Extra Credit 1 Report

tags: `class`

My Pin Tool & Explanation

First, I used

```
INS_AddInstrumentFunction(Instruction, 0);
```

to register a function I wrote called `Instruction` to instrument.

In the `Instruction` function, I inspect the instruction to see if the instruction is the `cmp` instruction that dictates whether we will execute the malicious section of the program. Once the instruction is found, I use `INS_InsertDirectJump` and `INS_Delete` to replace the comparison with a direct jump to the malicious code.

```
VOID Instruction(INS ins, VOID *v)
{
    if (INS_Opcode(ins) == XED_ICLASS_CMP) {
        string insString = INS_Disassemble(ins);
        *out << "ins:" << insString << endl;
        if (insString == "cmp eax, 0x5"){
            *out << "insert\n";
            ADDRINT addr = INS_Address(ins);
            INS_InsertDirectJump(ins, IPOINT_BEFORE, addr+0xD);
            INS_Delete(ins);
        }
    }
}
```

What I Lerned

Although I learned conceptually how pin works in class, it still took me a long time to figure out how to use it to actually perform instrumentation. In the end, although my code and understanding were still rough on the edges, I managed to get my pintool to work. I also learned how to use IDA to analyze an executable.