Legacy Code: Two Changes Needed to Redistribute 18-Year-Old Software

CS 4991 Capstone Report, 2022

Parth Raut
Computer Science
The University of Virginia
School of Engineering and Applied Science
Charlottesville, Virginia USA
plr6uqx@virginia.edu

ABSTRACT
Eritek Inc., a tech company based in Leesburg, Virginia, wanted to begin redistributing software they had originally written in 2004. However, that plan faced some issues because the company that helped with licensing the software went bankrupt and part of the software was written for 32-bit systems. To solve these issues, I had to write my own licensing algorithm for the software as well as rewrite the 32-bit pieces of code using a more modern library. For the licensing algorithm, I utilized RSA encryption to generate and verify the license keys for the software. I also used the Qt library to update the GUI code from 32-bit to 64-bit. In the end, I was able to produce a licensing algorithm that effectively verified the validity of the software and a GUI with functionality that fit modern 64-bit standards. In the future, this software will be ready to be distributed again in its updated version.

1 INTRODUCTION

What happens when the software you want to distribute was written over 15 years ago and the company helping with licensing that software goes bankrupt? It is time to rewrite parts of that software. This was the task that I was given as an intern at Eritek Inc., a tech company based in Leesburg Virginia, during the summer of 2021.

Software licensing is a way to ensure that a user has legitimately obtained the rights to use that software. This is especially important for commercial software, which often requires users to pay a fee to use it. However, this fee often leads to piracy, which financially hurts the companies that develop the software financially. Because the company I was working for no longer had a way to license their software, I had to write my own licensing algorithm. The algorithm utilized RSA encryption which was used to generate and verify the license keys that was distributed to the users.

Additionally, the software that I was working with was originally written in 2004, so parts of the code were written for 32-bit systems. This meant that the software was not up-to-date with modern 64-bit standards. While a 32-bit software can run easily on a 64-bit system, the discrepancy limits its capabilities by not meeting the modern standard. The part of the code I needed to update was the GUI. I used the Qt library, a C++ library that provides GUI functionality to desktop applications, to update the GUI from 32-bit to 64 bit.

2 RELATED WORKS

The RSA algorithm is an asymmetric cryptographic algorithm [1]. This means that the algorithm uses two keys, a public key and a private key, in order to encrypt and decrypt data. In general, the public key is used to encrypt data and the private key is used to decrypt it. Because of this, the RSA algorithm assumes the use of a one-way function for encryption and decryption. The formula for encrypting plain text is

$$C = Pe \,\%\, n$$

where *C* is the encrypted cipher, *P* is the plain text, *e* is the public key, and *n* is the product of two coprime numbers. *P* is calculated using the ASCII decimal values of each character in the plain text. The formula for decryption is

$$P = Cd \,\%\, n$$

where *P* is the plain text, *C* is the encrypted cipher text, *d* is the private key, and *n* is the product of the same coprime numbers [1].
This information was vital to writing the licensing algorithm since I was going to be using RSA

encryption/decryption to accomplish it. While I ended up using a library to compute most of the RSA functionality such as generating keys and encrypting/decrypting text, understanding the background and math behind the algorithm was important for me to successfully implement it into my solution.

3 SYSTEM DESIGN

The design of the system included three parts: generation of the license file, validation of the license file, and updating the GUI.

3.1 License File Generation

At the heart of the licensing algorithm, RSA encryption was used to generate and verify the license keys for the software using the OpenSSL library. These license keys were strings of encrypted text that consisted of the contents of the license file. The license file was a JSON file that contained the attributes of the license such as start date, end date, product number, software version, MAC address of the system that the software will run on, and the license key. In addition, there is a cookie attribute that has a value of an encrypted string containing the date and time of the last time the software was opened. A typical license file could look like Figure 1:


```
{
  "device-id":  "D0:50:99:19:EC:09",
  "product":  {
    "number": "10000000",
    "version":  "1.0",
    "start-date": "2021-08-01",
    "end-date": "2021-08-31"
  },
  "license":  "SlyXwGTDJOYeQ96z0HdckpZb9iYSH4+TtLJlEUn1klcBoR/
  ifQ589VSGA98isuHteG81weln0rlQyWoMAyvmNCExAqTv+HpDoVDHNpt7SGI6H1UHx3ZjfkS6r7
  bGzFeQMZLswm9d7p9n8dDuiDejxxKcFRLqdeTTXo3mhQkznhzNeVmGEHC5xPjnyXetfu8hWB2Hl
  3gyvKXowKvGRutdbc183gEOQDD1ot14pWJwcCeMIcjeX4Yv7P7zIX+vZ75Hipq8f/
  DVvV15f9ERq2JyG/
  fagr8EJnq9VFoXyCF7L8fRmFyu6fkrYfp6zijRhLTPBnAq+CAnycusRfRqlJD/ZA==",
  "cookie": "WmDxvb1eSnssUw3dvlJmq6d2Z5O3yvXVmKOOnvBNvTVPO/
  T7QLwkmJnGfg+DEWNVkfgJQpgQxP/qN+99N4kfybesixlAdBlK0Iv1YIv/
  Uh5mPUDepILciZ95S/5gHR74I79MJdVsY8K/
  AaVqOp1bTVxFTLpPg8FI31ItmJ56QjqBtXDy+MZzMi39EiKtnZaSkewxA6k5jOX7mlOj5xAxp/
  CptYaJqbguW/fcTOBfomzMMtHAf+Z3c5RwwULs/
  cNGPNOl+okgf0kpBSbyJZqH0zfySENthunWhTpWEyS0yvfY89MBl4At4qZXbm7gvXJO8ljNlna7
  Haml/wFqn/x2XA=="
}
```


