Unstable Radium Container

A CSC 102 Project

Team: A Gift From Chornobyl a.k.a. <Volodymyr, Joseph, Sebastian>

Oganesson CONTAINER DEFUSAL MANUAL

Version 1

Verification Code: <enter your code>

The Game

This project is based on the game **Keep Talking and Nobody Explodes**[1], a cooperative bomb defusing party game. As the game designers put it, "You're alone in a room with a bomb. Your friends, the 'Experts', have the manual needed to defuse it. But there's a catch: the Experts can't see the bomb, so everyone will need to talk it out - fast! Put your puzzle-solving and communication skills to the test as you and your friends race to defuse bombs quickly before time runs out!"

Their version is a software game. Our version takes the idea and realizes it as a physical device with buttons, switches, and more! Although our version can be played just like theirs, players can interact with both the bomb and this document at the same time (i.e., players can both defuse the bomb and serve as the "Experts", using this document to help disarm the phases).

The backend of our version of the game is a Raspberry Pi[2] computer that combines a typical computer with the ability to interact with the outside world through sensors. The underlying software is written in Python[3] and is the result of a final group-based project in CSC 102 (The Science of Computing II) in the Computer Science Program at the University of Tampa.

[1] https://keeptalkinggame.com/

[2] https://www.raspberrypi.com/

[3] https://www.python.org/

Defusing Bombs

The bomb will "explode" when its countdown reaches 0:00 or when too many strikes have occurred. You defuse the bomb by disarming all of its "phases" before the countdown expires.

Phases

The bomb has four phases, each of which must be disarmed to defuse the bomb. Once a phase is disarmed, it becomes inactive and changing it doesn't affect the bomb. Instructions for disarming the phases are provided in this document.

Strikes

A mistake in disarming a phase results in a strike. Get too many strikes, and the bomb "explodes". Sometimes, the remaining countdown time will be decreased and/or go by faster when a certain number of strikes has occurred.

Information

A different version of the bomb is randomly presented each time it is "booted". There are 6,720 unique versions of the bomb with a whopping 1,176,000 possible variations!

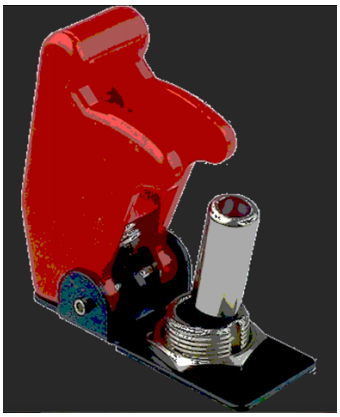
Disarming some phases will require specific information about the bomb. Pay close attention to the "bootup" text on the bomb's screen

Rules of the Bomb

This bomb has some rules associated with it. They are the following:

1. Keep an eye on the button. Red = Dead. Make sure to check it periodically and press it so it becomes green and don't let it stay red for more than a few seconds.
2. Get out your pen and paper, you might not be able to get the protective shields back up without some math.
3. Be swift, being near the Oganesson in this means you're exposed to the most radioactive element in the known universe. You start to feel radiation sickness almost immediately.

Regarding the Toggles



There's some important information to know about the toggles. They have two parts, both utilizing the **serial number** given to you. Do not touch the toggles until you know the correct combination. Here's how to decipher it:

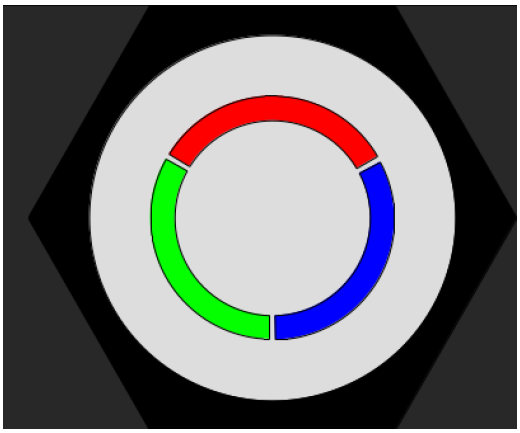
1. Keep the toggles all set to "off" to begin.
2. Find and take note of all the numerical digits within the serial number.
3. Find the sum of all these numbers.
4. Convert the sum from its decimal value to its base 2, or binary equivalent. and take down the four right-most digits. Note that if the value is less than 4 digits long you need to add the required trailing 0's.
5. Quickly, and in one motion, flip all required toggles on.

