ImageDetox: Method for the Neutralization of Malicious Code Hidden in Image Files

Dong-Seob Jung¹, Sang-Joon Lee²,* and Ieck-Chae Euom³

¹ HUNESION Co. Ltd., Seoul 06072, Korea; dsjung@hunesion.com
² School of Business Administration, Chonnam National University, Gwangju 61186, Korea
³ System Security Research Center, Chonnam National University, Gwangju 61186, Korea; iceuom@jnu.ac.kr

* Correspondence: s-lee@jnu.ac.kr

Received: 2 August 2020; Accepted: 14 September 2020; Published: 30 September 2020

Abstract: Malicious codes may cause virus infections or threats of ransomware through symmetric encryption. Moreover, various bypassing techniques such as steganography, which refers to the hiding of malicious code in image files, have been devised. Unknown or new malware hidden in an image file in the form of malicious code is difficult to detect using most representative reputation- or signature-based antivirus methods. In this paper, we propose the use of ImageDetox method to neutralize malicious code hidden in an image file even in the absence of any prior information regarding the signatures or characteristics of the code. This method is composed of four modules: image file extraction, image file format analysis, image file conversion, and the convergence of image file management modules. To demonstrate the effectiveness of the proposed method, 30 image files with hidden malicious codes were used in an experiment. The malicious codes were selected from 48,220 recent malicious codes purchased from VirusTotal (a commercial application programming interface (API)). The experimental results showed that the detection rate of viruses was remarkably reduced. In addition, image files from which the hidden malicious code had previously been removed using a nonlinear transfer function maintained nearly the same quality as that of the original image; in particular, the difference could not be distinguished by the naked eye. The proposed method can also be utilized to prevent security threats resulting from the concealment of confidential information in image files with the aim of leaking such threats.

Keywords: malicious code image; neutralization; steganography; antivirus; image format conversion; nonlinear transfer function

1. Introduction

According to the global vaccine research group AV-Test, 350,000 new malicious codes emerge each day. The organization found the number of malicious codes to have rapidly increased from 4.7 million in 2015 to 9.42 million in 2019 [1]. Notably, network separation technology has been developed to address increasingly intelligent cyber-attacks and the sudden increase in security incidents.

Network separation technology entails an environment in which business networks and Internet networks are separated to prevent attacks through the Internet and to prevent major leaks of internal information. In addition to general business networks, control networks, defense networks, Closed Circuit Television (CCTV) control centers, manufacturing facilities, and other networks can employ a dedicated network connected to the Internet that is separate from their own networks [2].

In an environment with network separation, because the Internet connection is blocked at the source, secure USBs and other measures should be used to exchange data with external entities. However, user inconvenience or the introduction of malicious code through a secure USB has resulted in internal information being continually leaked. Attempts to solve these problems have led to the...
emergence of an inter-network data transfer solution that enables the secure transmission of user PC data and a server stream between the separated areas (secure and non-secure areas) according to the specified security policy [3–5].

Inter-network data transfer requires a malicious code inspection to be conducted according to the security policy. In this regard, conventional antivirus solutions use the reputations or signatures based on well-known information. Content disarm and reconstruction technology can be employed to remove active content such as macros and scripts, e.g., JavaScript, from document-type files [6]. In addition, the leakage of confidential documents or personal information can be prevented using data loss prevention (DLP) and personal information detection technology. More recently, unknown malicious files have also been categorized through machine learning.

Malicious code causes viral infections or threats of ransomware owing to the use of symmetric keys. Although various solutions such as antivirus and advanced persistent threat (APT) have been released to detect known malicious codes, the detection rate of unknown malicious codes is still insufficient. Recent reports on the detection of unknown malicious codes have proposed non-signature-type malicious code detection techniques that employ machine learning based on features extracted from executable files (in portable executable format). These techniques are aimed at addressing the limitation of commercial antivirus solutions that depend on signatures. However, the error rate has continued to be non-negligible owing to the characteristics of machine learning; in particular, an image file containing malicious code has an extremely low detection rate, and complementary methods are required to solve this problem [7].

Various bypass techniques that hide malicious code in non-executable files, such as BMP, JPG, and PNG files, have been studied. Among files that are transferred into and out of a network separation environment, global virus analysis services, such as VirusTotal, have found several cases of image files containing malicious scripts [8]. Known malicious code hidden in image files can be detected by antivirus software based on reputation and signature. However, image files that contain hidden malicious codes cannot be detected by antivirus technology because neither the reputation information nor the signature is available.

