



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

Security Assessment Report

Version N.1

April 22, 2023

Security Assessment – Calculator

Table of Contents

1. Summary	3
1. Assessment Scope	3
2. Summary of Findings	3
3. Summary of Recommendations	5
2. Goals, Findings, and Recommendations	5
1. Assessment Goals	5
2. Detailed Findings	6
2.2.1 Internal Actor Threats	6
2.2.2 Standard Unit Testing	6
2.2.3 Endpoints of the Program	7
2.2.4 Global Variables	7
2.2.5 Remote Hosting	8
2.2.6 Transparency of Logic	8
2.2.7 Physical and Network Security	8
2.2.8 Automated Security and Policies	8
3. Recommendations	9
2.3.1 Internal Actor Threats	9
2.3.2 Standard Unit Testing	9
2.3.3 Endpoints of the Program	9
2.3.4 Global Variables	10
2.3.5 Remote Hosting	10
2.3.6 Transparency of Logic	11
2.3.7 Physical and Network Security	11
2.3.8 Automated Security and Policies	11
3. Methodology for the Security Control Assessment	12
3.1.1 Risk Level Assessment	12
3.1.2 Tests and Analyses	13
3.1.3 Tools	14
4. Figures and Code	15
4.1.1 Access Control and Network Security Checklist	15
4.1.2 Other figures of code	16
5. Works Cited	19

1. Summary

The goal of this security assessment is to identify the risks and vulnerabilities that are present in the Calculator program. If able, these vulnerabilities and risks should be eradicated or mitigated.

1. Assessment Scope

The program uses HTML, CSS, and JavaScript as its main languages used to develop the software. The program is hosted on GitHub and is released using GitHub Pages. The program was tested using Google Chrome and the Chrome Developer Tools. The code editor that was utilized is Visual Studio Code. Some major limitations that should be addressed are that there was only one person who developed and tested the software. The software was also only thoroughly tested using one OS, specifically a Windows PC.

2. Summary of Findings

Of the findings discovered during our assessment, 2 were considered Emergency level risks, 2 Major level risks, 3 Moderate, 1 Minor, and 3 Negatable risks. The SWOT used for planning the assessment is broken down as shown in Figure 1.

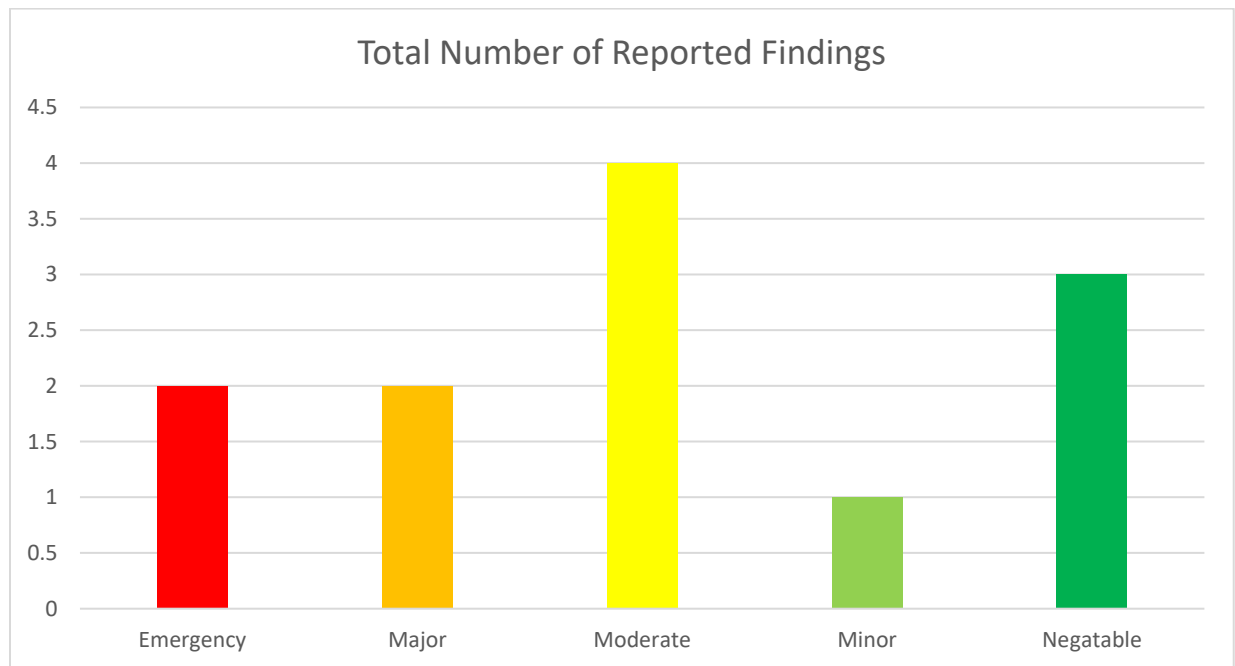


Figure 1. Findings by Risk Level

Figure 1 is a summary of the findings from the detailed Risk Assessment Matrix that is further detailed in [Figure 6](#). As illustrated above, the majority of the risks fell into the Moderate level which means that they should be addressed and are worth our attention when eradicating and mitigating these risks.



