**Code/Cyber Continuity Guide**

Hello, My Name, is c/Eckhoff I have lead code quest since 2022-2024 and cyber since 2023-2024 I also Interned at Lockheed through their program through code quest through this, I made good decisions to make Code and Cyber Better, I have also made mistakes which you should avoid when leading.

Over this guide I will go over the following to help you learn what you need to know to make your code and cyber experience go as best as possible.

* General Information about the competitions
* How to gain interest for the competition
* When to start preparing for the competition
  + Start Early
* Best ways for getting permission forms in
* In person VS. Online differences
* What needs to be learned for the competition
* How to get people their and back

Through this you should have everything you need to know to be able to lead the competition as well as possible and have the best cyber and code ever

**Glossary**

1. Code Cyber Continuity Guide
2. Basic Information about the competition
3. Code and Cyber Beginning of the year set up
4. Code Quest What Needs To Be Taught
5. Lessons Part one introduction
6. Lesson 1.0- intro to code
7. Lesson 1.1- data types and programming structure
8. Lesson 1.2- Variables in python
9. Lesson 1.3- Variables interactions
10. Lesson 1.4- Inputs & Flow Control
11. Intro to Lessons Part 2
12. Lesson 2.1- First Practice Problem
13. Lesson 2.2- More Practice Problems
14. Lesson 2.3- Loops in python
15. Lesson 2.4 - List in python
16. Intro to part 3
17. Lesson 3.1- Object Oriented Programming
18. Lesson 3.2 - Approach to questions in code quest
19. Lesson 3.3 - Debuging code
20. Cyber Quest Intro
21. Basic Computer Use
22. Cyber Quest- Basic Linux Commands
23. Cyber Quest- Cyber Chef
24. Cyber Quest – Types of Attacks
25. Cyber Quest – Web Standards and infrastructure
26. Cyber Quest – Cyber Security Tools
27. Team Building For Code Quest Part 1
28. Team Building For Code Quest Part 2
29. Getting to the competition
30. Resources
31. Reflection on the competition

**Basic Information About the competition**

**Basic Information-**

Both events are hosted by Lockheed Martin to get more people interested in cybersecurity and programming.

**Code QUEST-**

Code Quest is the **FIRST\*** of the competitions which is focused on programming and problem solving

\*Make sure to check the times because it isn’t always the first

**INFO-**

* Competition last 2.5 hours long
* 20-30 questions given
* 2-3 people per team
* 2-4 max teams

**HOPEFULLY YOU HAVE PARTICIPATED AND CODE QUEST BEFORE AND THIS IS JUST A REFRESHER**

**Cyber QUEST-**

Cyber Quest is focused on cyber security and attacking and defense in the cyber security field

**INFO-**

* Competition last 2 hours
* Capture the flag format
* 2-3 People Per Team

**Beginning of the year set up**

**When To Start:**

For both code and cyber both should be started early it takes about 3 months to train someone how to program for code quest, but it is recommended that you get close to 6 months of training before the competition to best train for code quest and a similar time for cyber-Quest.

Both code and cyber should be started as soon as possible after the school year starts to maximize how much time you must teach the content if you begin in September you should have at minimum 4 months to teach to content which should be plenty of time to teach the content

**How To Advertise:**

*What in the world is a Code Quest?* You need to be able to advertise to the program it takes time to be able to get 100s interested but eventually they will be willing to do it

* Morning Announcements
  + Explain What it is
* Advertise in you flights
* Look for people in the program who may be interested
* Advertise to people out of the program that their is stem opportunities in JROTC
  + Their is not many of these opportunities in plant as a whole
* Multiple interest meetings if necessary

**THE MOST IMPORTANT PART OF ADVERTISING THIS EVENT IS TO EXPLAIN WHAT IF IT IS NOBODY IS GOING TO KNOW WHAT CODE OR CYBER ARE JUST ENTERING INTO THE PROGRAM MOST PEOPLE IN THE PROGRAM DONT KNOW WHAT IT IS**