In particular, a steganography method (e.g., Stegosploit and Shellcode hiding) can be exploited to intentionally leak information by hiding and spreading malicious code or confidential information through an image file [9,10]. Moreover, it is extremely difficult to detect steganography attacks that use various hidden algorithms using existing analysis techniques. This has led to the development of methods to prevent information from being hidden by a random reprocessing of the group index of the entire image. An alternative approach involves a comparison of steganography encoding and decoding results. Nevertheless, a new approach is needed to solve these problems because of the limitations of existing methods with regard to determining whether the information is hidden.

In this paper, we propose a method for analyzing the structure corresponding to the original image file format and converting the image data area of an image file using a nonlinear transfer function, even in the absence of prior information such as the reputation or signature of the malicious code. This process follows the design of ImageDetox, which neutralizes malicious code hidden in an image file. ImageDetox was subsequently implemented and its effectiveness was experimentally verified and evaluated.

The remainder of this paper is structured as follows. Section 2 describes in detail the techniques that are employed to hide malicious codes within image files, discriminant techniques that can identify such codes, and research that has led to solutions for proactively preventing the concealment of information. Section 3 elucidates the structures of the image file formats and presents an analysis of the types of malicious code hidden within the image files. Section 4 suggests a method for using a nonlinear function and transformations based on the region of the image file format to neutralize malicious codes hidden in the image file. Section 5 presents the ImageDetox system and its operation, which is based on the method applied to neutralize malicious code described in Section 4. Section 6 details the experiments conducted to neutralize malicious codes using the proposed system, as well
as an analysis and a discussion of the results. Finally, Section 7 provides some concluding remarks summarizing the present study and briefly discusses the implications of our findings.

2. Related Work

In this section, we examine techniques, which have been gradually evolving, used to hide malicious codes in image files. In addition, previously proposed techniques for identifying hidden malicious codes and techniques that proactively prevent such codes are also considered.

2.1. Methods for Hiding Malicious Code in an Image File

Figure 1 shows malicious code data that are hidden in an image file. Image files (BMP, JPG, PNG, etc.) usually have a header at the beginning of the file, followed by the actual data. The image format is normally configured using this image file structure. Hiding techniques, such as those that add a malicious binary code or malicious code script at the end of the image data, insert a malicious code script into the additional information area of the image file format, or hide a malicious code in the image data area, are known to exist [11–13].

![Image Data](image.png)

**Figure 1.** Malicious code hidden in an image data.

Keum et al. and Lee et al. investigated the use of a hiding image shellcode technique [14]. In these studies, shellcodes were hidden in 24 bits of BMP images to verify that they would not be detected by existing malicious code detection techniques. An image decoder repository and three modules (scanning, decision, and hiding) were configured to apply this technique. The sequence of operations was to add a decoder from the decoder repository using a 24-bit image, scanning the image, and repeating to determine if there is an insertable decoder by communicating with the decision module, and creating a dummy or jump code. Because an image generated in this way is difficult to distinguish from the original image and a signature does not exist, it is not detectable using signature-based methods. Moreover, an emulation detection method has a long emulation time, making real-time detection difficult [15].

The Stegosploit (a term combining steganography and exploit) technology was recently developed. As shown in Figure 2, when executing an image file in which steganography was used, a script hidden within the image is run, and this enables various exploit attacks to be attempted [16]. Various types of damage can be incurred, depending on the type of exploit hidden in the script and the user’s environment. From a network traffic perspective, Stegosploit is simply an image file, but the script is hidden in pixels, and it is difficult to distinguish whether it is harmful in appearance. Stegosploit has a feature in that the inserted script is executed just by viewing the image.
Steganography technology has recently evolved into intelligent attacks that apply to various protocols such as CCTVs, smart TVs, and IoT devices [17]. These attacks are becoming a threat and can hide malicious or confidential information in files such as image, audio, and video files.

2.2. Methods to Detect and Prevent Malicious Code from Being Hidden in an Image File

Common techniques for distinguishing malicious code hidden in an image file rely on an encoding method to directly analyze the stego images in which information was skillfully hidden in the media data. A method was recently proposed to simultaneously distinguish both decoding and hiding. This was achieved by restoring the hidden image information in a random type chosen from the encoding library to determine whether it is correctly restored. This method can automatically detect encoded information hidden through image steganography by using trained distinguishing techniques, such as those based on the correlation between the entropy characteristics of stego images and the dispersion of pixel values, the distribution of DCT (discrete cosine convert) coefficients, and the dispersion characteristics of the images [15,18].

