Sang Choi
Kevin Wood

CS 5463 Project Report - Team 12

For analysis, IDA was the primary tool used. The entrypoint was buried in a bunch of boilerplate code that made it difficult to locate the main function. Running the program provided error strings to search for which helped locate user code. The check for command-line arguments and the use of the default filename “encrypted.txt” provided the key clues for identifying the main function.

Functions and variables were then named based on perceived purpose. Within many system functions, names were found referenced within the assembly code(perhaps debugging information?), which aided in identification of these functions. Cross-referencing with the provided C++ file template for the encryption program also aided in identifying patterns and deciding on variable and function names in the code.

After fully identifying the structure of the main and decryptFile methods 2 key sections were identified to be implemented within our encryptor. First, the password is read as the first line of the file and a hash generated to use in encryption. Although the acquisition of the password and hash already existed within the template, this information was used when coding the file output code. The second section of interest was the encryption loop, which performed 4 modifications on each character of the data.

- Step 1 - rotate left 7 bits, with a branch never taken
- Step 2 - swap the high 4 bytes with the low 4 bytes
- Step 3 - swap bytes 1-2 with bytes 3-4 and swap bytes 5-6 with bytes 7-8
- Step 4 - xor with byte 23 or byte 8 of the password hash, alternating every 4 characters

Most analysis of these steps was performed statically with IDA, although dynamic analysis of Step 4 was performed using the IDA debugger to verify the behavior alternating the xor character.

Following this analysis an initial version of the encryptor was built using Visual Studio and it was tested with the decryptor. After that failed spectacularly a set of modified decryptor executables were employed, each modified with HxD to only perform a single step of the decryption process. Using these modified executables it was determined that steps 1 and 4 had bugs.

After some investigation it was determined that the character needed to be cast to an unsigned char before shifting right to ensure the right shift produced 0s in the high bits for step 1. For step 4 it was determined that the operation was starting using byte 8 instead of byte 23.

Sang Choi
Kevin Wood

The encryption code was compiled and tested frequently using Microsoft Visual Studio. Once the entire flow from encryption to decryption was validated with the provided file, the generated executable was tested with different passwords and secret files for final testing.

Primary contributions were as follows, although both team members collaborated heavily over the course of the project.

- Sang Choi - Analysis of steps 1, 2, & 3, second coding pass to finalize methods, compilation, debugging, and testing of encryption program
- Kevin Wood - Analysis of main, decryptFile, & step 4, first coding pass to rough out methods, creation of alternate decryptor executables for testing, report