*Make sure that* ***even*** *if you only get a couple of people to sign up at first you still start the even early you do not want to be in the position of waiting to get more people only to not have time to teach the content to those who did sign up early*

**Setting Goals**

At the beginning of my years I set goals for myself and what I wanted to accomplish, and it is a good idea to set goals for yourself as well the goals should be achievable, meaningful, and attainable but don’t be afraid to aim high if you don’t meet your goals exactly then it can be a good idea to reflect upon an improve the setup for the next years competition

HERES AN EXAMPLE OF MY GOALS I SET THIS YEAR

An \* by the goal means it was a goal that I met and if they do not have an \* it means I did not meet the goal

* Have our teams do better than the last year \*
* Start preparation for code and cyber quest earlier
* Create a more exact curriculum for code and cyber \*
* Increase participation \*

Your goals can be as simple as these or be a more complex goal to reach

Here are examples of goals that I think are good to set for next year:

* Retain participation from 80% of the previous years participants (barring people who graduated)
* Improve or keep the same score that the teams score in the previous year
* Start preparation for code and cyber earlier

These do not need to be your exact goals however this provides you with a starting point to create your own goals to keep in mind

**What Needs To Be Taught For Code Quest**

Code Quest Has a large load of content you need to teach before people will feel confident about competing **HOWEVER** It is important to note that as long as you start early enough, **SEPTEMBER** Code Quest Should be a marathon not a sprint meaning you should have time to go over the content very well before the competition

**IN CASE YOUR READING THIS IN NOVEMBER I HAVE LABELED THE FOLLOWING WITH RED FOR VERY IMPORTANT BLUE FOR MEDIUM IMPORTANT BUT IF NECCESSARY CAN BE REMOVED AND GREEN FOR NOT IMPORTANT UNLESS YOU CAN GET TO IT JUST IN CASE YOU CANNOT COVER A TOPIC**

* DATA TYPES
* BASIC PROGRAMMING STRUCTURE
* INPUTS
* VARIABLES
* CASTING
* MATH
* MANIPULATING VARIABLES !STRINGS!
* IF STATEMENTS & ELSE & IF ELSE STATEMENTS
* METHODS
* LOOPS
* INDEXS
* ARRAYS
* .format !!!(FOR PYTHON)!!!
* FUNCTIONS, CLASSES, AND OBJECTS
* DICTIONARIES
* CODING STYLES
* MULTILEVEL PROGRAMS
* API CALLS
* OPENAI (MIGHT BE A FUN LAB TO DO WITH THEM IF YOU ARE ABLE TO FINISH TEACHING EVERYTHING AND THEY ARE DOING VERY GOOD)
* EDITING FILES
* CONTROLLING MOUSE MOVEMENTS WITH PYTHON

NOTE !!!THE THINGS THAT CAN BE TAUGHT WHICH ARE GREEN ARE OUTSIDE THE SCOPE OF WHAT THEY WILL NEED TO DO FOR CODE QUEST!!!

**Part 1 of lessons**

Note:

I would like to start off by saying that if someone was to give me a bunch of lessons when I started teaching code quest in my junior year and told me this is what I needed to teach I would be very annoyed and probably not used any of their lessons that they gave me. I also want to say that I am not perfect at teaching code if I was we would be first, second, and third place at code quest. Despite this I have gone through and written out each lesson and what I would teach as if I was planning on teaching code for another year however by no means are you required to even look at the lessons these lessons are meant to help you, you are going into your senior year and are probably going to be super busy and for me there were times that I wish I had a lesson already written out for me on the day of code quest meeting however I don’t expect you to use the lessons entirely and rely on them or to completely ignore them I expect you will probably use the lessons as a reference guide for you and make adjustments to them based on what you think will work best. Due to this I have written the lessons in a very loose way as to not force you into any one path of teaching these topics.