Figure 2. SWOT

The major issues that were found in the SWOT analysis were that the program utilized many variables and values that are stored in the global scope which allows for others to view and change these values in potentially malicious ways. Defining variables in the global scope does not enforce access control within a program which makes it easier for hackers to infiltrate a system of software. The original code had not been scanned for these vulnerabilities and developers must decrease the number of endpoints that exist in the logic of the program. Another major issue that should also be addressed is the integration of type checking in the logic of the program to ensure that the program is only processing a specific range of data from the user.

3. Summary of Recommendations

The changes made to increase the security and mitigate some of the vulnerabilities include implementing encapsulation for variables that were previously global, integrating type checking for user-inputted values, establishing a security policy, and enabling automatic code scanning. The security policy and automatic code scanning are both implemented using the features of GitHub. The code scanning is done using CodeQL. Changes that are still necessary include implementing unit testing to ensure that the software is working properly and that there are no vulnerabilities within these errors. Another necessary change is identifying and securing any vulnerable JavaScript libraries that are utilized.

2. Goals, Findings, and Recommendations

1. Assessment Goals

The main goal of this assessment is to identify potential threats to the program “Calculator” and the risks, and the level of risk associated with these threats. A risk is defined as “the potential for an unwanted or adverse outcome resulting from an incident, event, or occurrence, as determined by the likelihood that a particular threat will exploit a particular vulnerability, with the associated consequences.” (NICCS, 2023) While a threat is defined by the NICCS as being “A circumstance or event that has or indicates the potential to exploit vulnerabilities and to adversely impact (create adverse consequences for) organizational operations, organizational assets (including information and information systems), individuals, other organizations, or society.” (NICCS, 2023) In addition, we are aimed at mitigating or eradicating these risks under a proposed plan of action or simply by acting.

The purpose of this assessment was also to ensure that the system complied with the Access Control and Network Security requirements specified in the [Access Control and Network Security Checklist](#). For example, as specified we must ensure that “a cloud-based platform is being used for access control with only public available items being readable by general public.” We must also ensure that “standard unit testing is being used” and “the endpoints of the system have been accounted for and secured as much as possible”. In other words, this security assessment should account for issues regarding access controls and the network security of the program.

Another goal of this assessment is to ensure that the application has been securely maintained. This includes assessing the risks that are specified and analyzed in this assessment. Along with this, the goal is to ensure that these identified risks are mitigated or eradicated, if possible. To continue, the purpose of this assessment is to identify the need for the most important security investments. As stated in the article “Performing a Security Risk Assessment”, “Added security usually involves additional expense. Since this does not generate easily identifiable income, justifying the expense is often difficult.” (Schmittling & Munns, 2010) Because of this, security assessments allow us to be aware of the critical security issues that exist in a program and decide which ones are the most important to handle.

2. Detailed Findings

Risk Rating (Refer to Table 1)	Vulnerabilities
Informational	Internal Actor Threats
Moderate Risk	Standard Unit Testing
High Risk	Endpoints of the Program
High Risk	Global Variables
Observations	Remote Hosting
Observations	Transparency of Logic
Observations	Physical Security
Moderate Risk	Automated Security and Policies

Figure 4. Vulnerability Risk Ratings

2.2.1 Internal Actor Threats

An internal actor threat can be defined as an individual or group within the system that facilitates or has the intent to harm the program or system. This can range from employees or actors in the system accidentally sharing private information to deliberately exploiting any vulnerabilities for personal gain. In general, human error is always a threat to a system or program.

This was determined to be a vulnerability based on the [Access Control and Network Security Checklist](#). This vulnerability is classified as an **Informational Risk** (Table 1). Although internal actor threats are a vulnerability of the program, this document should focus more on the program itself and its interaction with other (technological) systems. Had the production of this program been larger, then we would be able to divert this finding to a human resources department or other that deals with the policies that need to be created to defend against internal actors in their purposeful or accidental threats.

Internal Actor threats have not been accounted for and policies or planning are not currently in place for these for the Calculator. Some threats are currently accounted for during the SWOT analysis and Risk assessment previously completed. However, many others may not be accounted for and should be expanded upon further.

2.2.2 Standard Unit Testing

Standard Unit “tests are written by developers to evaluate system behavior at different levels during implementation. As their names imply, unit tests focus on the behavior of isolated ‘units’ (classes or methods) in the code” (Nicole Gonzalez, 2021). Standard unit testing is commonly used to check the logic of a program to ensure that the program is doing as it should. However, standard unit testing can also be used to handle security concerns in a program. For example, standard unit testing can be used to test Token Authentication, Token Manipulation, Refresh Tokens, Login, and Logout functionality (Nicole Gonzalez, 2021).

