1. Give the titles and URLs of the three Tom Scott security-related videos you watched (to completion), together with a sentence or two on the purpose or lesson of each video.
2. [The Two Generals’ Problem](https://www.youtube.com/watch?v=IP-rGJKSZ3s)  
   Tom Scott explains what the Two General’s Problem is, and gives a real world problem, involving a food delivery application, as an example. The solution he suggested would be using an “idempotency token”, a unique identifying key that can be used to validate that a single request is from a client is processed, rather than multiple identical requests being processed.
3. [The Moonpig Bug: How 3,000,000 Customers' Details Were Exposed](https://www.youtube.com/watch?v=CgJudU_jlZ8)  
   Tom Scott explains a bug Paul Price found in Moonpig’s website that can be used to expose any customer’s private details. The lesson is “code like you are being attacked because you will be. Never trust user input. Assume that anyone sending anything to your site is malicious until proven otherwise.”
4. [How NOT to Store Passwords! - Computerphile](https://www.youtube.com/watch?v=8ZtInClXe1Q)  
   Tom Scott goes over multiple bad approaches to storing passwords that websites still use such as direct comparison, encrypting, and hashing. He says the approach “nowadays”, in 2013, was “hashing and salting”. The best approach is not to store passwords, but rather use OAuth, or using “Log in with Google/Facebook/Apple”.
5. Research the concept of *Security Through Obscurity*. Write up a couple paragraphs describing what this phrase refers to, give some examples, and describe why it is (generally) a bad thing.

Security Through Obscurity (STO) relies on the idea that a system can remain secure if the vulnerabilities are secret or hidden. STO assumes that because attackers can’t exploit a system’s weaknesses if they don’t know where the weaknesses are, then the system is secure. The logic behind STO is flawed because any system that has a weakness cannot be secure, regardless of whether those weaknesses are known or unknown.

Hiding a key to a locked door under the doormat is an example of STO. This is obviously a very unsecure since anyone can just pick up the key and open the door. Another example of STO is deploying decoy cars around the asset you are trying to protect, with only key players knowing which car the asset is in. This is also unsecure since hostiles could attack each car one by one until they found the car with the asset. Relying solely on STO is a bad idea, but using other security measures along with STO can be a good idea.

1. Give (software) examples of (a) a failure of confidentiality, (b) a failure of integrity, and (c) a failure of availability.
   * 1. a failure of confidentiality

The Moonpig bug is an example of a failure of confidentiality since it allowed unauthorized access to supposedly restricted data.

* + 1. a failure of integrity

The SolarWinds Orion Attack ([link](https://www.simplilearn.com/tutorials/cryptography-tutorial/all-about-solarwinds-attack#how_did_the_solarwinds_attack_happen)). Also, a Google Nest Thermostat software update that caused the battery to drain and led to the temperature of users home to drop. In both cases, there was a system that was working and after a software update, the systems failed.

* + 1. a failure of availability

HSBC’s IT outage in January 2016 lead to millions of bank customers to be unable to access their accounts online.

1. What is the difference between authentication and authorization? Give an example.  
    Authentication is process of verifying that someone or something is who they say they are. An example of authentication would be using a photo passport at an airport check-in desk to prove that you are the passenger that booked a seat on the flight.

Authorization is the process of determining a user’s or service's level of access. An example of authorization would be an FBI agent accessing the FBI database to read a confidential case report. This FBI agent is authorized to access the database and read the report, but citizens outside law enforcement, for instance a student, are not authorized to access the FBI database.

1. Select 3 guidelines each from the SEI CERT Guidelines in this assignment’s reading list (for C, C++, and Java). For each, give their name, their number in the CERT numbering scheme, a description of the standard *in your own words* and an example of your very own of code that is compliant with the selected guideline. (You may optionally include a non-compliant piece of code too, but please mark it as non-compliant.) (Be careful to select 9 guidelines that are markedly different from each other.)  
   C:
2. [[FLP30-C](https://wiki.sei.cmu.edu/confluence/display/c/FLP30-C.+Do+not+use+floating-point+variables+as+loop+counters)] **Do not use floating-point variables as loop counters**  
    Since floating-point numbers are limited in the amount of usable digits, FP numbers cannot represent all significant digits of large values. By limiting the number to the significand with an exponent, FP numbers are not as precise as integers to be loop counters.
3. [[ARR32-C](https://wiki.sei.cmu.edu/confluence/display/c/ARR32-C.+Ensure+size+arguments+for+variable+length+arrays+are+in+a+valid+range)] **Ensure size arguments for variable length arrays are in a valid range**  
    When creating arrays with a variable size, check that the value of the variable is in valid range, such as greater than 0 and less than an appropriate maximum size.
4. [[STR32-C](https://wiki.sei.cmu.edu/confluence/display/c/STR32-C.+Do+not+pass+a+non-null-terminated+character+sequence+to+a+library+function+that+expects+a+string)] **Do not pass a non-null-terminated character sequence to a library function that expects a string**  
    When working with strings in C, before passing the string to a function, make sure that the last element of the character sequence is the terminating character ‘\0’.