**Keep this in mind-**

* These lessons aren’t perfect don’t be afraid to edit them to best fit the next group of students you need to teach
* Many of these students have never programmed before and thus some things that might seem obvious to you will seem alien to others
* Be open to innovating on the content, bring games or activities make it fun for them to learn
* Coding should be fun the competition is not the most important thing make sure to they are having fun learning to code

**LESSON #1**

**THINGS TO KEEP IN MIND:**

* These Lessons are just suggestions feel free to be CREATIVE as you lead code quest or use one every once in a while if you have forgotten to prepare a lesson for the day the more YOU that you put into the lessons the better you will be
* Some people who will join code quest have never programmed before are going to have a very Difficult time understanding how to program and are going to take some time to be able to understand the syntax **!!!METAPHORS AND ALTERNATIVE WAYS OF UNDERSTANDING PROGRAMMING ARE YOUR BEST FRIEND!!!**
* Starting with a simple topic is important make sure that you are putting time in trying to understand how this will look to people who are new to programming
* For code quest and coding beginners using **PYTHON** is what we have used but if someone is new and are willing to put in the work let them go try other languages
* These Lessons were written with python in mind meaning if you plan on teaching another language then you must translate the lessons **HOWEVER IT IS RECOMENDED** That you use python as I have found it is the easiest for newer programmers to understands

**Lesson 1 Topics-**

* Basic Programming Sentence Structure
* Data Types
* Casting (OPTIONAL or if you need to go fast because you started late Casting is important and still must be taught at some point)
* Printing

**Lesson 1-**

**OVERVIEW-**

Go over basic python sentence structure and How Data Types Are Used in python such as adding together data types and OPTIONALLY how you add together two unrelated data types using Casting. How to print out those object

**Examples To Go Over-**

STRING + STRING

INT + INT

FLOAT + FLOAT

INT + FLOAT

**LESSON #2**

**Things To Keep In Mind:**

* Make Sure They Understand the content if you started in **SEPTEMBER** you should have plenty of time to make sure that the are understanding and able to apply the knowledge from the week before
  + If they are not able to apply the knowledge they learned from the previous week consider going back over the content in order to make sure they understand the lessons
* The earlier the lessons the less content you are going over the the content is much more important early on **YOU WILL USE TOPICS LIKE CASTING AND DATA TYPES IN NEARLY EVERY PROBLEM SO ITS VERY IMPORTANT THAT THEY UNDERSTAND IT**

**Lesson 2 Topics-**

* More Program Sentence Structure
* Variables

**!!!MAKE SURE THAT THEY UNDERSTAND THESE FIRST TWO LESSONS VERY WELL BEFORE MOVING TO LESSON #3 THEY MUST UNDERSTAND HOW THESE WORK IN ORDER TO DO THE BEST THEY CAN!!!**

**Lesson 2-**

**OVERVIEW-**

Go over how variables Function in Python and how the variables interact with Data types and how Casting Variables Works in Python

**Examples To Go Over-**

Basic Variable Format - [Variable Name] = [Data]

Casting Variables = str([INTEGER\_VARIABLE])

Comparing = [VARIABLE\_NAME] = “1” To [VARIABLE\_NAME] = 1

**Lesson #3**

**Things to keep in Mind-**

* The previous two Recommended lessons are very **IMPORTANT TO UNDERSTANDING OF PROGRAMMING** make sure that they understand it well before moving on
* From my experience last year **THE CADETS HAVE A VERY HARD TIME UNDERSTANDING VARIABLES IN PYTHON THEY MAY NEED MORE HELP THAN YOU THINK THEY DO**

**Lesson 3 Topics-**

* Adding variables between each other
* How data type interacts with each other
* Casting here

**Lesson 3 Overview-**

* For this topic they must be able to understand how math between variables works and how this can be used
* Make sure that they understand how variables of two different data types will interact and how adding them incorrectly can cause an error **MAKE SURE TO TEACH CASTING HERE**
* Make sure they understand how to identify different data type and the **ERROR** that can occur when you add two data types incorrectly

