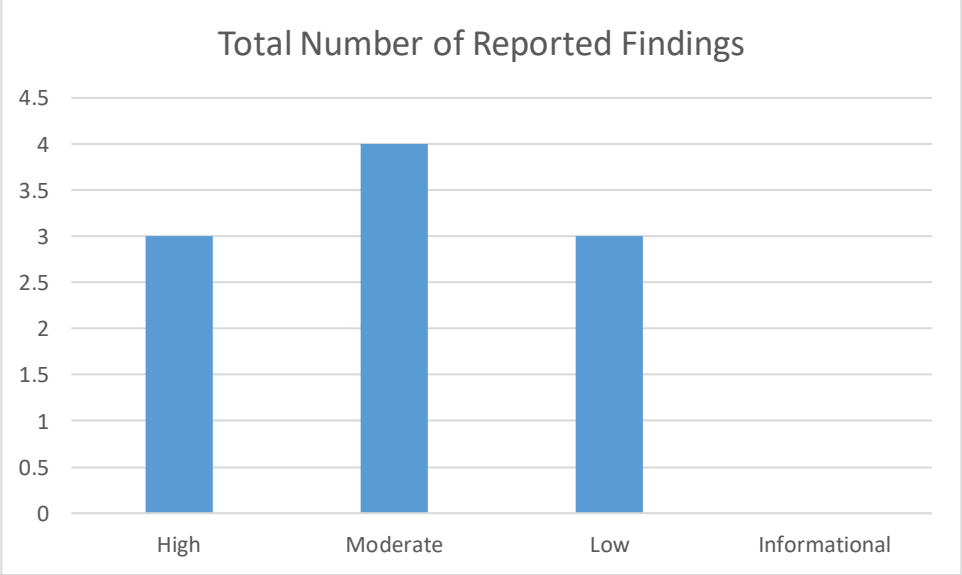


Figure 1. Findings by Risk Level

High risks included issues related to poor coding organization and documentation. Moderate risks were mostly related to unhandled user inputs and function reusability. Low risks were related to the ability for users to understand the game. A full table with explanations of risks is prodided here.

Figure 2. SWOT

Issues that were addressed from the above SWOT include the threat of unsecure code. Originally, my code was only located on my personal device in an unencrypted file. Now, the file is encrypted and is also stored on GitHub, a platform that allows users to adjust the security of their files by making them public or private.

3. Summary of Recommendations

Assessment of third party library security was completed as well as encypting files located on the personal device. Recommendations include continuing logging of new versions using GitHub or similar and reformatting code to be object oriented. This may be easier in a language such as C++.

2. Goals, Findings, and Recommendations

1. Assessment Goals

The purpose of this assessment was to do the following:

- Ensure that the system was not at risk of theft.
- Determine if the application was securely maintained.
- Assess the security of the code itself including clarity and organization.

2. Detailed Findings

A weakness is a software error or bug that can lead to software vulnerabilities. A software vulnerability allows the code to be exploited and is therefore greater risk. A threat is an external attack on a system.

1.0: The user will fail to locate information about the game or developers.	This is a weakness because it inhibits the access of information to the user but does not allow the code itself to be exploited.
2.0: The user will fail to understand the purpose of the game.	This is a vulnerability because a user who does not understand the game could very well “break” it.
3.0: The user will fail to change mistaken inputs.	This is a vulnerability where the weakness is “unhandled exceptions”.
4.0: Third-Party libraries are not up-to-date.	This is a vulnerability that is easily exploited. It is high risk for security issues related to the specific libraries previous downloads.
5.0: The developer will fail to log changes of code.	This is a weakness related to the developer’s ability to remember to log their changes. Without logging, there is risk of data loss.
6.0: The developer will fail to debug or edit the code easily.	This is a vulnerability related to the developer’s weakness of lack of organization.