Techniques that prevent malicious code from being hidden in an image file include a method to randomly mix and reprocess the image indexes [19]. As shown in Figure 3, this method divides the entire image into 16 groups and stores the reprocessed image indexes. This method does not give specific rules to the distribution of contrast, and hides information in online images. It can deal with the threat of information hiding that can be extracted later.

After malicious code has been inserted into the image, the spreadability effect of pixel values are usually observed. In this case, 5–10% of the extra code of the original image is hidden [20]. If the message has sufficient capacity, the presence of hidden codes can be detected by using a chi-square test, which is a statistical technique. Moreover, when the amount of hidden data is extremely small compared to the original image, it is possible to detect the location of the hidden data by employing a chi-square test using the pixel values of existing adjacent two pixel.

Blind detection techniques have been employed to visually, structurally, and statistically analyze files to detect the presence of steganography or malicious codes [21]. These techniques have also been studied to detect the presence of malicious codes through signature search methods; methods for analyzing key information, such as the file registry data; and heuristic methods [6,13,22]. However,
it is difficult to use traditional antivirus software with a commercial tool or service that can detect steganography [23], and conventional detection methods are problematic in that they either cannot detect malicious codes or they detect the wrong malicious codes [24].

Several suggestions were made with regard to the detection of image files containing malicious codes inserted through steganography at the RSA conference [25]. The consensus was that removing hidden areas or removing or replacing redundant data is more effective than attempting to detect malicious codes; in particular, it was noted that simply re-writing images to eliminate cross-site attacks was not effective [26]. By simply converting the image file format, the steganography malware characteristic values are not removed and transferred. A website provided by one of the presenters enables the presence of steganography to be distinguished with pre- and post-variation of the RGB values [27].

3. Malicious Code Hidden in an Image File

The analysis of types of hidden malicious code is aimed at identifying the area in which malicious code is inserted in the image file structure. A normal image file is configured with three areas: header information, additional information, and actual image data. The structures of image files containing malicious code are shown in Figure 4. Methods whereby malicious codes are hidden in an image with a normal file format structure are designed to insert or modulate these three areas and can be classified into several types.

![Figure 4. Types of malicious code hidden in an image file.](image)
In the first type (a), the image format is normally configured, although the malicious code is added at the end of the image data in the form of a binary, including the PE, Dynamic Linking Library (DLL), and Executable and Linkable Format (ELF) formats; furthermore, this type is commonly employed to insert malicious script and confidential information (e.g., corporate information, personal information). This approach exploits the fact that image viewer applications process only the end of the image (EOI) and ignore the malicious binary code. If the malicious binary code is a Drive by Download (DBD) type, it can be disguised as an image data at the time of download and flow into the main memory. Ransomware generates a symmetric key using DBD and causes damage when encrypting the victim’s data.

The second type (b) presents only the file identification signature for each image type, and the remaining area contains a malicious script written in JavaScript, HTML, and PHP, among other forms. Because this exploits the fact that several applications judge multipurpose Internet mail extension (MIME) types by using only the header information of the file, these malicious codes are neither detected nor blocked in the case of image files that permit the introduction of MIME types.

The third type (c) inserts a malicious script into the additional information area of the image file format; moreover, it can also hide the malicious code in the area containing the image data (the actual image pixel information). The use of various encryption methods and obfuscation algorithms to hide the malicious code or the use of steganography algorithms in the malicious script poses a highly difficult challenge. In such cases, the inserted code could bypass signature-based antivirus or reputation-based detection techniques, and it would also be difficult to detect and analyze such code using machine-learning-based techniques. Similarly, it is extremely difficult to detect and analyze malicious codes when confidential information is hidden in the pixel information within the image data areas exploited by various steganography algorithms or tools.

4. Methods for Neutralizing Hidden Malicious Code

As previously reported, existing detection methods to detect malicious codes hidden in image files cannot avoid false positives or false negatives. Thus, it is more effective to either remove or replace hidden data without relying on detection. In this paper, a method is proposed that eliminates malicious code without the loss of image quality by using a nonlinear transfer function during the conversion of the three structural areas in which the malicious code is configured and hidden in an image file.

4.1. Image Conversion

Figure 5 presents a method to convert the areas of an image by using a file extraction step and a format analysis step. The image header information conversion step (TF1) changes the identification signature of the converted image format, and the additional image conversion step (TF2) applies a specific string filtering conversion method. The image pixel data conversion step (TF3) applies a nonlinear transfer function with a specific range value to convert the attribute value of the original image.

