New Approach on Steganalysis: Reverse-Engineering based Steganography SW Analysis

Hanseong Lee
Div. of Computer Engineering
Hanshin University,
137, Yangsan-dong, Osan, Gyeonggi,
+82-10-5698-0599, Rep. of Korea
jkhanseong@naver.com

Hyung-Woo Lee
Div. of Computer Engineering
Hanshin University,
137, Yangsan-dong, Osan, Gyeonggi,
+82-10-2228-8299, Rep. of Korea
hwlee@hs.ac.kr

ABSTRACT

Recently, steganography tools for concealing messages in images have been widely used, and internal mechanisms for hiding messages using steganography tools are mostly unknown to public. Therefore, we adopted the reverse engineering for unveiling the information hiding algorithm and operation mechanism applied to 'Steg' steganography software using IDA tool. By reversing the steganography software, we disclosed the internal steganographic mechanism used on 'Steg' software. Therefore, we can propose a new approach for efficiently detecting and automatically extracting hidden messages in stego data using the software reversing technique.

CCS Concepts

- Security and privacy → Software reverse engineering
- Security and privacy → Software and application security → Software security engineering

Keywords

Steganalysis; Reverse Engineering; Information Hiding; Steg Software; Anti-Forensics Tools; Hidden Message Detection.

1. INTRODUCTION

Steganography [1] is an art and science of communicating information in a covert manner such that the existence of this communication is not detectable. Therefore, steganography is the art of covered or hidden messaging. It is far different from cryptography which is the art of making something inevitable to understand, unless the cryptography key is known. Steganography hides a message in a cover medium, but no one understands hidden message's existence unless he is aware of it. Steganalysis [2] is the opposite procedure of steganography. Primarily, we try to detect the existence of steganographic content in a digital device and secondly discover the hidden message. From this point of view, steganalysis can be classified into two major categories i.e. passive or active. Passive steganalysis tries to classify a cover medium as stego and identify the steganographic embedding algorithm, while active steganalysis additionally tries to estimate the embedded message length and ideally extract it from the cover medium[3,4]. Most of the Steganography tools do not disclose information hiding

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.

ICSCA 2020, February 18–21, 2020, Langkawi, Malaysia

© 2020 Association for Computing Machinery.

ACM ISBN 978-1-4503-7665-5/20/02...\$15.00

<https://doi.org/10.1145/3384544.3384571>

algorithms and operation mechanisms applied software inside. And it is very difficult to determine whether information is hidden in arbitrary image file. Anyone can easily download and install the Steganography tool via the Internet, and the recently developed Steganography tool can hide the encrypted message by applying the cryptography technology. Most of the Steganography tools do not disclose information hiding algorithms and operation mechanisms applied inside software. Therefore, it is very difficult to determine whether information is hidden when the message is hidden using the Steganography tool. In this study, a reverse engineering process was performed on the "Steg" steganography tool [5]. And we performed the reverse engineering analysis process and found the secret steganographic algorithm and operation mechanism applied to the Steg software. We also proposed a method to automatically detect and extract hidden message concealed in stego image files using the Steg software.

2. ABOUT "STEG" SOFTWARE

Steg Easy cross platform steganography [5] is a cross-platform and portable software, written in C++. It uses steganography and cryptography techniques to hide information inside uncompressed and compressed images. JPEG (JPG), TIFF, PNG, BMP image formats are supported. It really hides arbitrary data inside the image itself (not in metadata/comment fields or appended to the end of the file). With its easy graphical user interface, it is possible to fine tune steganography parameters, evaluates image's artifacts and uses both symmetric-key and asymmetric-key cryptography.