Security Assessment – Calculator

“Modern development models recognize the benefits of early and repeated testing and work to ‘shift testing left’ by incorporating appropriate testing activities into each lifecycle phase” (Nicole Gonzalez, 2021). As illustrated in the journal article, implementing unit testing early in the development process of a product allows for faster recognition of security vulnerabilities and logical errors. It also uncovers ways that hackers may be able to infiltrate a system.

Currently, Standard Unit Testing has not been implemented for the functionalities of the program, including the basic calculations that are at the core of the program. In the scope of this program, standard unit testing was determined to be a vulnerability based on the [Access Control and Network Security Checklist](#). This vulnerability is classified as a [Moderate Risk](#) (Table 1). Because of the lack of some of these technologies, the lack of standard unit testing does not hinder the program as much as other systems. However, this still applies in our case as errors in the logic, such as the lack of type checking, can allow malicious users to inject code or malware into the program.

2.2.3 Endpoints of the Program

An endpoint of a program or endpoint security is regarded as the defense of “what is now thought of as an enterprise’s perimeter – the devices that are the gateways into the network – from known as well as unknown threats.” (VMware, n.d.) In essence, an endpoint can be described as an entry point into a network or system. This can be applied to our current program by illustrating how users can interact with the program itself.

This was determined to be a vulnerability based on the [Access Control and Network Security Checklist](#). This vulnerability is classified as [High Risk](#) (Table 1). As stated before, the scope of this program and the size of its production can be seen as positive as there is not a large network that we are currently concerned with. Although we have the capability for users of the program to interact with our current network, it can be said that there are only a few endpoints that can be infiltrated which coincides with the utilization of remote hosting. However, this is still a high-risk vulnerability because the people interacting with our program should be accounted for, especially with how they interact with the program. This item is also a high priority because it is one of the main ways that the program could be hacked. The endpoints of the program are vital to the logic of the program, it essentially is the data store for the site and could be easily exploited since it is not secure.

2.2.4 Global Variables

Global variables are variables that are within the scope of the entire logic of the program. The main logic of the program is currently written in JavaScript. Because of this, “Global variables can be modified by any part of the script, which makes it difficult to keep track of changes and can potentially introduce security vulnerabilities.” (Home, 2023) This expands to the utilization of Chrome Developer Tools and how simple it is to access and utilize scripts within the program. This vulnerability is classified as a [High Risk](#) (Table 1).

2.2.5 Remote Hosting

Using remote hosting, in our case, GitHub, could be a potential vulnerability of the program. Since the site is hosted using this software, our security can only be as good as GitHub's. If there are any security issues with GitHub, this will also affect the program. The software is stored and released on GitHub Pages. If this system is down or gets shut down, then the program cannot be used until it is moved to another hosting site. This vulnerability is classified as an [Observation](#) (Table 1).

2.2.6 Transparency of Logic

All the code is currently open-source and stored on GitHub. This can be seen as both a positive and a negative. Our focus, however, is how this is concerned with the security of the program. Because the code can be viewed by anyone, anyone can analyze the code to figure out a way to infiltrate it. Because of this, the software is vulnerable to code injection because of the lack of type checking and the transparency of logic. It is much easier to hack a piece of software knowing how the logic is structured and how it works. This vulnerability is classified as an [Observation](#) (Table 1).

2.2.7 Physical and Network Security

The physical and general network security of the program should be factored into identifying some of the vulnerabilities of the program. Some of the general physical vulnerabilities include if my laptop gets stolen or my personal laptop gets broken into. If this occurs, the actor will have access to the original code base in addition to access to some of my accounts that cause an unauthorized change in the program. To include network security, I do not currently use a VPN so there may be some vulnerabilities that I am exposed to when on personal or public Wi-Fi networks. This vulnerability is classified as an [Observation](#) (Table 1) as there is only so much that we can do to prevent these risks.

2.2.8 Automated Security and Policies

Previously, the program did not have any open license enabled for its security. As paraphrased from the MIT license that can be found in the GitHub repository for the calculator program “users of software using an MIT License are permitted to use, copy, modify, merge publish, distribute, sublicense and sell copies of the software” (MIT, 2022) as long as credit is given to the original author/publisher. Because of this, refraining from including a license allows others to use the code in the repository without the authors getting recognition. This is a security issue since anyone can use your code without giving you credit.

The other policy that is missing in the system is the enabling of automated resources that are included on GitHub. GitHub features some security functionalities like automatic Code Scanning that can help detect vulnerabilities in code. This can help reduce any security vulnerabilities that may exist, and it would be a vulnerability to not use the software. The current program also did not have a security policy that would inform viewers of the current state of the program and how they may go about reporting bugs or vulnerabilities. This vulnerability is classified as a [Moderate Risk](#) (Table 1).