First, the area containing the image header information changes to the identification signature of the converted image format upon the application of TF1 [28]. This has the effect that the content related to the malicious code inserted into the EOI is automatically removed by converting only the data preceding the EOI in the file format conversion process. Then, TF2 is employed within the area that contains additional information regarding the image in order to convert the string associated with the malicious code script into a specific value. For example, the keywords (html, head, script, type, and so on) related to JavaScript, PHP, and HTML can be changed to copy their value to 0x00, which does not affect their original size. This has the effect of preventing the malicious code script from working properly. Because the area containing additional information in the image file does not affect the image itself, there is no damage to the image even if additional information changes to a certain value. If there was a keyword used to induce malicious behavior in the additional information area, the malicious keywords would be changed into “0x00”, which would prevent malicious behavior from
being executed. However, if the original value of the additional information is partially lost, and this additional information can be inferred or restored if its history of change is stored in the system.

![Figure 5. Conversion function for each area of the image format.](image)

### 4.2. Image Transformation with Nonlinear Transfer Function

TF3 in Figure 5 is applied to the area containing the image pixel data to neutralize the hidden malicious code by converting the RGB value ([RGB]in) of one pixel of the original image into a converted pixel RGB value ([RGB]out) corresponding to the same location by using the nonlinear transfer function. This function is defined in Equation (1).

\[
[RGB]_{out} = (W \cdot [RGB]_{in}^{1/\gamma}),
\]

where \( \gamma \) denotes a value within a specific range (0.950 < \( \gamma \) < 0.995 or 1.005 < \( \gamma \) < 1.050) characterizing a nonlinear transfer function, and \( W \) represents the application of an alpha channel.

Human vision reacts nonlinearly according to Weber’s law. Image editing tools such as Photoshop use a gamma (\( \gamma \)) correction to convert into the optimal image quality, and the nonlinear transfer function is used for gamma (\( \gamma \)) correction. If the gamma value is too low or too high in terms of range, the value of the image data area is severely modified, and the quality of the image is degraded. In our experiment, we found that if the gamma value is close to 1, there is little change in the pixel value, and thus the malicious code contained in the image continues to exist. Thus, the range of gamma values was set to a range (0.950 < \( \gamma \) 0.995 or 1.005 < \( \gamma \) 1.050).

The value of \( W \) is calculated only in image format (GIF, PNG, etc.) when an alpha channel is applied. By limiting the value of \( \gamma \) to a specific range of values in the nonlinear transfer function (Equation (1)), the calculated conversion value is limited such that the least significant bit (LSB) of each pixel only changes from less than 1 to 4 bits. That is, the calculation with a limited range of values enables the quality to be maintained such that it is nearly similar to that of the original image; that is, the difference in quality is difficult to distinguish by the naked eye. Even if malicious codes were to be inserted or leaked information was to be hidden in the file, the attribute value would be changed, and the code would be neutralized. The range of \( \gamma \) values in this study was selected as a suitable pixel range wherein the differences among the images cannot be distinguished by the naked eye when observing a variety of target images.

Figure 6 shows an example of the change in the RGB colors of an image after application of the nonlinear transfer function. In this example, the value of one pixel, [RGB]in, of the original image is (205, 107, 66), and the value of the pixel [RGB]out, as converted using the nonlinear transfer function in Equation (1) is calculated as (212, 110, 68). A comparison between the color corresponding to the pixel
value of the original image and the color corresponding to that of the converted image shows that the converted pixel is actually indistinguishable from the original pixel.

Figure 6. RGB color change of image applied for nonlinear transfer function

5. Implementation

In this section, we describe the conceptual structure of the ImageDetox system, as shown in Figure 7. This system has the function of neutralizing hidden malicious code or information and utilizes the methods proposed in this paper.

The system is configured with four modules: an image file extraction module, an image file format analysis module, an image file conversion module, and a converted image file management module. Figure 8 shows a flowchart of these processes.