7.0: The developer will fail to reuse functions easily.	This is a vulnerability related to the developer's weakness of lack of organization and simple functions.
8.0: The developer will fail to understand the purpose of each funciton	This is a vulnerability related to the developer's weakness of poor documentation and comments.

3. Recommendations

Issue 1.0: Third-Party Libraries:
Description: "Third-Party libraries used in code are up-to-date and have been checked to ensure no security issues exist."

Third Party libraries that were used in the project include time, datetime, threading, and random. It is recommended to assess each library for updated and security issues using the documentation found at Python.org. Open-source Python libraries are susceptible to Identity Hijacking, Typosquatting, and Dependency Vulnerabilities. Report any suspicious libraries to Python.

Issue 2.0: Authentication
Description: "PKI and other encryption and authentication methods are used to connect to cloud platform."

The project is stored on GitHub as well as a personal device. Authentication methods of logging into GitHub are used and previously passwords were stored on Google Chrome. Now, BitWarden, an open-source password manager is being used to store passwords. Bitwarden uses encryption, multi-factor authentication, and reports password breaches. To complete this protection, NordVPN is recommend so that my network connection to GitHub is in a safe environment.

Issue 3.0: Organization
Description: "Code is as efficient, clear, and easy to read as possible"

Considering the length of the project (almost 700 lines of code) it was found that there was not enough time to edit the code in such a way that it would be easy to read. For future planning, I recommend reorganizing the code to be Object Oriented. Organizing the code by category such as an inventory, career, and character object would allow for easy passability rather than using 5-10 parameters. It is recommended to reprogram this in a language like C++ that allows that flexibility.

Issue 4.0: Accounting:
Description: "Process includes logging (tracking of changes, user making changes…)"

This project previously did not have any previous versions saved. Keeping track of changes is important for the process of cleaning code and making it efficient. For the future, changing from procedural coding to object oriented will likely require access to previous versions of the code so the goal of each function can be understood. Continuous logging is also important from a security standpoint in that there is always a previous version available, even after incorrect changes have been made. If this code were to be shared or distributed, without a copy of a previous version, there would be concern about any drastic changes. It is recommended to continue to track changes using the GitHub repository or similar.

Issue 5.0: Standard Unit Testing

Unit testing was not conducted or created due to the large size of the program and lack of organization. For the future, it is recommended that after the project is organized to be object oriented, a unit test for the "daily choice" function should be created. It should assess various user inputs using the Python Unit Testing Framework and assess accordingly how to handle any exceptions or errors that occur. This is not recommended to be completed until the previous issues are handled.

3. Methodology for the Security Control Assessment

3.1.1 Risk Level Assessment

Each Business Risk has been assigned a Risk Level value of High, Moderate, or Low. The rating is, in actuality, an assessment of the priority with which each Business Risk will be viewed. The definitions in **Error! Reference source not found.** apply to risk level assessment values (based on probability and severity of risk). While Table 2 describes the estimation values used for a risk's "ease-of-fix".

Table 1 - Risk Values

Rating	Definition of Risk Rating
High Risk	Exploitation of the technical or procedural vulnerability will cause substantial harm to the business processes. Significant political, financial, and legal damage is likely to result
Moderate Risk	Exploitation of the technical or procedural vulnerability will significantly impact the confidentiality, integrity and/or availability of the system, or data. Exploitation of the vulnerability may cause moderate financial loss or public embarrassment to organization.
Low Risk	Exploitation of the technical or procedural vulnerability will cause minimal impact to operations. The confidentiality, integrity and availability of sensitive information are not at risk of compromise. Exploitation of the vulnerability may cause slight financial loss or public embarrassment
Informational	An "Informational" finding, is a risk that has been identified during this assessment which is reassigned to another Major Application (MA) or General Support System (GSS). As these already exist or are handled by a different department, the informational finding will simply be noted as it is not the responsibility of this group to create a Corrective Action Plan.
Observations	An observation risk will need to be "watched" as it may arise as a result of various changes raising it to a higher risk category. However, until and unless the change happens it remains a low risk.

