I thought programs were supposed to be more secure now? Is it really even practical to try to do exploit development?

- Yes if you are skilled. Stat analyzers don't catch everything. Also, many people don't use them, do security testing on their products, or know anything about secure coding. So there tends to be several vulnerabilities in all software written. You can also change your client side executable to do things like: cheat in video games, get the full version of software, and find buffer overflows server side.
- What about programs that utilize security measures like DEP, ASRL, and SEH? Yes you can still exploit this it just takes a talented person. Not all programs use DEP, ASLR, and SEH though.

Do we need debuggers to develop exploits?

- Yes. Even if we know the buffer is say 1000 bytes long, we still don't know how we could potentially execute our own code. We need to put it in a debugger to know where to jump around, what parts of execution we can change, etc etc.

What is the difference between a disassembler, debugger, and a decompiler?

- **Disassembler**:
  A disassembler is a software tool which transforms machine code into a human readable mnemonic representation called assembly language.
- **Debugger**:
  Debuggers allow the user to view and change the running state of a program.
- **Decompiler**:
  Software used to revert the process of compilation. Decompiler takes a binary program file as input and output the same program expressed in a structured higher-level language.

So what tools can we use to get these things done?

- There are certain tools that can do all 3 of these things. IDA Pro and x64dbg are the two most popular at the moment. We will be using x64dbg because IDA Pro is extremely expensive and neither one is incredibly superior to each other. Immunity debugger and evans debugger also seem like great tools.
- "Detect it easy" tells us the compiler and linker so we know what to put in x64dbg

What tools do we need to use to exploit our target systems?

- For simplicity, all of our exploits will be written in python. You can write them in c/c++ if you like.

So we need to know two main things before getting started

- How are programs built/run?
- How does the operating system do all of these things with assembly instructions?

So are there any good practice or tutorials?

- Book: Practical reverse Engineering