The original image file extraction step extracts only image files from among the various files introduced into the inter-network data transfer section. It then analyzes whether the data structure of the image file corresponds to the reference format for each image of the original image file extracted during the analysis step of the image file format structure. If the image file format is judged to have a normal configuration, the hash value of the original image file is calculated. The system then verifies whether the file information with the same hash value is stored in the converted image file management module. The image file conversion step converts each area of information of the original image file, as shown in Figure 5, and the converted image file is saved in the storage of the converted image file. The image file saving and periodic update step is used to store the file based on the creation time of the stored image file for a certain period of time. Image files that have been stored longer than the specified period are deleted, and this process is periodically repeated.
The image file extraction module is used to extract image files (JPEG, GIF, PNG, BMP, etc.) or image object linking and embeddings (OLEs) from various document files (doc, ppt, xls, hwp, odp, etc.) that are introduced through an inter-network data transfer. In this case, the identification signature of the image file format is determined and extracted from the extension or file header information of each file. This enables image files to be distinguished from all files that were transferred.

Figure 6. RGB color change of image applied for nonlinear transfer function

5. Implementation

In this section, we describe the conceptual structure of the ImageDetox system, as shown in Figure 7. This system has the function of neutralizing hidden malicious code or information and utilizes the methods proposed in this paper.

Figure 7. Conceptual structure of ImageDetox system.

The system is configured with four modules: an image file extraction module, an image file format analysis module, an image file conversion module, and a converged image file management module. Figure 8 shows a flowchart of these processes.

Figure 8. Flowchart of the process used by ImageDetox to neutralize hidden malicious code.

To analyze the structure of the image file format extracted from the aforementioned image file extraction module, the image file format analysis module first identifies the type of image (JPEG, GIF, PNG, BMP, and so on) from the file header information. The module then transfers this information to the image file conversion unit by determining whether the format structure of an image file has three components (the file identifier of the image format, the additional image information, and the image pixel data area) as per the corresponding standard image file structure.

The image file format conversion module applies each hiding technique by converting the image file area shown in Figure 5, as proposed in Section 4.1. First, with regard to the image header information, the file identifier information of the original image header is replaced with that of the conversion image header. The original additional image information, except for a specific string, is then copied into the additional information area of the conversion image. With regard to the image pixel data, the attribute value of the original image is converted by applying the nonlinear transfer function within a specific range to neutralize the characteristics of malicious codes or hidden information. The nonlinear transfer function is defined and used as per Equation (1) of Section 4.2.

The conversion image file management module saves the hash value information of the original image file in the converted image file once it is stored in the management unit for a period of time. This is necessary to resolve the reduction in the processing performance resulting from the repetitive process of conversion for the same original image file. The hash value information of image files that have surpassed a certain storage period is deleted from storage, and the information is updated periodically.

6. Evaluation

6.1. Experimental Setup

In this study, we used 48,220 of the latest malicious codes purchased from VirusTotal (commercial API). VirusTotal has been incorporated as a subsidiary of Google and has cooperated with global antivirus companies to share malicious code information. Specifically, it is a commercial cloud-based...
service to which billions of samples, including malicious codes, URLs, and packet captures (PCAPs) are uploaded by general users worldwide. These samples are inspected using the engines of approximately 69 antivirus products, and the results are provided in real time.

Among the aforesaid samples, 30 image files containing hidden malicious codes were extracted to measure the detection rate by VirusTotal. In addition, 30 self-produced steganography images that were created using well-known malicious codes were utilized in the same experiment. The number of samples is limited because the vulnerabilities in applications are infrequently encountered. However, as found in previous studies, the risk of hidden malicious codes in the continuously increasing number of image files is extremely high in terms of the impact of the exploiting attack or the potential risk rather than with respect to the sample being small.

6.2. Results and Analysis

6.2.1. Result of Neutralizing Hidden Malicious Code in Image Files

The image format cannot be converted during the process used to neutralize the image file malware and hidden information. A malicious code can be neutralized by replacing it with an alternative image (self-produced).

In addition, TF2 in Section 4.2 was applied to the malicious script in the additional image information area. The neutralization experiment then verified that the file was judged as a normal file by VirusTotal, thereby confirming that the malicious code hidden in the original file had been removed. The results of this experiment are shown in Figure 9 (before) and Figure 10 (after).

![Figure 9: Example file with malware in additional image information for TF2 (detected in red).](image-url)
When a malicious binary code and a malicious code script are added at the end of the image data, the image file format conversion process converts only the EOI to automatically remove any content related to the malicious code inserted at the EOI. The results of this experiment are shown in Figure 11 (before) and Figure 12 (after). The figure shows the results of a neutralization experiment of a structure file, build.png, with a PE malicious code place at the EOI; the figure indicates that the PE malicious code was removed and that the file was judged by VirusTotal to be clean without the presence of malware.