Table 2 - Ease of Fix Definitions

Rating	Definition of Risk Rating
Easy	The corrective action(s) can be completed quickly with minimal resources, and without causing disruption to the system or data
Moderately Difficult	Remediation efforts will likely cause a noticeable service disruption • A vendor patch or major configuration change may be required to close the vulnerability • An upgrade to a different version of the software may be required to address the impact severity • The system may require a reconfiguration to mitigate the threat exposure • Corrective action may require construction or significant alterations to the manner in which business is undertaken
Very Difficult	The high risk of substantial service disruption makes it impractical to complete the corrective action for mission critical systems without careful scheduling • An obscure, hard-to-find vendor patch may be required to close the vulnerability • Significant, time-consuming configuration changes may be required to address the threat exposure or impact severity • Corrective action requires major construction or redesign of an entire business process
No Known Fix	No known solution to the problem currently exists. The Risk may require the Business Owner to: • Discontinue use of the software or protocol • Isolate the information system within the enterprise, thereby eliminating reliance on the system In some cases, the vulnerability is due to a design-level flaw that cannot be resolved through the application of vendor patches or the reconfiguration of the system. If the system is critical and must be used to support on-going business functions, no less than quarterly monitoring shall be conducted by the Business Owner, and reviewed by IS Management, to validate that security incidents have not occurred

3.1.2 Tests and Analyses

As previously stated, standard user testing was not created or conducted due to the size of the program and the disorganized nature

3.1.3 Tools

File encryption was completed using Windows 10 file properties and logging was completed using GitHub.

4. Figures and Code

```
general_store_calculator(main_month, main_funds, main_oxen_spent,
                         main_food_spent, main_clothing_spent,
                         main_ammunition_spent,
                         main_spare_parts_spent, main_amount_spent,
                         main_oxen, main_food, main_clothing,
                         main_ammunition, main_spare_parts,
                         main_pay_items, main_exit_store)

main_inventory = inventory(main_oxen_spent, main_food_spent,
                           main_clothing_spent, main_ammunition_spent,
                           main_spare_parts_spent)
```

Figure 3.0

Figure 3.0 Shows the inefficiency of functions that contain multiple parameters. It would be extremely easy to call the function incorrectly or use the wrong variable.

```
# This function will display the date, their item options, how much they've
# spent (Will start at 00.00), and how much money they have. This function
# is used in the general_store_calculator function so that after each
# item is selected and purchased, the general_store() function will display
# the new amount.
def general_store(local_month, local_funds, local_oxen, local_food,
                  local_clothing, local_ammunition, local_spare_parts,
                  local_amount):
    date = datetime.datetime(1848, local_month + 2, 1)
    print("-" * 150)
    print("\nLou's General Store \nIndependence, Missouri")
    print(date.strftime("%B %d, %Y\n"))  # Date/Time ---> Source #13
    print("Items Available:         Amount spent per item:")
    print(" 1. Oxen                 $" + format(local_oxen, '.2f'))
    print(" 2. Food                 $" + format(local_food, '.2f'))
    print(" 3. Clothing             $" + format(local_clothing, '.2f'))
    print(" 4. Ammunition           $" + format(local_ammunition, '.2f'))
    print(" 5. Spare Parts          $" + format(local_spare_parts, '.2f')
          )
    print("-" * 50)
    print("Amount you have:         $" + format(local_funds, '.2f'))
    print("Total cost:              $" + format(local_amount, '.2f'))
    print('\nType "Pay" to Pay for Items')  # Pay for items
    print('Type "Exit" to exit the store\n')  # Leave Store
    print("-" * 150)
```