Figure 1: Example License File

The reason ~~that~~ the MAC address is part of the license file is because it is the main identifier for whether the computer running the software is allowed to use it. The MAC address value is generated through a simple routine that checks for ethernet MAC addresses on the machine that will be running the software. Once all ethernet MAC addresses have been found, the routine displays them to the user as well as a company email that the user can send them to.

Once the company receives the MAC address(es) of the user, they can be inputted into a GUI that generates the license file. This GUI contains fields that can set the start date, end date, software version, and MAC address attributes of the license file. The license key attribute is also filled by encrypting the other attributes in the format

$$"MAC\ address, product\ number, version, start\ date, end\ date"$$

using RSA encryption. The cookie attribute is left blank because the user has not opened the software yet. The newly generated license file is then sent back to the user to be used by the software.

3.2 License File Validation

In the license file validation routine, the first thing it does is decrypt the license key attribute value to get the values of the license file when it was generated. These values are then checked against the values that are present in the license file. If there are any discrepancies, the license file is considered invalid and the software is not allowed to run. This is to ensure that the license file was not tampered with in an attempt to gain unlicensed access to the software. The license key essentially serves as a validation key for the license file.

The next part of the validation process involves checking the MAC address attribute. The routine gets the MAC address from the computer and compares the value returned to the one in the license file. If the MAC address values do not match, the license file is considered invalid.

Once the MAC address validation is complete, the routine checks the validity of the cookie. The cookie attribute is decrypted to get the date and time of the last time the software was opened. The purpose of the cookie is to ensure that the user does not attempt to change the date and time of the computer in order to bypass the license file. The routine checks whether the computer's date and time comes after the date and time stored in the cookie. If it passes this check, the current

date and time is compared to the start and end dates in the license file. If the current date is not within this range of dates, the license file is considered invalid.

If the license file validation routine determines that the license file is valid, the user is allowed to run the software. When the user closes the software, the date and time stored in the cookie attribute in the license file is updated and re-encrypted.

3.3 Updated GUI

Because the previous GUI of the software was written using a 32-bit library, I needed to find a new 64-bit library. Additionally, because the rest of the software was written in C, I needed to find a library that was compatible. The library that I chose was the Qt library in C++, a derivative language of C.

C++ contains classes, which help organize data and their modifiers and accessors. Each class can take on the properties of other classes by extending other classes. The Qt library contains a *Widget* class that I could extend. A *Widget* is any GUI element that that can be displayed. For example, a *Widget* could be a window, button, or drop-down box. For this reason, for every window the software had, a new class was created as a *Widget*. Widgets can also be added to each other, so buttons, drop-down boxes, and text fields can be added to a window to create a full GUI. Because I was using a new library, I had to rewrite the GUI from scratch using the Qt library. This method of organization is the design I used to implement the GUI.

4 RESULTS

Currently, my code is being implemented into the rest of the software. Because my code was written almost externally from the rest of the software, the proper functions calls need to be made in the software to run my code. Additionally, the existing functions that were written in the software need to be called in the updated GUI to provide proper functionality to the GUI.

Once everything is properly implemented, the software will be ready to be redistributed. Due to the previous licensing issues and outdated code, the software could not be given to clients who wanted to use it. The company had to turn down those clients because of these issues and most likely lost them to competitors. However, now that the software has the necessary updates and features, it can be distributed to clients.

Additionally, the project was left untouched for years because the company was busy working on other projects. Through my efforts, I was able to revitalize the project and offer the company a way to redistribute the software, even though it is over 18 years old.

5 CONCLUSION

This project's end goal was to update and redistribute software that was originally written in 2004. There were two main parts of this project that needed to be done to reach this goal: writing a licensing algorithm for the software and updating the GUI to 64-bit standards. The licensing algorithm utilized RSA encryption/decryption to generate and validate the license keys. Additionally, the GUI was completely rewritten using the Qt library. While the software isn't completely ready for redistribution because my code is being implemented into the rest of the software, all the components necessary have been completed. After years of being outdated, the software will finally be updated to meet today's standards.

6 FUTURE WORK

As of now, most of the hard work of updating the software has been done. The only aspect that is left is integrating the code that I wrote into the rest of the software. This would require making the proper function calls in both the software and my code. In addition to this, testing the software to make sure the integration does not cause any problems will also need to be done. While I am not the one doing this integration, others at the company are. The project is expected to be completely done by the end of April 2022.

REFERENCES

[1] Hengki Tamando Sihotang et al 2020 J. Phys.: Conf. Ser. 1641 012042

[2] OpenSSL. 2021. OpenSSL Cryptography and SSL/TLS Toolkit. [online]: https://www.openssl.org/