Joseph Arbolino

SRE Assignment 7

April 5, 2022

**Procedure Summaries**

**Question 1**

To unpack part1.exe I started by analyzing it in Ghidra and finding the entry point:

A picture containing text

Description automatically generated

The first thing I noticed was the PUSHAD command which was mentioned in the lectures. I then used x32dbg to follow the PUSHAD to the POPAD and locate the relevant JMP command:

Graphical user interface, application, table, Word

Description automatically generated

Following that jump command leads to the unpacked code that I was searching for:

Text

Description automatically generated with low confidence

From there I used Scylla to find the IAT and dump the binary to a file. I also used Scylla to fix the binary and prepare it for execution. I received part1\_dump\_SCY.exe from this procedure:

Graphical user interface, application, Word

Description automatically generated

Here is the unpacked entry:

Table

Description automatically generated with low confidence

With this the executable has successfully been unpacked.

**Question 2**

To unpack part2.exe I again started by analyzing in in Ghidra and finding the entry point:

Table

Description automatically generated

I noticed immediately that this file is using a different packing method than the first because there is no PUSHAD command. Because of this, I began looking for very long jumps to large blocks of obscured code. I found this:

Text

Description automatically generated with medium confidence

Which led to:

Table

Description automatically generated

Using this address in x32dbg I was able to find the packed code and use this address to find the IAT and get the imports:

Table

Description automatically generated

Using Scylla at this address:

Graphical user interface, application

Description automatically generated

Form there I dumped the binary and fixed it using Scylla and received part2\_dump\_SCY.exe which I then analyzed in Ghidra. Here is the unpacked entry:

Text

Description automatically generated

**Question 3**

To unpack part3.exe I again started by analyzing it in Ghidra and finding the entry point:

A picture containing table

Description automatically generated

I noticed that this looked different than the first two files and had a JMP command in it. I followed the JMP command to find it led to a PUSHAD command like in part1.exe:

Text

Description automatically generated with low confidence

From here I followed the steps from part1.exe and loaded the binary into x32dbg to find the POPAD command and the tail jump:



POPAD:

A picture containing text

Description automatically generated

Following the JMP command at this location led to the rest of the packed code:

Graphical user interface, application, table

Description automatically generated

From here I used Scylla to unpack the binary and received part3\_dump\_SCY.exe from this process:

Graphical user interface, application

Description automatically generated

Now that I had the binary unpacked, I reloaded it into Ghidra and began trying to patch it to have any username and password activate the victory screen. Here is the unpacked entry:

Table

Description automatically generated

I followed the strings output to the function responsible for display the victory message box:

Table

Description automatically generated with medium confidence

I then found all references to this function and found a command calling it:

Table

Description automatically generated

The current command to call the win function is a JNC command. To patch this binary, I changed this line to a JMP command:

Graphical user interface, application, Teams

Description automatically generated

This allows the program to jump unconditionally to the win function regardless of what the user enters into the program. There was a strange effect of doing this in the program however, only usernames greater than 3 characters and passwords greater than 0 characters work. The user can put any combination of letters, symbols, and numbers, but they must be greater than the previously mentioned values. I am not sure if this behavior is intended or if it is a result of a poor choice on my part for the patch solution. With this the binary was successfully unpacked and patched.