3. Recommendations

Risk Rating (Refer to Table 2)	Vulnerabilities
Moderately Difficult	Internal Actor Threats
Moderately Difficult	Standard Unit Testing
Very Difficult	Endpoints of the Program
Moderately Difficult	Global Variables
No Known Fix	Remote Hosting
No Known Fix	Transparency of Logic
Easy	Physical Security
Easy	Automated Security and Policies

Figure 5. Vulnerability Ease of Fix Ratings

2.3.1 Internal Actor Threats

I consider this to be difficult because it is hard to think about all the ways that internal actors can be a threat to a system. Given this context, it is slightly easier since I am the only internal actor, but it can be difficult to think about all the scenarios that may occur. Not only should they be accounted for, but policies and planning should be detailed to prevent any of these threats from becoming a reality. This process can be very difficult and time-consuming, so that is why I labeled it as difficult. To fix this, however, some research and documentation should be allocated to identifying some of the potential threats to the system that the only actor may cause. This vulnerability is classified as **Moderately Difficult** to Fix (Table 2).

2.3.2 Standard Unit Testing

I need to implement some type of testing software that will test some of the different test cases needed to ensure that the software is working properly. The major testing framework that could be used is Jest which is made to test JavaScript which is the primary language for the program. This vulnerability is classified as **Moderately Difficult** to Fix (Table 2) because unit testing can often be difficult to not only create but to go back and ensure that all the tests are in correspondence with the program. In other words, it can often be troublesome because testing requires a lot of editing to ensure that all the unit tests pass. Other testing frameworks like Mocha and Cypress can be utilized so it is important to assess the pros and cons of each framework before I begin unit testing.

2.3.3 Endpoints of the Program

Securing the endpoints of the program connects to the fixes of **Global Variables** and **Transparency of Logic**. Should the program expand, for example, by creating a login system, some form of accounting and encryption should be integrated into the system to track who is

Security Assessment – Calculator

interacting with the system. Accounting can help identify a “bad actor” of the program. This vulnerability is classified as [Very Difficult](#) to Fix (Table 2).

2.3.4 Global Variables

Previously, 3 major variables were declared and initialized globally which allowed for easy access to some of the main functionalities of the calculator. By encapsulating all of these variables in a class complete with getters and setters, it is difficult for hackers to inject their code into my program. Integrating encapsulation also allows for fewer endpoints within the logic of the program.

There were still some global variables that were present in my code after removing and encapsulating the 3 major variables from above. To improve the security of these variables, I encapsulated them within the functions of the logic so that they could not be easily accessed using the console. I also noticed that there was a global variable "container" that was declared but never utilized, so I also decided to remove that as this is an unnecessary back door. The DOM accessors to the buttons and the display were deleted as global variables and declared and initialized inside the functions of the code.

Removing these variables as global variables reduces the risk and the capability of hackers being able to manipulate and inject code into the program. Integrating encapsulation also allows for fewer endpoints within the logic. I also took another look at the floating-point precision of the floating-point math of the program. I was reassured that this was mainly handled at the beginning of development with the "roundDecimals" function. However, I did need to add the utilization of this function to the add function to ensure that the math would output correct values. The roundDecimals function had already been implemented into the other operation functions.

This vulnerability is classified as [Moderately Difficult](#) to Fix (Table 2) as much of the logic of the program had to be readjusted to accommodate for encapsulation.

See Figures 7 – 9 in [4.1.2 Other figures of code](#)

2.3.5 Remote Hosting

The cloud platform used is GitHub which provides controls like read-only permissions and settings to change access controls that account for this issue.

Using GitHub, as my cloud platform, is a great advantage in improving the security of my software. Since GitHub has good security, by association, my software also has good security. It is to be noted that the software on GitHub has good security, using GitHub does not necessarily improve the security of my network and devices.

GitHub uses SSH certificate authorities to authorize users that attempt to interact with Git through GitHub. This allows GitHub to authenticate users through different methods. When interacting with GitHub through Git, the user is prompted to enter a password or temporary private key. GitHub also uses 2-factor authentication when editing any important information in

Security Assessment – Calculator

a repository. The cloud platform, GitHub, implements the functionality that only collaborators can contribute and edit to the repository. At this time there are no collaborators, but the functionality is there if needed in the future.

If any issues with GitHub may arise, for my production, I have no influence on fixing those potential issues. This vulnerability is classified as [No Known Fix](#) (Table 2).

2.3.6 Transparency of Logic

Within the new class that holds all of my important variables, I implemented type-checking to ensure that I am only getting a specific range of data from the user. The integration of type checking constructs a layer of protection from code injection. So, even though the code is open-source hackers cannot inject code as it will throw an error. The other fixes for Transparency of Logic fall under [Automated Security and Policies](#). Even though the code can be viewed by anyone some policies have been implemented to ensure that the program stays protected.