**Lesson #4**

**Things to keep in mind**

* Inputs are difficult to find an exact spot but I found that this lesson is the best place to teach them because you need them to properly demonstrate flow control
* **After this lesson you can start working on problems and walking through them however I have included several addititional lessons which are meant to help with certain more difficult topic**

**Lesson 4 Topics**

* Inputs
* Boolean expressions
* If, else, elif statements;

**Lesson 4 overview**

* Go over how inputs work and run them through how to write and type your input into the console
* Go over Boolean Expressions
* Go over if, else, and elif statement and how they are defined and what order the need to define them in to work properly

**Intro to lessons part 2**

At this point you have gone over the basics of programming this is enough to go over several of the code quest academy practice problems and start seeing which parts they are struggling on

**Important notes-**

* Each practice problems allow a section to contextualize the tools which they have learned up to this point
  + Teaching them smaller topics through doing a practice problem which uses those tool can be more interesting and helpful to teaching a method or tool they will use less than a more broad topic like variables or flow control
  + Some students may have a hard time understanding through powerpoints some people will learn in different ways because of this I would recommend trying to teach through practice problems is someone is having a particular trouble understanding certain topics
* I would recommend still using some power points for broader topics or as a way to lead into a practice problem which will use the tool you are teaching they will give an area to teach before letting them go into a problem and learn it themselfs
  + **This method can allow them to go through a problem alone at the meeting without you having to go through step by step like I have had to do in this year and last year for years past and lead to an overall stronger programmer for code quest**

**Lesson 1 part 2**

**Things to keep in mind-**

* This is the first lesson I will be implementing a section after the power point to include a practice problem for the students to work on
* You should look at and solve a practice problem before you try and lecture about it in a meeting

**Lesson 1 Part 2 Topics-**

* Split method in python
* Indexing in python

**Lesson 1 Part 2 Overview-**

* In this lesson you are trying to teach so they are able to solve the **SORT OF SELF DRIVING PROBLEM**
* This was selected for the first problem because it was the easiest and uses things that they will use on nearly every problem

**Lesson 1 Part 2 Practice Problem Walk Through-**

#don't include top portion

spl = input().split(":") #dont include split on the same line as input

mps = float(spl[0])

ttw = float(spl[1])

if(mps != 0):

if ttw/mps <= 1:

print("SWERVE")

elif ttw/mps <= 5:

print("BRAKE")

else:

print("SAFE")

else:

print("SAFE")

* Make sure to simplify when lecturing on it
* Don't immediately correct them or give them the right answer let them divide by zero and have to see why what they did was wrong

**Lesson 2 Part 2**

**Things to keep in mind-**

* It is important to make adjustments to which practice problems you are teaching based on what they are having trouble on for this lesson I selected another which they should have an easy time solving
* This is the first one that I did not include a power point at all because they already should understand everything that is gone over in the question

**Lesson 2 part 2 topics-**

* Math with casting
* More spliting
* Indexing

**Lesson 2 Part 2 Overview-**

* For this lesson I chose the **Addiply** problem it is very similar to the sort of self driving problem and thus it should be a quicker meeting
* If you want the meeting to go longer and give more practice you could also do the **Animal Farm Problem** as it is similar to the **Addiply** problem but it is not neccessary

**Lesson 2 Part 2 Practice Problem Walk through-**

**ADDIPLY-**

spl = input().split(" ")

First = int(spl[0])

Second = int(spl[1])

print(str(First + Second) + " " + str(First \* Second))

**ANIMAL FARM-**

#please don't solve it like this in front of them it will confuse them

spl = input().split(" ")

Turkeys = int(spl[0]) \* 2

Goats = ((int(spl[1]) + int(spl[2])) \* 4) + Turkeys

print(Goats)

**Lesson 3 Part 2**

**Lesson 3 Part 2 Practice problem Walk through-**

**Things to keep in mind-**

* There is probably a better practice problem this practice problem was chosen because it allows you to demonstrate a single loop and a double
* There is a power point on loops for this section