You may keep the toggles how they were from part 1 of the toggles phase. Now, going back to the serial number given, solve the second part:

1. Now, write down all letters within the serial number.
2. Convert all these letters to their associated decimal value, where A=1, B=2, C=3, ..., Z=26.
3. Next, find the sum of these and write it down.

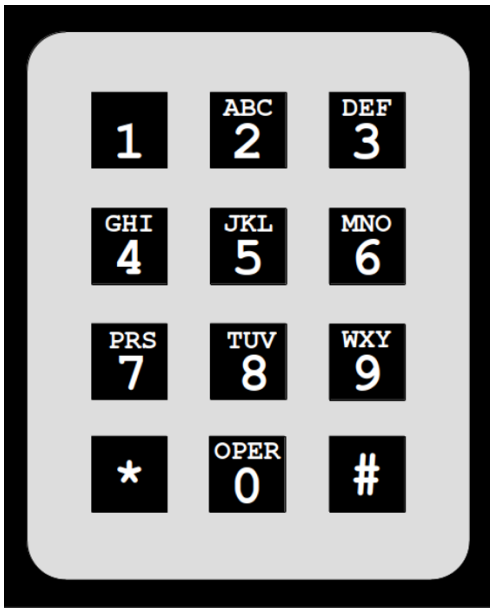
4. This number needs to be converted from decimal to binary as well, but you only need the four right-most binary digits. Again, add any needed trailing 0's.
5. Once you have your answer, flip the required toggles!
6. If done properly, this will now be disarm

Regarding the Button



This button is based on the timer. It will begin with a random color, between Red and Green. After some time goes by the button is going to change. Remember the first rule. Do not let the button stay red for too long. Green is the safe color. **Keep it green.**

Regarding the Keypad



This container utilizes RSA encryption. Luckily for you, the RSA tab on the bomb will do most of the work. Take note of all the defuse keyword and the keys. The values are given in the following manner:

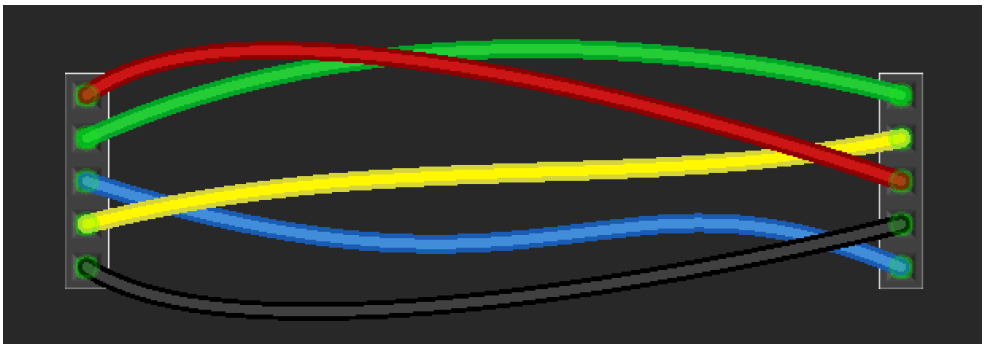
Defuse Keyword: “c-value”/ENC/“base #” / 0x“e-value”/ENC/”base #”/;

Keys: 0b “p-value” ; 0b “q-value” /ENC/ “base #”/;

After taking note of these, go to the RSA tab, which can be accessed from a mouse click or hitting the star (*) on the keypad. Quickly enter the decimal values for the given keyword/keys. If entered correctly, you will get a keyword. If a keyword does not appear, you must redo your steps. Here’s another tip: check the button, I bet you forgot rule 1 when you were trying to decrypt this.

What to do with your keyword: type the keyword onto the button panel, hit the pound key (#) to enter this value, the combination will say “disarmed” if the correct value was entered.

Regarding the Wires



Keypad required to be complete

Keyword dictionary is listed below, each letter is assigned a bitstring:

{'00000': 'A', '00001': 'B', '00010': 'C', '00011': 'D', '00100': 'E', '00101': 'F', '00110': 'G', '00111': 'H', '01000': 'I', '01001': 'J', '01010': 'K', '01011': 'L', '01100': 'M', '01101': 'N', '01110': 'O', '01111': 'P', '10000': 'Q', '10001': 'R', '10010': 'S', '10011': 'T', '10100': 'U', '10101': 'V', '10110': 'W', '10111': 'X', '11000': 'Y', '11001': 'Z', '11010': '0', '11011': '1', '11100': '2', '11101': '3', '11110': '4', '11111': '5'}

Each bitstring of length 5 given corresponds to a keyword. These values represent whether a wire should be plugged in (1) or unplugged (0).