[GitHub Link](https://github.com/Ulq15/CMSI662_Secure_Software_Development/blob/main/HW2/problem5_C.c) to code compliant with FLP30-C, ARR32-C, and STR32-C:

#include <stdio.h>

#include <stdlib.h>

#include <ctype.h>

#include <string.h>

*/\** *Encrypts a string by 1st adding 100 to each characters ASCII value*

*then multiplying that by 4. This way each character's ASCII value is*

*comprised of 3 digits. Then add each triple of digits into a new string.*

*To decode, divide each group of 3 digits by 4 then subtract by 100 \*/*

char\* encrypt(char \*s) {

    size\_t size = strlen(s); *// compliant with STR32-C*

    if (size <=1 || size >= 100){

        return -1;

    } *// compliant with ARR32-C*

    char \*ptr = (char \*)calloc((size+1)\*3, sizeof(char));

    for (size\_t i = 0; i <= size; i++) { *// compliant with FLP30-C*

        int encodedChar = (s[i] + 100)\*4;

        sprintf(ptr+(i\*3), "%ld", encodedChar);

    }

    return ptr;

}

int main() {

    char raw\_message[] = "Encrypt this message.";

    printf("1. The raw message is:\t\t%s\n", raw\_message);

    char\* array = encrypt(raw\_message);

    printf("2. The encrypted message is:\t%s\n", array);

    return 0;

}

C++:

1. [[MEM51-CPP](https://wiki.sei.cmu.edu/confluence/display/cplusplus/MEM51-CPP.+Properly+deallocate+dynamically+allocated+resources)] **Properly deallocate dynamically allocated resources**  
   When you use the new operator, then you need to use the deconstructor rather than the delete operator, because the delete operator might not be able to free memory that was used but not returned by new(). Do not use delete on uninitialized pointers, instead declare the pointers as nullptr, then when you are done with them, use delete.
2. [[STR50-CPP](https://wiki.sei.cmu.edu/confluence/display/cplusplus/STR50-CPP.+Guarantee+that+storage+for+strings+has+sufficient+space+for+character+data+and+the+null+terminator)] **Guarantee that storage for strings has sufficient space for character data and the null terminator**

If you want to make a string, instead of using an array of characters like in C or using a buffer to build a string character by character, use the string.h library, this way your strings are always null terminated.

1. [[CTR55-CPP](https://wiki.sei.cmu.edu/confluence/display/cplusplus/CTR55-CPP.+Do+not+use+an+additive+operator+on+an+iterator+if+the+result+would+overflow)] **Do not use an additive operator on an iterator if the result would overflow**

Java:

1. [[FIO01-J](https://wiki.sei.cmu.edu/confluence/display/java/FIO01-J.+Create+files+with+appropriate+access+permissions)] **Create files with appropriate access permissions**   
    When creating and modifying files, you must set the appropriate access permissions so that the file can be read, modified, or deleted by only authorized users. The compliant code below is given as a example from [this site](https://wiki.sei.cmu.edu/confluence/display/java/FIO01-J.+Create+files+with+appropriate+access+permissions).

public class Problem5\_Java{

  public static void fileExample(){

    Path file = new File("somefile.txt").toPath();

*// Throw exception rather than overwrite existing file*

    Set<OpenOption> options = new HashSet<OpenOption>();

    options.add(StandardOpenOption.CREATE\_NEW);

    options.add(StandardOpenOption.APPEND);

*// File permissions should be such that only user may read/write file*

    Set<PosixFilePermission> perms = PosixFilePermissions.fromString("rw-------");

    FileAttribute<Set<PosixFilePermission>> attr = PosixFilePermissions.asFileAttribute(perms);

    try (SeekableByteChannel sbc = Files.newByteChannel(file, options, attr)) {

      sbc.write(ByteBuffer.wrap("some data\n".getBytes()));

    };

  }

}

1. [[MET09-J](https://wiki.sei.cmu.edu/confluence/display/java/MET09-J.+Classes+that+define+an+equals%28%29+method+must+also+define+a+hashCode%28%29+method)] **Classes that define an equals() method must also define a hashCode() method**  
    When designing an objects class, if you want to override the equals() method, so that you can compare objects of your class, then you have to also override the hashCode() method. Java’s Object class, which all other Java classes inherit from, uses the hashCode() method when equals() is called. Object’s equals() method compares the values returned by each object’s hashCode(), and if the values are equal, then the equals() method will return true, otherwise false.

Example compliant code can be found commented out in my submission for [HW1](https://github.com/Ulq15/CMSI662_Secure_Software_Development/blob/main/HW1/src/Main.java) from line 420 to 473.

1. [[OBJ10-J](https://wiki.sei.cmu.edu/confluence/display/java/OBJ10-J.+Do+not+use+public+static+nonfinal+fields)] **Do not use public static nonfinal fields**  
    Class fields that are declared public and static can be accessed and modified by other objects. To prevent any changes to the class’s members from outside the class, the members should be declared as final, so that once they have been set, they cannot be changed.

Example of uncompliant code can be found [here](https://github.com/Ulq15/ZorkJava/blob/master/Game/HelpStage.java) on line 18. This was a project I worked on a while ago to make a Zork port using Java.