Figure 4.0

Figure 4.0 shows the definition of one of these complicated functions. Having an Inventory as an object would have made this function more usable.

```
        if str(item_choice) in pay_items:
            # If the user has enough money, they will pay.
            if (local_funds - local_amount) >= 0.00:
                print("Your total is $" +
                      str(local_oxen + local_food + local_clothing +
                          local_ammunition + local_spare_parts))
                time.sleep(2)
                print("You now have $" + str(local_funds - local_amount) +
                      " left.")
                time.sleep(2)
                local_funds -= local_amount
            # If the user does not have enough money, they will be kicked out.
            elif (local_funds - local_amount) < 0.00:
                print("'What in tarnation?! You haven't got a penny "
                      "to your name. Get out of my store 'for I "
                      "call the police.'")
                time.sleep(2)
                print("You have exited the store.")
                user_making_choice = False
                time.sleep(1)
        elif str(item_choice) in exit_store:
            # If the user does not want to purchase anything they can exit.
            print("You have exited the store.")
            user_making_choice = False
            time.sleep(2)
        elif int(item_choice) == oxen:
```

Figure 5.0

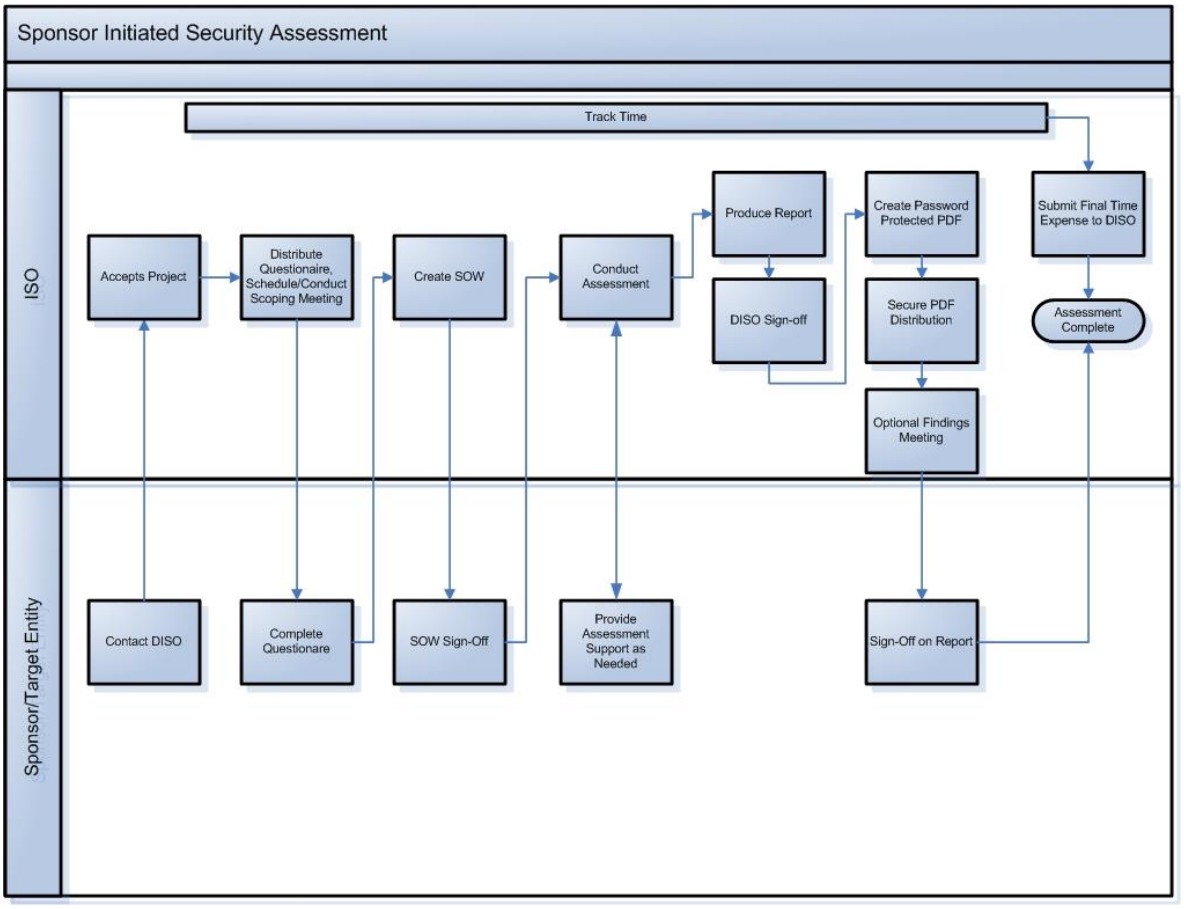
Figure 5.0 is a small snippet of the general_store_calculator() function. This is one example of the poor organization and logic of the functions contained in this program. This function alone is over 100 lines of code long.

```
# Title: The Oregon Trail Game
#   Dev: Samantha Walsh
#  Date: 9/10/2021

# Description:
# This program is a recreation of The Oregon Trail game developed by
# Bill Heinemann, Don Rawitsch, and Paul Dillenberger

# Sources:
# 1  (Oregon Trail Info)https://en.wikipedia.org/wiki/The_Oregon_Trail_(series)
# 2  (Oregon Trail Game)https://classicreload.com/oregon-trail.html
# 3  (If statements)https://www.w3schools.com/python/python_conditions.asp
# 4  (Booleans)https://www.w3schools.com/python/python_booleans.asp
# 5  (Functions)https://www.w3schools.com/python/python_functions.asp
# 6  (Time module)https://www.programiz.com/python-programming/time
# 7  (Operators)https://www.w3schools.com/python/python_operators.asp
# 8  (While loops)https://www.w3schools.com/python/python_while_loops.asp
# 9  (While loops)https://wiki.python.org/moin/WhileLoop
# 10 (Exceptions)https://www.w3schools.com/python/python_ref_exceptions.asp
# 11 (Lists)https://www.w3schools.com/python/python_lists_comprehension.asp
# 12 (Date/Time Module)https://www.w3schools.com/python/python_datetime.asp
# 13 (String Formatting)https://realpython.com/python-string-formatting/
# 14 (Try/Except)https://www.w3schools.com/python/python_try_except.asp
# 15 (.upper)https://www.w3schools.com/python/ref_string_upper.asp
# 16 Andrew Krupp (Student Assistant)
# 17 Prof. Scott Vanselow
```

