

Hacker Challenge Phase 3 Instructions

1 Background

Phase 3 uses a similar codebase to Phase 1; however, the protections have been implemented in a different manner. As with Phase 1, the reverse engineer has two objectives: 1) to reverse engineer a formula; and, 2) to extend the application by removing a limit.

As with Phase 1, the application uses two files, the application itself (final.exe) and the data file (data.txt). On first run, the program will not work properly - it seems to require a password. Upon finding the correct password, the application will produce the results seen in Figure 1 and stored in the file **gold.txt**.



```
c:\> Command Prompt
1 4 11.0938
35 0 0 0 0 0 0
19.4142 155.586 1
2 4 58.9931
35 0 0 0 0 0 0
58.9931 41.0069 2
3 7 -59.0079
35 17 10 5 6 10 8 4
-103.264 278.264 1
4 7 -59.1907
33 17 10 5 6 10 8 4
-115.422 310.422 1
5 3 -17.8219
35 14 5 0 0 0 13 0
-31.1883 206.188 1
6 3 -20.395
35 0 0 0 0 1 1 1
-20.395 120.395 2
>
```

Figure 1: Program output after password is found.

2 Objectives

Objective 1: Reverse Engineering a Formula

The first objective requires the reverse engineering of the formula that produces -59.0079 that can be found in the 3rd column of the 7th row of the output file (see Figure 2).

Be sure not to track the input flow of the data values being used in the program. The focus should be on the formula from the invocation of the function call. For example, if the formula uses data values, global values, or parameters, just prefix each of them with a **d** (for data), **g** (for global), **p** (for parameter), etc... For example:

```
Int global_var = 0xBEEF;
Foo (0xDEAD);

Void Foo(int param1)
{
    int result = param1 + 3.14 + global_var;
```

}

Do not worry about where the param1 came from, just give the formula in pseudo code form:

```
result = p1 + 3.14 + g1;
```



Figure 2: Find the formula that produces the circled value

2.1 Objective 2: Anti-tamper

There is an upper limit to the second field in each row of data.txt that will not allow you to use a value of 200 or more. The anti-tamper objective is to remove this limit and to demonstrate the removal by changing the 4th line of the datafile from "195" to "220" (see Figure 3).

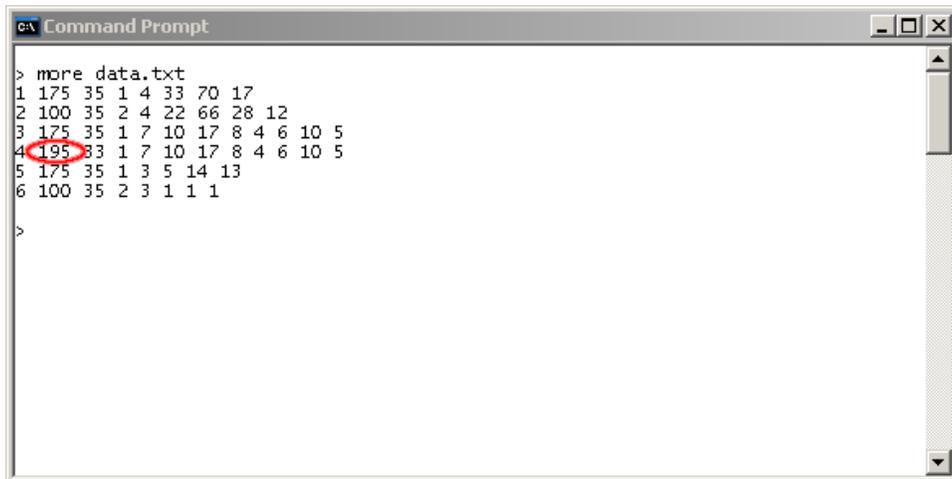


Figure 3: Change this value to 220

Figure 4 shows the correct output if the data.txt value "195" is changed to "220." The circled values will indicate a successful break of this limitation. These results are also stored in the file **gold2.txt**.



```
C:\> Command Prompt
1 4 11.0938
35 0 0 0 0 0 0 0
19.4142 155.586 1
2 4 58.9931
35 0 0 0 0 0 0
58.9931 41.0069 2
3 7 -59.0079
35 17 10 5 6 10 8 4
-103.264 278.264 1
4 7 -59.1907
33 17 10 5 6 10 8 4
-130.22 350.22 1
5 3 -17.8219
35 14 5 0 0 0 13 0
-31.1883 206.188 1
6 3 -20.395
35 0 0 0 0 1 1 1
-20.395 120.395 2
>
```

Figure 4: A successful defeat of the code limit.