Weeks 8 and 9. Reversing our first program: Interactive disassembly, static code analysis

---

Reverse Engineering: The Demo

- This is a live demo that will take ~3-4 class periods to complete. Unlike a traditional live demo, the structure of this one will be as follows.
  - As in paired programming, you (the professor) will be the driver. While the class provides directions.
    - You may help guide them in the correct direction, but the students should be coming to the answer and giving it to you.
    - This is not intended to fly by, let the students make mistakes, follow them, and help them work backwards when they realize that they messed up / went down the wrong rabbit hole.
    - Go for raised hands first, resort to just going down the aisle if desks if no one / the same 3 people are answering everything. There are no wrong answers.
- This demo will be conducted the same manner as the final project will be structured for the students.
  - Create or Download a simple crackable program. crackmes.one
  - With the intention of cracking or figuring out how to solve the program (discover and create a valid password / key, etc) ask what the first step should be.
    - Continue this line of questioning, do not be obtuse when it comes to tool commands (radare2 commands aren't expected to be memorized).
    - If the class is entirely stumped, throw a bone by googling some things that might jog their memory, moving on to the next step should only be done when absolutely necessary to move the class forward, this is intended to take more than 1 class time.
  - This should be a review and culmination of all of the classes presented thusfar.
- General steps that you should look for:
  - Using the strings UNIX function
  - Using the man UNIX function

G Leaden - Week 8 and 9 **Reverse Engineering**

- Using the file UNIX function
- Running the program
  - Test input
  - Test lots of input
  - Various input
    - Find out what info/error messages pop up when, start building a map of (when I add a number, my message changes)
  - Use the whiteboard or scratch paper
- Opening it up in radare2
  - VV mode
- Debug mode
  - With various inputs
  - GO SLOW
  - Step through the program.
- Slowly decipher when jumps occur to present the various info/error messages.
  - Add those to your map
- End goal should be deciphering what the requirements are for the program to complete successfully.
- This is extremely similar to the **poster** example presented for the final project.