Figure 6.0

Figure 6.0 shows the header of the program. This is one of the few pieces of the program that is well formatted and documented.

4.1.1 Process or Data flow of System (this one just describes the process for requesting), use-cases, security checklist, graphs, etc.



5. Works Cited

Asif, By Kaynat. *How to Convert a Python Program to C/C++ Code*. linuxhint.com/convert-python-program-to-cpp.

"Functional Programming HOWTO." *Python Documentation*, docs.python.org/3/howto/functional.html.

GeeksforGeeks. "Object Oriented Programming in C." *GeeksforGeeks*, Apr. 2023, www.geeksforgeeks.org/object-oriented-programming-in-cpp.

"Viewing Push Logs - GitHub Enterprise Server 3.4 Docs." *GitHub Docs*, docs.github.com/en/enterprise-server@3.4/admin/monitoring-activity-in-your-enterprise/exploring-user-activity/viewing-push-logs.

"Viewing the Branch History - GitHub Docs." *GitHub Docs*, docs.github.com/en/desktop/contributing-and-collaborating-using-github-desktop/making-changes-in-a-branch/viewing-the-branch-history.

"Unit Testing Tutorial | CLion." *CLion Help*, www.jetbrains.com/help/clion/unit-testing-tutorial.html.

"Unittest — Unit Testing Framework." *Python Documentation*, docs.python.org/3/library/unittest.html.

"Errors and Exceptions." *Python Documentation*, docs.python.org/3/tutorial/errors.html.