**Lesson 3 Part 2 Topics-**

* For loops
* While loops
* Nested loops

**Single loop-**

inp = int(input())

count = 0

for i in range(inp \* inp):

count += 1

if count == inp:

print("#\n")

count = 0

else:

print("# ", end='')

**Double loop-**

inp = int(input())

count = 0

for i in range(inp):

for j in range(inp):

count += 1

if count == inp:

print("#\n")

count = 0

else:

print("# ", end='')

**Lesson 4 Part 2**

**Things to keep in mind**

* List are going to be one of the more difficult topics you will cover but they are not as necessary as they would be in languages other than python as most of the time you can just use a string instead of a list because strings act very similar to lists in python

**Lesson 4 part 2 topics-**

* Lists
* List methods
* NOT IN POWER POINT BUT YOU COULD COVER MULTIDEMTIONAL LISTS HERE

**Practice problem**

Anagram checker:

spl = input().split("|")

#converting to list

word1 = list(spl[0])

word2 = list(spl[1])

#sorting list into alphabetical order

word1.sort()

word2.sort()

#compare the letters in each list

if(word1 == word2):

print("ANAGRAM")

else:

print("NOT AN ANAGRAM")

**Intro To Part 3-**

Before going into part three there are some more optional subjects that you should consider covering as they could be helpful but overall I would say they aren't as necessary these topics include:

* Dictionaries
* .format
* Creating your own methods **SPECIFICALLY A ROUNDING METHOD**
* **USE NumPy For Rounding-** pip install numpy

**Part 3-**

Part 3 mostly pertains to how to write code effectively and how to think like a programmer these are more difficult topics to teach as they will probably either get the topic or won’t at this point they should know most of the code they will need to do well in the competition and now you are just trying to improve the efficiency that they program at and their problem solving ability

**The topics I go over in this section are:**

* Object Orented Programming
* How to approach the questions in code quest
* How to debug your code when things haven’t gone to plan

**!!!REMEBER YOU CAN STILL BE GOING OVER UNRELATED PRACTICE PROBLEMS AFTER A TOPIC OF A MEETING**

**Lesson 1 Part 3**

**Things to keep in mind-**

* This lesson covers object oriented programming however unless they plan on doing Hard level questions this should not even come into play but for the sake of being complete and because it is the correct most efficient way to do things I have put it here **HOWEVER THIS TOPIC CAN BE EASILY SKIPPED OVER AND THEY CAN DO FINE**
* Due to the nature of the topic I have not included a practice problem however you could go over a practice problem an point out where different methods are used
* I began to create a power point for this topic but stopped because how you teach to topic will depend greatly on what you have taught in the past and you **MUST HAVE ALREADY TAUGHT ABOUT METHOD CREATION BEFORE TEACHING THIS TOPIC**

**Lesson 1 Part 3 Topics**

* Object oriented programming
* Dry methodology (Don’t Repeat Yourself)

**Lesson 1 Part 3 Overview-**

* This lesson goes over more advanced formating in python this is things that you will have to keep in mind if you are trying to prepare your teams to be able to complete medium and hard questions during the competition

**Lesson 2 Part 3**

**Lesson Overview-**

* This lesson covers how to approach questions in code quest this can be helpful as a lesson you cover just before the competition because it will allow for the cadets to keep in mind how to go about starting programming

**Things to keep in mind-**

* Parts of this lesson should be covered as you go along however this consolidates the lesson into one place

**Lesson Topics-**

* Approach to code quest questions

**Lesson 3 Part 3**

**Overview-**

* This lesson covers the common types of errors in python, examples of code which contain the errors, and how to debug the code with these errors

**Things to keep in mind**

* In this lesson I went over the types of errors in python and examples of when you might encounter them
* This lesson is short so you should have plenty of time to do a practice problem afterward

**Topics Covered**

* Error Types in python
* Debugging python programs

**Cyber Quest Intro**

Cyber Quest as you know already is far more knowledge based than code. While code you must know the basic coding tools cyber you must know areas in all parts of cyber security and are asked to answer questions about these areas. Because of this how you teach cyber quest will be significantly different than how you teach code and will require far less practice in these areas but instead far more teach specifics in these areas

