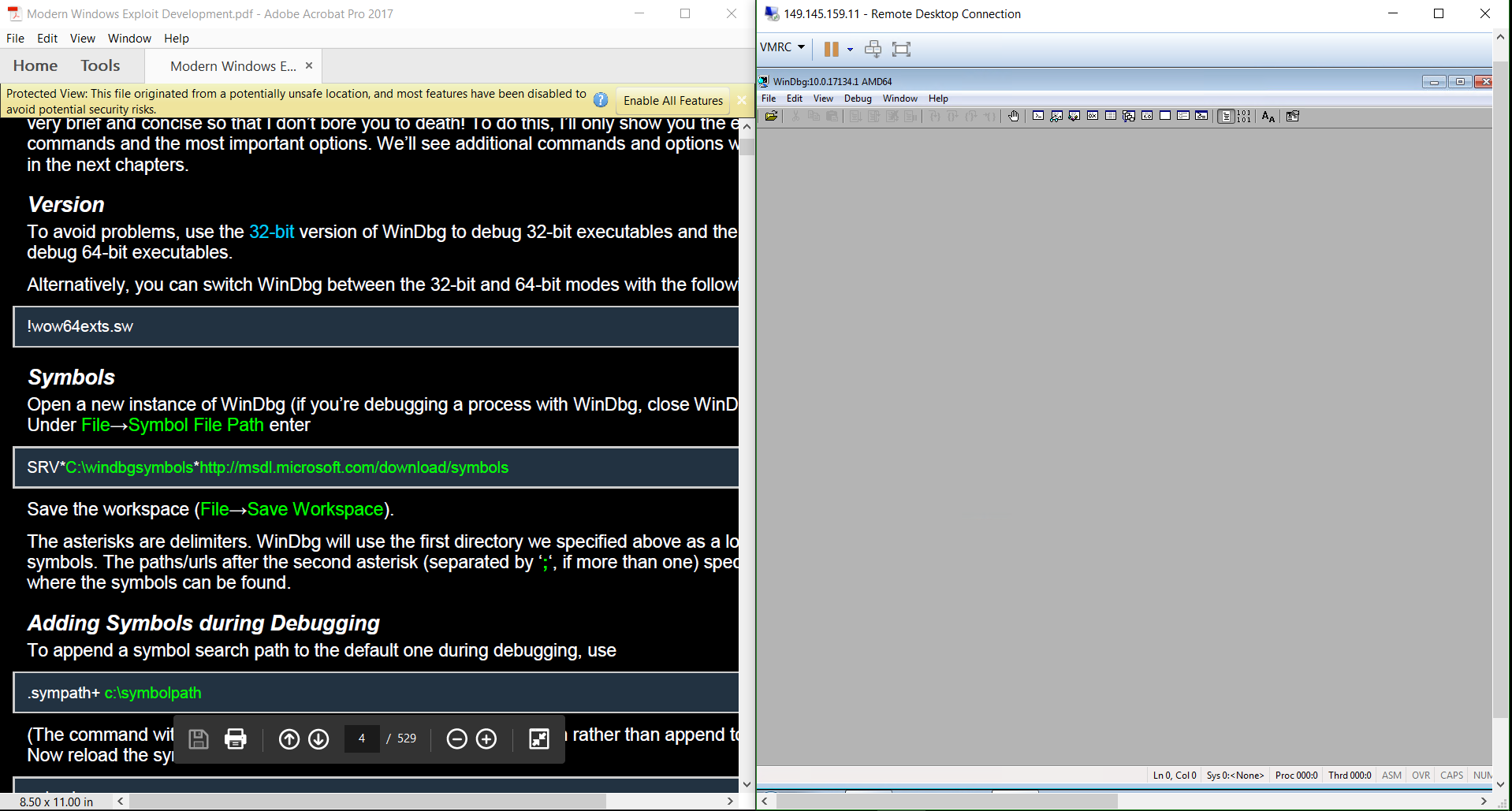
I spent 1-2 hours every day learning how to do these things and still didn’t get as far as I would have liked. If you don’t want to put in the time or mess around in assembly I’d drop the class now.