This vulnerability is classified as [No Known Fix](#) (Table 2) since we are not able to “hide” the logic of the code completely. I could make the repository private, but in Chrome Developer Tools, anyone can view the code for a site unless it is using React (or technology of the like), as there is a functionality to hide the code from Chrome Developer Tools.

2.3.7 Physical and Network Security

The physical security of my personal computer is adequate. I use biometrics to log in to my computer or a passcode (which I am the only one that knows). I also have the most recent update of Windows 11. It is to be noted that both Samsung and Windows updates are installed and fully updated on my device. As a precaution, I continue to update my computer to ensure that there are known vulnerabilities that may exist on my operating system. To ensure some network security, I refrain from accessing private information on public Wi-Fi and also have a virtual machine that I can use to test any suspicious links or activity. This vulnerability is classified as [Easy to Fix](#) (Table 2).

2.3.8 Automated Security and Policies

GitHub offers some methods in improving the security of a program and a repository. To improve the security of the program I set up a Security Policy to inform users about the current state of the program. This includes any vulnerabilities that we are aware of and how users can submit any vulnerabilities that they might find while using the program or reviewing the code. The security policy linked below (See Figure 10 in [4.1.2 Other figures of code](#)) describes these concepts in further detail.

I have also implemented an open license, specifically the MIT license, to ensure that I can get credit for the code that I have written. This increases the security of my code, as no one can use it without referencing what I have written. In addition, GitHub also has a Code Scanning functionality that uses CodeQL to scan for any vulnerabilities that may exist in the files of the program. Currently, GitHub is actively checking for vulnerabilities within my JS file which

Security Assessment – Calculator

holds the main logic of my code. The setup of CodeQL and its results are linked below. As of now, no vulnerabilities have been found. This vulnerability is classified as [Easy](#) to Fix (Table 2).

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 <ul style="list-style-type: none">• 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 <ul style="list-style-type: none">• 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: <ul style="list-style-type: none">• Discontinue use of the software or protocol

Security Assessment – Calculator

Rating	Definition of Risk Rating
	<ul style="list-style-type: none"> Isolate the information system within the enterprise, thereby eliminating reliance on the system <p>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</p>

		Probability ----->				
Severity		Frequent	Probable	Likely	Possible	Rare
I	Emergency				Hackers inject malicious code into software through global variables or input.	My Github account gets hacked or broken into.
I	Major				The main storage of values can be accessed globally / in the console.	Someone with access to the repository or my laptop changes the code maliciously.
I	Moderate		Lack of unit testing may hide vulnerabilities that persist in the logic of the program.	All code can be viewed in various ways; GitHub and Chrome Developer Tools .	Lack of security policy makes the software more susceptible to threats.	Github Pages could get disabled or shut down.
I	Minor				The program files get deleted from personal computer.	
I V	Negatable				The program files get deleted from Github.	1. My personal laptop gets stolen. 2. My personal laptop gets hacked or broken into.

Figure 6. Risk Assessment Matrix

Most of the findings in the Risk Assessment Matrix were accurate. Some risks may be slightly far-fetched, but most happen to be in correspondence with the brief and they are elaborated in [Goals, Findings, and Recommendations](#).

3.1.2 Tests and Analyses

3.1.2.1 White Box Testing

I tested my software specifically by conducting a White Box Test. White Box testing is a “low-level process usually done by a software developer who knows programming and application logic. The tester will be testing the internal components of the application or piece of software and, for example, ensuring that specific functions work correctly and within a reasonable amount of time.” (cmnatic, 2021) This was my default testing as I was already aware of the logic of the code and the potential vulnerabilities of the system.

3.1.2.2 Research into System Vulnerabilities

During testing and analyses, I conducted quite a bit of research to understand some of the potential vulnerabilities of the system. This included many of the concepts we covered in the course which included integrating type checking, establishing security policies, and my general knowledge of understanding that global variables are not secure. Much of my research was cited in the [Detailed Findings](#) section. Further research was also conducted after completing major assignments like the [Access Control and Network Security Checklist](#). These assignments provided guidance on what to focus on and how a program may be susceptible to “bad actors”

3.1.3 Tools

This was completed using:

- Chrome Developer Tools
 - This was used to test and find how the global variables could be accessed and change the results of the current code.
- Visual Studio Code
 - This was used to access and edit the code base to fix the security issues that were found.
- GitHub
 - This was used to implement a security policy, enable an open license, and enable code scanning to check for vulnerabilities in the code.
- Linux Command Line
 - The Linux command line was used for file management and navigation. I was specifically using Windows Subsystem for Linux (WSL) so that I could use it on my operating system. I also used the command line to use Git as my VCS (Version Control System).