Topics To Teach in Cyber Quest:

* Basic Computer Use
* Commands (Linux/Windows)
* Cyber Chef Use
* Encryption
* Decryption
* Steganography
* Web page security and infrastructure
* How computers work
* Finding information about computers/IP
* Mapping networks
* Cybersecurity tools
* Cyber security standards
* AI security standandard

***!!!IMPORTANT!!! DUE TO LOCKHEEDS NEW IMPLEMENTATION OF CYBER ACADEMY I HAVE NOT GONE AS INDEPTH ON MANY OF THE TOPICS DUE TO A NEW CURRICULUM NEEDING TO BE ESTABLISHED ONCE CYBER QUEST ACADEMY GOES LIVE!!!***

**Basic Computer Use**

Overview- this should be the first thing you teach many of the people you will be teaching are going to be at different points in knowledge and due to this you will need to catch some people up to the point they need to be this first lesson can also be a good place to get an idea of where each person is in their knowledge and what areas they are going to need to be helped more in

**TOPICS-**

* Basic computer terminology
* How to open the terminal
* Typing commands in the terminal
* how computers work (IP, \*sending information over the internet)
* \*Binary
* How to install software and how to do it safely

\*these topics probably can be taught later and will need to be retaught later

**Commands**

**Overview-**

This lesson should be used as an introduction to Linux and the command line interfaces for most people

**Topics**

* CLI vs GUI
* Basic Command
* Flags for commands in Linux
* How the terminal works

|  |  |
| --- | --- |
| **Command** | **Use** |
| man [COMMAND\_NAME] | Stands for **MANUAL**. Gives information and flags about specified command |
| cd [DIRECTORY\_PATH] | Stands for **CHANGE** directory changes the current directory of the command line |
| mkdir [NEW\_DIRECTORY\_NAME] | Stands for **MAKE** directory creates a directory inside of the current directory or in a specified path |
| rmdir [DIRECTORY\_PATH] | Stands for **REMOVE** directory removed the specified directory |
| mv [FILE\_NAME] [DESTINATION PATH] | Stands for **MOVE** |
| ls [OPTIONAL\_DIRECTORY\_NAME] | Stands for **LIST.** List out the directories and files which exist in the current directory or in a specified directory |
| Touch [FILE\_NAME] | **CREATES** a file of a specified type |
| Cat [FILE\_NAME] | **PRINTS** the text in a file without opening the file |
| nano [FILE\_NAME] | Opens a **TEXT EDITOR** which allows users to edit text files from inside the CLI |
| grep [“STRING”] [FILE\_NAME] | **FINDS** the location of a specific string in a text document |
| netstat | List information about the current **NETWORK** |
| nslookup [DOMAIN\_NAME] | Looks up information about a specified **DOMAIN NAME** |
| ping [IP\_ADDRESS | Checks if a specified **IP ADDRESS** is reachable |
| sudo [COMMAND] | Stands for **SUPER USER DO** runs a command as a super user  **USE IF YOU DO NOT HAVE PERMISSION** |
| hostname | Shows the **IP ADDRESS** of the system |
| chmod | Stands for **CHANGE MODE** allows users to change permissions added to a file  Chmod 777: gives everyone access to a file reading writing and executing |
| Ipconfig | Gives information about the current **TCP/IP CONFIGUATION VALUES** |
| ifconfig | Displays the curretn **NETWORK PROTOCOL PERAMETERS** |
| Traceroute [domain/IP] | Displays the route which packets take to get to a specific IP or domain name |
| Whois [domain/IP] | Displays information about a specifc domain or IP address (**recommending using IP instead of domain names use nslookup for domains**) |

**CYBER CHEF**

**Overview-** to do the best in the competition they should be able to effectively use basic cyber chef prompts for decryption and steganography