Figure 10. Results of neutralizing a file with malware in additional image information by TF2.
(a) Binary code of an image with malware in pixel data area.

(b) Detection of malware hidden in pixel data area.

Figure 11. Example file with malware in the image pixel data area for TF3 (detected in red).
Next, an image file containing hidden malicious code that was inserted using the OpenStego tool (open steganography program) was examined [29]. The file was converted into RGB values ([RGB]out) by applying the gamma value to the previously described nonlinear transfer function, and the OpenStego tool was used to check whether the file could be decoded again. However, decoding was impossible, as shown in Figure 13, verifying that the malicious code was neutralized.
The open steganography software, OpenStego is a well-known tool and has been cited in many research papers [30,31]. OpenStego hides message files, malicious files, confidential information, and other data in the image pixel area of the original image by converting the pixel data values through steganography algorithms. This tool can extract hidden files or information through the steganography algorithm used in encoding. In this study, by determining the gamma value of the nonlinear transfer function, the image quality was verified using OpenStego. Decoding should not be possible, as shown in Figure 13c. If decoding is possible, as shown in Figure 14, a malicious code.

Figure 13. (a) Hide malicious code in image data by OpenStego. (b) Results of neutralizing steganographic malicious code in an image data. (c) Decoding impossible by neutralizing steganographic malicious code.
The open steganography software, OpenStego is a well-known tool and has been cited in many research papers [30,31]. OpenStego hides message files, malicious files, confidential information, and other data in the image pixel area of the original image by converting the pixel data values through steganography algorithms. This tool can extract hidden files or information through the steganography algorithm used in encoding. In this study, by determining the gamma value of the nonlinear transfer function, the image quality was verified using OpenStego. Decoding should not be possible, as shown in Figure 13c. If decoding is possible, as shown in Figure 14, a malicious code can be executed again. This case occurred when the value of $\gamma$ in the nonlinear transfer function applied to TF3 was not within the range $0.950 < \gamma < 0.995$ or $1.005 < \gamma < 1.050$.

The file in which the malicious code was hidden using the OpenStego tool was again examined, but this time using the antivirus engine of VirusTotal [32]. Notably, the antivirus engine did not succeed in detecting the malicious code in the image and could not identify the file as containing a hidden malicious code.

### 6.2.2. Validation Analysis of Neutralization Result

Table 1 compares the results of malicious code detection utilizing VirusTotal before and after application of the proposed neutralization method to 30 image files that contained a hidden malicious code. VT Detection (A) represents the results of analyzing whether each original malicious image file is malicious as determined by VirusTotal, and VT Detection (B) represents the results of analyzing whether it is malicious after applying the method proposed in this paper. The value of VT Detection (A) is the number of antivirus programs that was able to detect malware in an image file divided by the number of antivirus programs used to detect viruses in an image file. The VT Detection (A) results show that, on average, only 45.4% of the 30 image files in which malicious code was hidden were detected by the antivirus engine, indicating its limited ability to detect malicious code hidden in an image. The denominator (59, 60, 59, 59, … ) is the number of antivirus programs inspected by the VT service. The number of antivirus programs used for each service request varies from time to time. Thus, the number of antivirus programs engines are slightly difference. Here, the numerator (30, 15, 27, 27, … ) indicates the number of vaccines that were determined to be malignant during the test.

The VT Detection (B) result is the detection result after the proposed neutralization technique was applied to the files containing the hidden malicious code. During this experiment, VirusTotal was used to re-verify that the malicious code had been removed after the original image files (GIF) were converted into other data files (JPG). For example, the Trojan/bgcolor.gif file contained a hidden malicious code in the form of an “iframe” tag. When the corresponding file was tested using VirusTotal, 27 of 59 detection engines determined it to be a malicious file. After our proposed system was used to delete the string value in the additional area of the image file by changing the value of textString HEX
Importantly, even when certain codes are perceived to be malicious by a small number of antivirus engines in VirusTotal, such codes would not be able to perform their normal operations, either because they are incorrectly detected by a signature-based detection engine or because the parts that would enable the malicious code to execute have been removed.

7. Conclusions

In this study, we investigated the structure of an image file, analyzed the malicious code hidden in the image file, and proposed a technique to neutralize the malicious code. No attempt was made to analyze or detect the malicious code hidden in the original image file; instead, we analyzed the structure of the image file format and used a nonlinear transmission function to convert the pixels to neutralize the operation of the malicious code. We presented a method to convert the areas of an image by using a file extraction step and a format analysis step. The image header information conversion step (TF1) changes the identification signature of the converted image format, and the image additional conversion step (TF2) applies a specific string filtering conversion method. The image pixel data conversion step (TF3) applies a nonlinear transfer function with a specific range of values to convert the attribute value of the original image.