Figure 1. Steg SW (<http://fabionet.org>)

Steg runs on GNU/Linux, Microsoft Windows and Apple Mac OS X. And quite a lot of users are using Steg software. However, the details of the structure and features of the message hiding mechanism applied inside the Steg software are not known at all, except as disclosed on the homepage. Therefore, in order to analyze the Steg software, the conventional method was to estimate the

internal mechanism applied to the tool by analyzing the differences of Stego Image generated through the concealment process. However, this process requires a lot of effort and time, and has a problem that the accuracy of the reconstructed internal algorithm is not so high.

Figure 2. Steg SW User Interface and Function

Steg software provides the ability to hide text and binary files in the input cover image. It is known that noise can be added in this process, an RGB channel can be set in the concealment process, and encryption can be performed on a hidden message using a public key method as well as a symmetric key encryption method. However, it is not known at all how the concealment process is performed internally. Therefore, in this study, we analyze the basic functions, structure, and features of Steg software, which provide the following functions, and estimate the internal working method expected to be used by the tool. We propose a method to reconstruct the concealment algorithm applied inside the software.

Figure 3. Reverse Engineering based Steganalysis

3. “STEG” SOFTWARE REVERSING

3.1 Reversing on Message Generation Module

The *Interactive Disassembler* (IDA) [6] is a debugger and disassembler that is often used by reverse engineers to analyze programs. Additionally, IDA creates a flow chart for functions to help the developer understand which code paths are used in which case. IDA analyzes the exported and imported functions and automatically creates so-called *XRefs* between them. This allows the user to search for a specific import library and automatically list all occurrences of that method call in a window. Another important point is that IDA automatically parses all strings in the executable and lists them in a separate sub-view. This allows quick navigation

through the code by looking for a string that is known to be used at a specific point in the application. Therefore, by running Steganography software directly using IDA, the internal operation process can be analyzed inversely, and through the disassembly process in the executable file, the operating mechanism and algorithm applied in the executable file can be inversely estimated. Therefore, we can create an internal flowchart of the executable file by analyzing the start function in the executable file step by step, and identify various unknown information or hidden algorithms reversely. The figure below shows the internal structure reconstructed based on the result of reversing the Steg software using IDA. For messages (files) to be hidden in the cover image, the Steg tool generates a header with a SHA-1 [7] based hash. At this time, the hidden file name and file size are stored in the header, and the SHA-1 hash value for the original hidden message is added to ensure its integrity.

Figure 4. Reversing of Steg Message Generation Step

3.2 Reversing on Message Encryption Module

The integrated Steg message generated above is encrypted using the key entered in the concealment process. Using the IDA tool, we analyzed the message encryption process and concealing mechanism for the encrypted message using the IDA tool as shown below.

Figure 5. Reversing of Steg Message Encryption Step

In the case of the Steg tool, when a message or file to be hidden is entered, it is padded with 'x' characters, and then each block is symmetrically encrypted by AES-256-CBC [8] algorithm using the user-specified Encryption Key or the internal default key value. And it was confirmed through reversal process that IV (Initialization Vector) random number is concatenated with cipher

text as a to be concealed. And it was confirmed that the decoding process proceeds in the reverse process.

Figure 6. Reversing of Hidden Message Encryption Algorithm on Steg SW

The process of encrypting and concealing a hidden message in the Steg tool and vice versa is as follows.

- Message encryption and concealment method: Steg tool creates its own cipher text based on the concealment option, the Crypto Mode selection information, and the hidden message size information.
- Hidden message extraction and decryption method: Extracts hidden messages in the image using the hidden option, Crypto Mode selection information and input Stego image file set during Stego image generation, and performs decryption process using AES-256-CBC block mode method. And restore the hidden message finally.