4. Figures and Code

4.1.1 Access Control and Network Security Checklist

Access Control and Network Security Checklist						
ID	Security Control Issue	Applicable (Y/N)	Complete (Y/N/NA)	To be Done Priority (High, Mid, Low)	Ease of fix (Easy, Moderate, Difficult, Not Fixable)	Full Description of processes to address issue
1	A cloud based platform is being used for access control with only public available items being readable by general public.	Y	Y	Low	Easy	The cloud platform used is Github which provides controls like read-only permissions and settings to change access controls that account for this issue.
2	The cloud based platform provides for hiding information and this is used to protect sensitive information and code.	N	N/A	Low	Easy	N/A; Note the cloud-based platform does have this functionality but it is not needed for the purposes of the project.
3	OS access controls are used to only allow authorized changes to be made to code.	Y	Y	Low	Easy	OS access controls were automatically implemented when the project was created. I, the user, and the admin are the only users that have read and write access.
4	Platform user groups are used to only allow changes to be made to code by authorized individuals.	Y	Y	Low	Easy	The cloud platform, Github, implements the functionality that only collaborators are able to contribute and edit to the repository. At this time there are no collaborators but the functionality is there if needed in the future.
5	Backup Policy is in place and being used.	N	N/A	Low	Moderate	N/A
6	Third-Party libraries used in code are up-to-date and have been checked to ensure no security issues exist.	N	N/A	Low	Easy	N/A
7	Physical Security of actual computer code is stored on is adequate	Y	Y	Low	Easy	The physical security of my personal computer is adequate. I use biometrics to login to my computer or a passcode (which I am the only that knows it). I also have the most recent update of Windows 11. It is to be noted that both Samsung and Windows updates are installed on my device.
8	Accounting: Logging is integrated into the code itself (for exceptions, errors, and user input failures at minimum)	N	N/A	Low	Moderate	N/A
9	Accounting: Process includes logging (tracking of changes, user making changes, access attempts, etc)	N	N/A	Low	Moderate	N/A
10	PKI and other encryption and authentication methods are used to connect to cloud platform	Y	Y	Low	Easy	Github uses SSH certificate authorities to authorize users that attempt to interact with Git through Github. This allows Github to authenticate users through different methods. When interacting with Github through Git, the user is prompted to enter a password or temporary private key. Github also uses 2 factor authentication when editing any important information in a repository.
11	Internal Actor threats are accounted for and policies/planning is in place for these.	Y	N	Mid	Difficult	Some threats are currently accounted for during the SWOT analysis and Risk assessment previously completed. However, there are many others that may not be accounted for as human error, in general, is a threat to the system. However, as I am the only internal actor, it is imperative that I take the proper measures to secure my physical and network security. I currently do not have any standard/automated unit testing for the system, however this is fairly easy to implement. I need to implement some type of testing software that will test some of the different test cases needed to ensure that the software is working properly.
12	Standard Unit Testing used	Y	N	Mid	Easy	
13	Security Testing used (the type varies)	N	N/A	Low	Moderate	N/A
14	The endpoints of the system have been accounted for and secured as much as possible.	Y	N	High	Moderate	Most of the endpoints have been accounted for but have not been properly secured. To address this issue I must do more research as to what those endpoints may be and how I should go about securing them. Then I must secure them.

Questions:

Which of these were accounted for on your SWOT or Risk Assessment and how have you started adding countermeasures for them (or how will you start)?

1, 7, 10, and 14 were accounted for in my SWOT Analysis and Risk Assessment. Some countermeasures that I already implemented is that I have made my physical and network security more adequate. I have installed many updates on my computer since then. Using Github, as my cloud platform is a great advantage in improving the security of my software. Since Github has good security, by association, my software also has good security. It is to be noted that the software on Github has good security, using Github does not necessarily improve the security of my network and devices. The last countermeasure that needs to be implemented is related to the endpoints of the software. It was noted in both the SWOT and Risk Assessment that this is a major issue. One possible solution is to edit the logic of the program to increase security, this would allow for less global variables in the program, more specifically the global variables would not hold a key data store for the program.

Select a High Priority item - why do you consider it "high priority"?