We configured four modules (an image file extraction module, image file format analysis module, image file conversion module, and a converged image file management module) and presented the
process used by ImageDetox to neutralize a hidden malicious code. The ImageDetox system proposed in this paper was evaluated experimentally to assess their effectiveness and the resultant image quality. As a major advantage of the proposed method, it has the effect of neutralizing the behavior of a malicious code in advance without any prior information on the signature or the characteristics of the code, whether known or unknown. The quality of the image file that was produced through the conversion of the original image file to neutralize the malicious code was similar to that of the original image, such that the difference could not be distinguished by the naked eye. In addition, the proposed method can also be utilized to prevent security threats resulting from the concealment of confidential information in image files with the aim of leaking such threats.

**Author Contributions:** Conceptualization, D.-S.J. and S.-J.L.; methodology, D.-S.J. and S.-J.L.; software, D.-S.J.; validation, D.-S.J., S.-J.L. and I.-C.E.; formal analysis, D.-S.J.; investigation, S.-J.L.; resources, D.-S.J.; data curation, D.-S.J.; writing—original draft preparation, D.-S.J.; writing—review and editing, S.-J.L. and D.-S.J.; visualization, D.-S.J.; supervision, I.-C.E.; project administration, S.-J.L.; funding acquisition, S.-J.L. and I.-C.E. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was supported by an Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT) (No. 2019-0-01343).