Figure 7. Reversing of Embedded Message Encryption Step

3.3 Reversing on Data Hiding Module

Messages encrypted with AES256-CBC are hidden at the LSB location of the cover image. At this time, the method of concealing the image is considerably more complicated than the existing Steganography tools. 1 byte for storing the encryption mode portion and 4 bytes for storing the data size portion were allocated. Total 5 bytes information is hidden in the RGB portion of the pixel in the image. The encryption mode and data size concealment are affected by the RGB option set by the user, and the offset value where the concealment starts is changed each time according to the RGBA option set by the user. Therefore, it can be seen that the encrypted data is hidden in non-linear form at different offset positions according to the RGB option or encryption mode and data size information. Through the reversing process, the final formula applied to the concealment process could be derived as shown

below. Using the formula, Stego image files are generated by hiding messages in non-linear order for each pixel in the cover image based on the Crypto++ library.

Figure 8. Reversing of Data Hiding Step with Reconstructed Data Hiding Formula

3.4 Analysis of Message Hidden Pattern

The following figure shows how the hidden location varies depending on the size of the hidden data and the RGB options set by the user. In the figure, the first red part shows the pixel where the option information is hidden, the green part shows the encryption mode and data size information, and the black part corresponds to the location information where the encrypted hidden data is stored. The right part of the figure shows an enlarged view of the options, the encryption mode and the location where the data size is hidden. As shown in the figure, it can be seen that the encryption mode information is stored linearly at a fixed position in the Stego image. The encryption mode and data size are stored linearly from a specific offset position. However, it can be seen that the encrypted data is distributed and hidden in a non-linear form, and the distribution position varies depending on the data size and RGB options.

Figure 9. Visualization of Data Hiding Location based on Option, Mode and Data Size

4. DETECTION AND EXTRACTION OF HIDDEN MESSAGE FROM STEG SW

4.1 Hidden Message Detection/Extraction Algorithm

Detecting and extracting hidden messages in Steg software begins with the validation process. After importing an arbitrary image and converting it to an image library, pixel information is obtained. The first step in detecting hidden messages is to extract the RGB option information hidden in pixels (0, 0) to (0, 7). The extraction process applies the formula to extract the Config Offset value of the RGB image option. After extraction, it calculates the Pixel Index value and creates each Image Pointer object. From the Image Pointer object, we calculate the encryption mode information, Steg message size information X, Y coordinates, and RGB LSB order. After verifying the extracted information, if the extracted information is a valid Crypto Mode value, the AES256 decoding process is performed after extracting the value stored in the calculated RGB LSB position.

Figure 10. Proposed Hidden Message Detection and Extraction Algorithm

Lena.bmp was used as the cover (or carrier) image and the README-win32.txt file included in the Steg software distribution was hidden.

Figure 11. Implementation of Hidden Message Detection Module from Steg Steganography Tool

Stego images created in Auto mode can be extracted directly from hidden messages without entering a password. For Stego images in Symmetric PassPhrase mode, we must enter a password. By parsing the Steg message structure hidden in the image, three files are stored: the hidden data file name, the data file, the SHA-1 value for the data file, and the embedded text message set in the general options. As a result, it is possible to check the hidden data extracted normally in both Auto mode and Symmetric PassPhrase mode.

Figure 12. Implementation of Hidden Message Extraction Module from Steg Steganography Tool

4.2 Comparison

The following Table 1 show the comparative analysis of the proposed Steganalysis methods with existing mechanism [9,10,11,12,13]. Therefore, comparing with the existing techniques, the proposed reversal-based steganalysis technique provides improved functions in terms of complexity, efficiency, and performance than the conventional technique.

Table 1. Comparison of Steganalysis Methods

Methods & Comparison	Signature or Statistical Analysis	Universal or Blind Steganalysis	Reverse Engineering based Steganalysis
Used Mechanism	Pattern/LSB Matching, etc.	Machine Learning, SVM, CNN, GAN, etc.	Software Reverse Engineering
Main Objective	Detection & Extraction of Concealed Message	Classification of Used Stego Mechanism	Disclosure of Message Hiding Algorithm
Core Input Data	Cover or Stego File (JPG/PNG)	Stego and DB File (IMG)	Executable File (EXE)
Complexity	Low	High	Middle
Efficiency	Low	Middle	High
Performance	Low	High	High