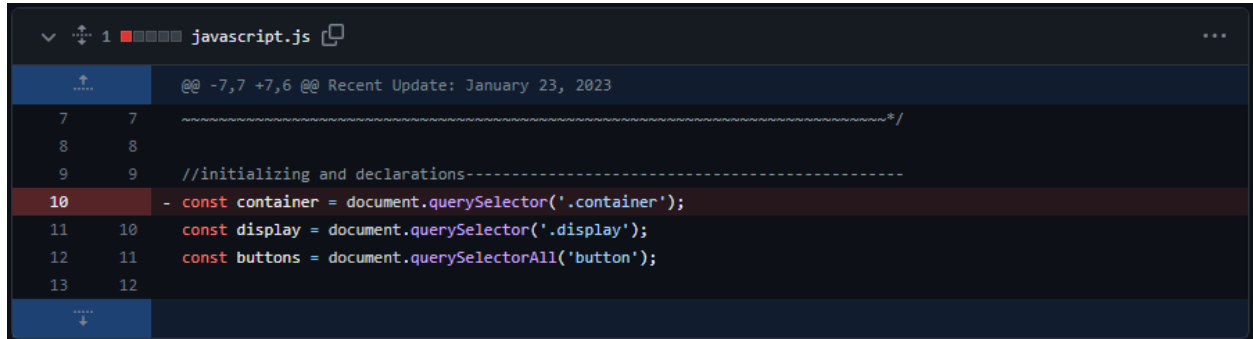
The only high priority item I have is 14, "The endpoints of the system have been accounted for and secured as much as possible." I consider this to be a high priority item because it is the main flaw in the software. Many of the other items have been addressed by using other secure systems and software. This item is also high priority because it is one of the main ways that my software could be hacked. The endpoint has been accounted for but not necessarily addressed which is a big issue. The endpoint is vital to the logic of the program, it essentially is the data store for the site, and could be easily exploited since it is not secure.

Select a "Not Fixable" or "Difficult" item - why did you select this value for it?

The only difficult item I have is 11, "Internal Actor threats are accounted for and policies/planning is in place for these." I consider this to be difficult because it is hard to think about all the ways that internal actors can be a threat to a system. Given this context, it is a slightly easier since I am the only internal actor, but it can be difficult to think about all the scenarios that may occur. Not only should they be accounted for, but policies and planning should be detailed to prevent any of these threats from becoming a reality. This process can be very difficult and time consuming, so that is why I labeled it as difficult.

Security Assessment – Calculator

4.1.2 Other figures of code



```
1  javascript.js
@@ -7,7 +7,6 @@ Recent Update: January 23, 2023
7  7  ..... */
8  8
9  9  //initializing and declarations-----
10 - const container = document.querySelector('.container');
11 10  const display = document.querySelector('.display');
12 11  const buttons = document.querySelectorAll('button');
13 12
.....
+
```

Figure 7. Removal of Unused Global Variable

Security Assessment – Calculator

```

9      9      //initializing and declarations-----
10     - const display = document.querySelector('.display');
11     - const buttons = document.querySelectorAll('button');
12     -
13     10     let numbers = new Set("1234567890.");
14     11     let operators = new Set("+-*/*");
15     12
+----+
+
+----+
85     82     //functions -----
86     83
87     84     //operations
88     - function add() {
85     + function add(display) {
89     86         //must be numbers because of string concatenation as default
90     87         let tempArr = Calc.getExpressionValues();
91     88         tempArr.push(roundDecimals(parseFloat(tempArr.pop()) + parseFloat(tempArr.pop())));
+----+
+
+----+
94     91         display.textContent = (Calc.getExpressionValues())[0];
95     92     };
96     93
97     - function subtract() {
94     + function subtract(display) {
98     95         let tempArr = Calc.getExpressionValues();
99     96         let tempVal = tempArr.pop();
100    97         tempArr.push(roundDecimals(tempArr.pop() - tempVal));
+----+
+
+----+
103    100         display.textContent = (Calc.getExpressionValues())[0];
104    101     };
105    102
106    - function multiply() {
103    + function multiply(display) {
107    104         let tempArr = Calc.getExpressionValues();
108    105         let curr = roundDecimals(tempArr.pop() * tempArr.pop())
109    106         tempArr.push(curr);
+----+
+
+----+
112    109         display.textContent = (Calc.getExpressionValues())[0];
113    110     };
114    111
115    - function divide() {
112    + function divide(display) {

```

Figure 8. Part 1 Encapsulation of “buttons” and “display” variable

Security Assessment – Calculator

```
112 + function divide(display) {
116 113   let tempArr = Calc.getExpressionValues();
117 114   let temp = tempArr.pop();
118 115   tempArr.push(roundDecimals(tempArr.pop() / temp));
@@ -136,19 +133,21 @@ function roundDecimals(result) {
136 133
137 134   //decipher which operation to do
138 135   function operate() {
136 +   const display = document.querySelector('.display');
137 +
139 138   let currOperator = Calc.getCurrOperation();
140 139
141 140   if(currOperator == '+') {
142 -   add();
141 +   add(display);
143 142   }
144 143   else if(currOperator == '-') {
145 -   subtract();
144 +   subtract(display);
146 145   }
147 146   else if(currOperator == '*') {
148 -   multiply();
147 +   multiply(display);
149 148   }
150 149   else if(currOperator == '/') {
151 -   divide();
150 +   divide(display);
152 151   }
153 152
154 153   Calc.setCurrOperation("");
@@ -160,6 +159,9 @@ function operate() {
160 159   function populateDisplay() {
161 160     let idx = 0;
162 161
162 +   const buttons = document.querySelectorAll('button');
163 +   const display = document.querySelector('.display');
164 +
163 165   buttons.forEach(button => button.addEventListener('click', button => {
164 166
165 167     let temp = button.composedPath()[0].id;
```