While students are getting their environment up and running for a week they should know what exploits are and how people find them. There is also lots of different types of exploit development but in this class we will just be covering c/c++ on Windows. Dealing with web applications is a whole different beast. They need to understand that exploits are found by analyzing source code or reverse engineering the executable. An attacker extract valuable clear text information out of binaries, change an executable, overflow integers, parse strings incorrectly, and most famously redirect the flow of the program’s execution. It is still important to conduct your own exploit development against products you field in today’s era. Developers should be trained on how to code securely, write tests, and use static analyzers to catch bugs, but there should still be exploit development done to validate the application is safe. Security is getting a TON better with operating systems and developer tools so that’s why we are doing this course in windows 7 while using up to date tools. By the end of this course you should understand how to develop more securely, how to decompile someone else’s code, how to exploit some basic vulnerabilities, and apply how attackers today could use these same methods.

Hopefully by this point your target machine is in a healthy state. If not look at the installation text file in this directory. You should be able to launch windbg64 and nicely view that and the book at the same time.

Make the directory “C:\windbsymbols” and another “C:\projects” for them to put their vulnerable code in. This windbsymbols directory will save all of your symbol syntax in the coming exercises.



A good intro on what this course is about is 64bit reverse engineering. Know how assembly works.

* <https://www.youtube.com/watch?v=75gBFiFtAb8>
* <https://www.youtube.com/watch?v=rxsBghsrvpI>

Also good to watch is what ida can do: <https://www.youtube.com/watch?v=fgMl0Uqiey8&list=PLt9cUwGw6CYG2kmL5n6dFgi4wKMhgLNd7>

What is a .dll file: <https://stackoverflow.com/questions/484452/what-is-a-dll>

What is a PE file: <https://www.youtube.com/watch?v=l6GjU8fm8sM>

Before diving in, watch this video to see what exactly windbg is: <https://www.youtube.com/watch?v=8zBpqc3HkSE>

What the hell is a symbol? Read all of this content to find out: <https://docs.microsoft.com/en-us/windows-hardware/drivers/debugger/symbols>

Summary of a symbol?

So looking at the symbols can tell you a lot about how a program works. They say hey if you want this function start right here and it has these parameters.

* Global variables
* Local variables
* Function names and the addresses of their entry points
* Frame pointer omission (FPO) records
* Source-line numbers

So what is the difference between a public and private symbol? Are public variables/functions bad?

Alright now that we know what symbols are, how does the computer know where each executable’s symbols are?

* By contacting SymSrv. This is a symbol store that keeps track of all symbols and can give them to your debugger when loading executables. Awesome!

So how are we going to use windbg?

* We are going to open it up and load our exercises in it. You could be fancy and have windbg open up a port on your test machine and remotely use windbg. But this doesn’t make a lot of senses because you won’t ever be reverse engineering applications on your target devices. So don’t worry about this.

Now take example1.exe, load it into windbg. Set the symbol search path to “C:\projects;SRV\*C:\windbgsymbols\*http://msdl.microsoft.com/downloads/symbols” without the quotes

Type in “lmn” to see all of the modules loaded

So how can we see info about our example1?

* X /v /f example1!
* <https://docs.microsoft.com/en-us/windows-hardware/drivers/debugger/x--examine-symbols-> is the cheat sheet

Goal for tomorrow

* Load in executable to IDA
* View the function
* View public and private variables
* View the flow in the if statement
* Try to search for the password string
* Try to decompile
* Try compile a visual c++ file
* What the hell is cheat engine?

Stuck

* I am not pleased with the book so far, chapter 1 was confusing and not clear at all. Many steps left out
* Right now I can’t get the symbol files loaded. Also there is an error with the Microsoft ones they say deffered not sure if that is right
* Now sure if the mona commands work in windbg

Try

* Looking at IDApro, ollydbg, windbg, evans, and immunity debugger. Get pros and cons and platforms. – looks like IDA free with plugins is the winner. IDA can use windbg as its debugger so I think the answer is to keep playing around with both books and tools at the moment.
* Look for other books on exploit development and windbg setup: IDA pro
* What are some decompiling tools? – snowman plugin for IDA and x64dbg
* Is there a good static analysis tool to run on a C/C++ source file?
  + Cppcheck