**Topics-**

* Encryption Types (AES-256, base 10, Hex)
* how to use Cyber Chef
* How to decrypt within cyber chef

|  |  |
| --- | --- |
| **ENCRYPTION NAME** | EXAMPLE: HELLO WORLD |
| Binary | 01101000 01100101 01101100 01101100 01101111 00100000 01110111 01101111 01110010 01101100 01100100 |
| Hex Code | 68 65 6c 6c 6f 20 77 6f 72 6c 64 |
| Base 64 | aGVsbG8gd29ybGQ= |
| Unicode | hÌ¶Ì²eÌ¶Ì²lÌ¶Ì²lÌ¶Ì²oÌ¶Ì² Ì¶Ì²wÌ¶Ì²oÌ¶Ì²rÌ¶Ì²lÌ¶Ì²dÌ¶Ì² |
| Octal | 150 145 154 154 157 40 167 157 162 154 144 |
| Hex Dump  ***This is the output in wireshark*** | 00000000 68 65 6c 6c 6f 20 77 6f 72 6c 64 |hello world| |
| AES (ADVANCE ENCRYPTION STANDARD) | 8c49d56931c48f163b8e25d42c94c3a3 |

* How cyber chef can be used in stegonography and stegonographic detection
* Make sure to check last years stegonography questions and teach to the questions

**Types of attacks**

**Overview-** many of the knowledge based questions in cyber quest require the background knowledge of cyber security attacks so it is worth while to teach them

Topic

* Types of attacks
* How these attacks are defended against

|  |  |
| --- | --- |
| Attack Name | What it does |
| Man in the middle attack | Allows a bad actor to look at conversations over the internet between two users |
| Torjan Horse attack | Malicious software is hidden in what appears to be reputable sources (email, download, message) |
| Fishing, spear fishing, whale fishing, smishing, vishing , etc | Sending a random or targeted message in the hopes of getting someone to click on it and install malicious software |
| keylogger | a malicious software use to spy oin keystrokes and then report back to a bad actor |
| Spyware | Software used to spy on a user without their knowledge and extract information from them |
| Password attacks (rainbow table, brute force, dictionary) | Any attack that hopes as an end goal to extract a password |
| Dos DDOS | An attack meant to deny service from a piece of software or a website |
| Spoofing | Replacing a domain name with a malicious site in the hopes of fooling a user into entering information |

**Web Standards and infastructiure**

**Overview –** how to use the web and websites along with their domain names in order to exploit into places you are not meant to be or to find information

**Topics-**

* How to use inspect
* Reading html
* How to navigate the site source code
* How html, js, and css work together
* How to edit the sites url to make things happen

**Web security standards link-**

<https://owasp.org/www-project-application-security-verification-standard/>

**Cyber Security Tools-**

|  |  |
| --- | --- |
| **Tool Name** | **Tool Name** |
| NMap | Creates a map of a network with all connected ips and scanning open ports |
| wireshark/tcpdump | Captures packets passing through a network |
| burpsuite | Intercepting proxy servers in a network |
| 7-zip | Tool for looking into and extracting files from a file |
| Image Magik | Stegonography tool |

**COMMON WEB VULNERABILITIES**

<https://owasp.org/>

**Team Creation For Code Quest**

For Code Quest The Teams Especially Matter even more than cyber because not everyone has the same skill level to coding

**Two Approaches To Teams**

Approach 1- Make The Best Single Team Approach-

There are enough cadets which are good enough at programming every year to make at least 1 super strong team to be able to complete 5+ questions however this significantly decreases the quality of other teams

Pros and Cons of Approach 1-

Pros-

* Able to get very good score
* Work better and more efficiently with people who all do well
* People who want coding internships have better scores to apply
* Rewards those who work hard with the best possible team

Cons-

* Less people will want to do it again\*
* Teams do not have evenly spread scores
* Those who work hard but still are not as good as the best 3 will be put in worse teams

Approach 2- Mini Team With 1 Team Leader-

This Approach is what I have used there is no one correct solution however this approach allows for people to have easy access to people who can help them to do the best they can on the competition. The idea of the approach is to put one of the better programmers as the lead for a team and then that allows for people at each table to be able to have a resource to ask questions to