Figure 9. Part 2 Encapsulation of “buttons” and “display” variable

Security Assessment – Calculator

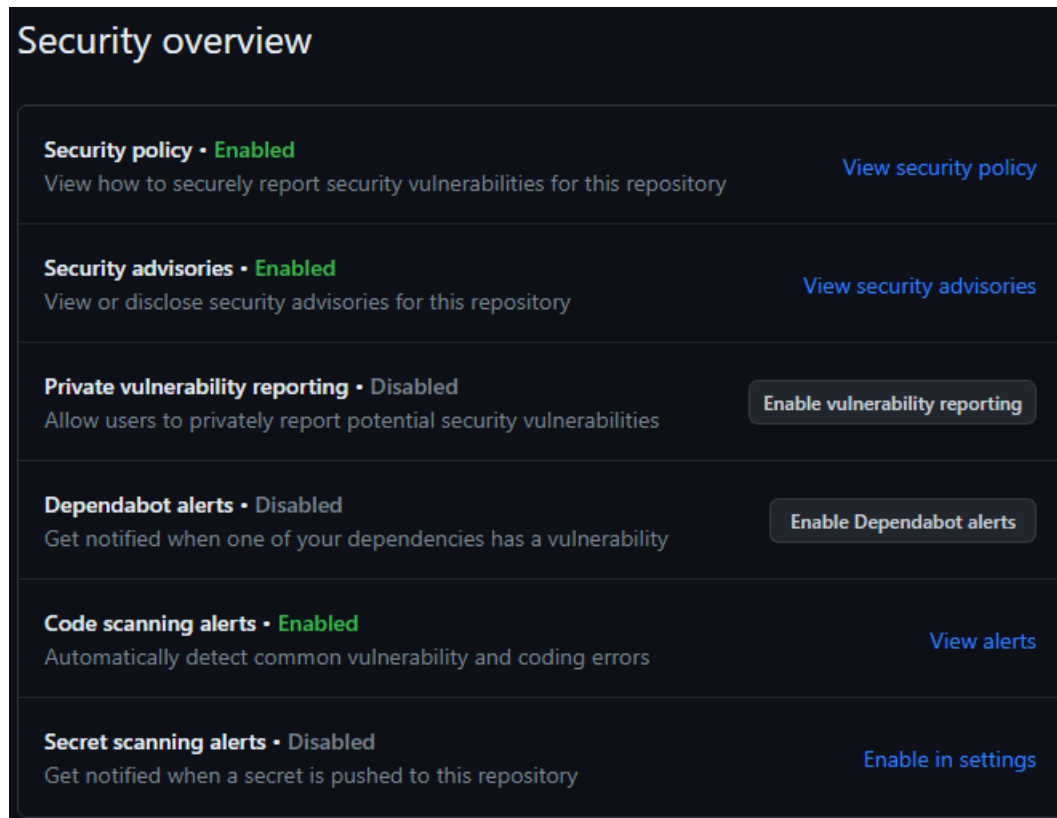


Figure 10. Security Policy Created and Code Scanning Alerts Enabled

5. Works Cited

- cmnatic. (2021, September 10). *Pentesting Fundamentals*. Retrieved from TryHackMe: <https://tryhackme.com/room/pentestingfundamentals>
- Home, C. (2023, February 22). *The Pros and Cons of Using Global Variables in JavaScript*. Retrieved from Medium - Plain English: <https://javascript.plainenglish.io/the-pros-and-cons-of-using-global-variables-in-javascript-be5c25d2ba57>
- MIT. (2022, December 15). *Open Licenses: Creative Commons and other options for sharing your work*. Retrieved from University of Pittsburgh: Library System: <https://pitt.libguides.com/openlicensing/MIT#:~:text=Users%20of%20software%20using%20an,and%20the%20X%20Windows%20System.>
- NICCS. (2023, March 16). *Explore Terms: A Glossary of Common Cybersecurity Words and Phrases*. Retrieved from NICCS: National Initiative For Cybersecurity Careers and Studies: <https://niccs.cisa.gov/cybersecurity-career-resources/vocabulary#R>
- Nicole Gonzalez, D. (2021). The State of Practice for Security Unit Testing: Towards Data Driven Strategies to Shift Security into Developer's Automated Testing Workflows. *RIT Scholar Works*, 118.

Security Assessment – Calculator

Schmittling, R., & Munns, A. (2010, January 1). *Performing a Security Risk Assessment*. Retrieved from ISACA: <https://www.isaca.org/resources/isaca-journal/past-issues/2010/performing-a-security-risk-assessment>

VMware. (n.d.). *Endpoint Security*. Retrieved from VMware.