5. CONCLUSIONS

Most of the Steganography tools do not disclose information hiding algorithms and its operation secret mechanisms applied inside. In this study, we analyzed the internal structure and operation method of steganography software such as Steg tool. Specifically, we performed the steganographic software reverse engineering and analyzed the information hiding algorithm with operational mechanism applied to steganography tool. Through this complicated reversing process, it is possible to provide a more accurate discrimination process for the stego image that has been hidden by the message, and the detection function for the hidden data can be provided. Therefore, in this study, by applying the reverse engineering technique, we can propose a new method which is more efficient than the conventional steganalysis technique and improved in terms of complexity and performance.

6. ACKNOWLEDGMENTS

This work is partially supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) (NRF-2017R1D1B03035040).

7. REFERENCES

- [1] J.R. Krene, "Steganography and Steganalysis," [Internet] 2019 Nov [cited 2019 Dec. 11]; Available from : <http://www.retawprojects.com/uploads/steganalysis.pdf>
- [2] K. Karampidis, E. Kavallieratou, G. Papadourakis, A review of image steganalysis techniques for digital forensics: *Journal of Information Security and Applications*. Elsevier. 2018; 40:217–235. DOI:10.1016/j.jisa.2018.04.005.
- [3] Arooj Nissar, A.H. Mir, Classification of steganalysis techniques: A study, *Digital Signal Processing*, Vol.20, No.6, Dec. 2010, pp.1758-1770, Elsevier, DOI: 10.1016/j.dsp.2010.02.003.
- [4] P. Richer, Steganalysis: Detecting hidden information with computer forensic analysis, SANS Institute Information Security Reading Room, 2019, [Internet] 2019 Aug [cited 2019 Nov 5]; Available from: <https://www.sans.org/reading-room/whitepapers/steganography/steganalysis-detecting-hidden-information-computer-forensic-analysis-1014>.
- [5] Steg Easy cross platform steganography, [cited 2019 Nov 5]. Available from: <http://fabionet.org/stegdownload>
- [6] IDA Support: Download Center, Hex-Rays, [cited 2019 Nov 5]. Available from: <https://www.hex-rays.com/products/ida/support/download.shtml>
- [7] National Institute of Standards and Technology (NIST). FIPS-180-4: Secure Hash Standards (SHS) (2015), <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf>
- [8] National Institute of Standards and Technology (NIST). FIPS-197: Advanced Encryption Standard, AES (2001), <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf>
- [9] F. Jessica, G. Miroslov, Practical Steganalysis of Digital Images – State of the Art, *Proceedings of SPIE – The international Society for Optical Engineering* 4675, DOI: 10.1117/12.465263.
- [10] T. Quach, Extracting hidden messages in steganographic images: *Digital Investigation*. Elsevier. 2014; 11:S40-S45. DOI:10.1016/j.diin.2014.05.003.
- [11] O. Lysne, The Huawei and Snowden Questions, *Shimula Springer Briefs on Computing*, Chap 6. Reverse Engineering of Code, p.47-55, 2018 Feb; DOI: 10.1007/978-3-319-74950-1_6.
- [12] M. Pavani, S. Naganjaneyulu, C. Nagaraju, "A Survey on LSB Based Steganography Methods," *International Journal of Engineering and Computer Science*, Vol.2, Issue.8, pp. 2464-2467, August. 2013. [cited 2007 Feb 22]. Available from: https://pdfs.semanticscholar.org/496a/a39eab2244ca6b95e0f06c5d886c3b43058b.pdf?_ga=2.30132287.1643476557.1565772102-1769689455.1520330387.
- [13] Sabnis, S. K., Awale, R. N. Statistical Steganalysis of High Capacity Image Steganography with Cryptography: *Procedia Computer Science*, 2006; 79:321–327. DOI:10.1016/j.procs.2016.03.042.