Pros and Cons of Approach 2

Pros-

* Allows weaker programmers to do better
* Helps people not to fail and never want to do code quest again

Cons-

* Punishes Stronger programmers
* Weaker Team Scores Overall
* Doesn't push weaker programmers to do better

Neither Approach is right or wrong there is times when it might be best to use one or the other or a mixture of both

**WHILE LEADING YOU ARE CONSTANTLY BALENCING HAVING THE MOST FUN, WITH DOING GOOD IN COMPETITION THESE APROACHES ATTACK THIS PROBLEM IN DIFFERENT WAYS BUT NEITHER SOLVE THE PROBLEMS COMPELETLY**

**ADDITIONAL NOTES-**

* Don’t Use Method One If Lockheed Doesn’t allow more than one computer during competition I have Tried that and the person who is the Strongest programmer will do all the programming during the competition so if that is the case try and put people who are at an equal level to them

**NOTES FROM AFTER COMPETITION-**

* I had used the first approach in cyber quest this year and the weaker team that I created had did better than the stronger team so it is up to you and you may be surprised with the results
* If you are in a **SITUATION** when **NOT EVERYONE MAY BEABLE TO ATTEND** what I have done is **PRIOROTIZE OLDER CADETS OR CADETS WHO MIGHT NOT BEABLE TO ATTEND THE FOLLOWING YEAR**

**EXAMPLE OF MY PROCESS OF CREATING TEAMS (NAMES REMOVED)**

V – Must Be in Team with K or E

M – Must Be in Team with K or E, Try And Not Put With People From Charlie, C?

C- Must Be In Team Of Three

S- Wants To Be On A Team With E

L- Must Be On A Team With K

K- Must Be On A Team With L

N- Inexperienced – no team of 2

E- Must Be On A Team With M Or V

Team 1-

K,

L,

V

Team 2-

E,

N,

M

Team 3-

S,

C – May not beable to go leaving S a team of one

**Resources**

**Past Power Points/Newly Created Power Points git hub**



**Code quest academy-**

<https://lmcodequestacademy.com/>

**Cyber Chef-**

<https://gchq.github.io/CyberChef/#recipe=Generate_QR_Code('PNG',5,4,'Medium')&input=aHR0cHM6Ly9naXRodWIuY29tL011cnBoeUVja2hvZmYvQ29kZVF1ZXN0LmdpdA>

**Code Chef-**

<https://www.codechef.com/>

**Cyber Quest Challenges and skills overview-**

<https://lockheedmartin.com/content/dam/lockheed-martin/eo/documents/CyberQuest/2019/LM-CYBERQUEST-Challenge-Overview_PIRA.pdf>

**Planning for getting to the competititon**

If you are planning on going into the event in person then you should have a plan put together for getting there and getting back with everyone

**Step 1-**

Find available drivers

**AS A RULE IF SOMEONE GOT THEIR DRIVERS LICENCES LESS THAN A YEAR AGO AND/OR HAVE NOT DRIVEN ON THE HIGH WAY THEY ARE NOT ALLOWED TO DRIVE OTHER PEOPLE TO THE COMPETITION**

**Step 2-**

Make a list of drivers and people who need to be driven to the competition and coordanate drivers based on their distance to the people who need to be driven

**Step 3-**

Make sure that you check in regularly with drivers on the way to the competition

**!!!it is roughly a 2 hour drive\* to the competition site!!!**

**Reflecting On The Competition**

One of the most important parts of leading code quest is reflecting and building upon the competition based on what worked and what didn’t work

**The most important thing is that score don’t go down and that we are trying to improve year over year**

**Things to look for when trying to improve:**

* Look for problems which the teams were unprepared for in the competition
  + My own example was AI security in cyber quest that I did not teach the teams about
* Look at what method helped teams learn best some ways of teaching topics work better than others
* Look for better methods/strategies to use in the competition