**Conflicts of Interest:** The authors declare no conflict of interest.

**References**

1. AVTEST Malware Statistics. 2019. Available online: [http://www.av-test.org/en/statistics/malware/html](http://www.av-test.org/en/statistics/malware/html) (accessed on 1 September 2020).
2. Financial Supervisory Service. *Financial Information Network Separation Guide*; Financial Supervisory Service: Seoul, Korea, 2013.
3. Hunesion i-oneNet. 2012. Available online: [http://www.hunesion.com/?page_id=3192](http://www.hunesion.com/?page_id=3192) (accessed on 1 September 2020).
4. ITSCC Common Criteria Certification. *Korean National Protection Profile*. 2012. Available online: [http://www.itssc.kr](http://www.itssc.kr) (accessed on 1 September 2020).
5. Lee, H.J.; Cho, D.I.; Kou, K.S. A Study of Unidirectional Data Transmission System Security Model for Secure Data transmission in Separated Network. *Asia-Pac. J. Multimed. Serv. Converg. ArtHumanit. Sociol.* 2015, 5, 539–547. [CrossRef]
6. OPSWAT. Image-Borne Malware. 2016. Available online: [https://www.opswat.com/blog/image-borne-malware-how-viewing-image-can-infect-device](https://www.opswat.com/blog/image-borne-malware-how-viewing-image-can-infect-device) (accessed on 1 September 2020).
7. Jeon, D.J.; Park, D.G. Real-time Malware Detection Method Using Machine Learning. *J. Korean Inst. Inf. Technol.* 2018, 16, 1598–8619. [CrossRef]
8. Jung, D.S.; Lee, S.J.; Ryu, D.J. A study on the correspondence malicious traffic between the network data transfer systems in a network isolation environment. *Winter Proc. J. Commun. Netw.* 2016, 1152–1153.
9. Threatpost. Stealthy Malware Hidden in Images Takes to GoogleUserContent. 2018. Available online: [https://threatpost.com/stealthy-malware-hidden-in-images-takes-to-googleusercontent/13418](https://threatpost.com/stealthy-malware-hidden-in-images-takes-to-googleusercontent/13418) (accessed on 1 September 2020).
10. Virusbulletin. How It Works: Steganography Hides Malware in Image Files. 2016. Available online: [https://www.virusbulletin.com/virusbulletin/2016/04/how-it-works-steganography-hides-malware-image-files](https://www.virusbulletin.com/virusbulletin/2016/04/how-it-works-steganography-hides-malware-image-files) (accessed on 1 September 2020).
11. Suarez-Tangil, G.; Tapiador, J.E.; Peris-Lopez, P. Stegomalware: Playing Hide and Seek with Malicious Components in Smartphone Apps. In *Lecture Notes in Computer Science*; Springer: Berlin/Heidelberg, Germany, 2015; pp. 496–515, LNCS 8957.
12. Kaspersky. PNG Embedded—Malicious Payload Hidden in a PNG File. 2016. Available online: [https://securelist.com/png-embedded-malicious-payload-hidden-in-a-png-file/74297](https://securelist.com/png-embedded-malicious-payload-hidden-in-a-png-file/74297) (accessed on 1 September 2020).
13. Park, I.H.; Seoung, W.S. A Study on Concealing Malicious Code Using a Image File: Method and Countermeasure. *J. Inst. Electron. Eng. Korea* 2009, 2009, 235–236.
14. Keum, Y.J.; Choi, H.J.; Kim, H.K. Hiding Shellcode in the 24Bit BMP Image. *J. Korea Inst. Inf. Secur. Cryptol.* 2012, 22, 691–705.
15. Lee, J.H.; Kim, C.L.; Lee, S.H.; Park, J.I. Image steganography and its discrimination. J. Broadcast. Eng. 2018, 23, 462–473.
16. Saumil, S. Stegosploit: Hacking with pictures. In Proceedings of the 6th Annual HITB Security Conference, Amsterdam, The Netherlands, 26–29 May 2015.
17. Cabaj, K.; Caviglione, L.; Mazurczyk, W.; Wendzel, S.; Woodward, A.; Zander, S. The New Threats of Information Hiding: The Road Ahead. IT Prof. 2018, 20, 31–39. [CrossRef]
18. Rama, P.; Sahoo, P.K. Scanning Tool for Identification of Image with Malware. Int. J. Adv. Comput. Tech. Appl. 2016, 4, 170–175.
19. Subhedar, M.S.; Mankar, V.M. Current status and key issues in image steganography: A survey. Comput. Sci. Rev. 2014, 13, 95–113. [CrossRef]
20. Ji, S.S. Locating and Searching Hidden Messages in Stego-Images. J. Korea Ind. Inf. Syst. Soc. 2009, 14, 37–43.
21. Wingate, J.E.; Watt, G.D.; Kurtz, M.; Davis, C.W.; Lipscomb, R. Defending against insider use of digital steganography. In Proceedings of the Conference on Digital Forensics, Security and Law, Arlington, VA, USA, 18–20 April 2007; pp. 175–184.
22. Park, B.H.; Kim, D.Y.; Shin, D.C. A Study on a Method Protecting a Secure Network against a Hidden Malicious Code in the Image. Indian J. Sci. Technol. 2015, 8, 26. [CrossRef]
23. George, G.; Savaridassan, P.; Devi, K. Detect Images Embedded with Malicious Programs. Int. J. Pure Appl. Math. 2018, 120, 2763–2777.
24. Karampidis, K.; Kavallieratou, E.; Papadourakis, G. A review of image steganalysis techniques for digital forensics. J. Inf. Secur. Appl. 2018, 40, 217–235. [CrossRef]
25. Wiseman, S.R. Poison Pixels—Combatting Image Steganography in Cybercrime. In Proceedings of the RSA Conference, San Francisco, CA, USA, 16–20 April 2018. HTA-W02.
26. Saurabh, C.; Amritha, P.P.; Sethumadhavan, M. Stegware Destruction Using Showering Methods. Int. J. Innov. Technol. Explor. 2019, 8, 256–259.
27. Wiseman, S.R. Steg Analysis. 2020. Available online: https://rsa2018.deep-secure.com/poisonPixels/poisonPixels.html (accessed on 1 September 2020).
28. File Signatures. Public Database of File Signatures. 2020. Available online: https://filesignatures.net/ (accessed on 1 September 2020).
29. OpenStego. Free Steganography Solution. Available online: https://sourceforge.net/projects/openstego/ (accessed on 1 September 2020).
30. Lin, X. Steganography and Steganalysis. In Introductory Computer Forensics; Springer: Berlin/Heidelberg, Germany, 2018. [CrossRef]
31. Al-Sanjary, O.I.; Ahmed Ibrahim, O.; Sathasivem, K. A New Approach to Optimum Steganographic Algorithm for Secure Image. In Proceedings of the 2020 IEEE International Conference on Automatic Control and Intelligent Systems (I2CACIS), Shah Alam, Malaysia, 20 June 2020; pp. 97–102. [CrossRef]
32. VirusTotal. Premium Service. 2019. Available online: https://support.virustotal.com/hc/en-us/articles/115003886005-Private-Services (accessed on 1 September 2020).

© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).