www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 2 Issue 5 May, 2013 Page No. 1479-1486

TECHNIQUES OF DEFEATING STEGANOGRAPHY: A STATE OF ART SURVEY

Vishwajeet Singh[1], Mohd Aman[2], Vaibhav Gupta[3], Mayank Parashar[4]

[1,2,3] U.G., Dronacharya group of institution, greater Noida, U.P.(INDIA)

Vishwajeet2010@yahoo.com, vaibhavgupta6070@gmail.com

ABSTRACT

Steganography is a useful tool that allows covert transmission of information over an overt communications Channel. Combining covert channel exploitation with the encryption methods of substitution ciphers and/or one time pad cryptography, Steganography enables the user to transmit information masked inside of a file in plain view. The hidden data is both difficult to detect and when combined with known encryption algorithms, equally difficult to decipher.
This paper discusses the functional areas in the field of Steganography, how Steganography works, what Steganography software is commercially available and what data types are supported, what methods and automated tools are available to aide computer forensic investigators and information security professionals in detecting the use of Steganography , after detection has occurred, can the embedded message be reliably extracted, can the embedded data be separated from the carrier revealing the original file, and finally, propose some methods to defeat the use of Steganography even if it cannot be reliably detected
.

1. INTRODUCTION

Within the field of Computer Forensics, investigators should be Aware that Steganography can be an effective means that enables Conceald data to be transferred inside of seemingly innocuous carrier files . Knowing what software applications are commonly available and how they work gives forensic investigators a greater probability of detecting, recovering, and eventually denying access to the data that mischievous individuals and programs are openly concealing. Generally speaking, Steganography brings science to the art of hiding information. The purpose of Steganography is to convey a message inside of a conduit of misrepresentation such that the existence of the message is both hidden and difficult to recover when discovered. The word Steganography comes from two roots in the Greek language, “Stegos” meaning hidden / covered / or roof, and “Graphia” simply meaning writing.

Similar in nature to the sleight of hand used in traditional magic, Steganography uses the illusion of normality to mask the existence of covert activity. The illusion is manifested through the use of a myriad of forms including written documents, photographs, paintings, music, sounds, physical items, and even the human body. Two parts of the system are required to accomplish the objective, successful masking of the message and keeping the key to its location and/or deciphering a secret. When categorized within one of the two fundamental security mechanisms of computer science (cryptographic protocols and maintaining control of the CPUs instruction pointer), steganography clearly fits within cryptography. It closely mirrors common cryptographic protocols in that the embedded information is revealed in much the same manner as substitution or Bacon cipher mechanism.

This paper will highlight some historical examples, discuss the Basic principles of steganography showing how most instances work, identify software that can be used for this purpose, and finally provide an overview of current methods employed to detect and defeat it.

2. HISTORICAL EXAMPLES

Hiding messages by masking their existence is nothing new. Classical examples include a Roman general that shaved the head of a slave tattooing a message on his scalp. When the slave's hair grew back, the General dispatched the slave to deliver the hidden message to its intended recipient.

Ancient Greeks covered tablets with wax and used them to write on. The tablets were composed of wooden slabs. A layer of melted wax was poured over the wood and allowed to harden as it dried. Hidden messages could be carved into the wood prior to covering the slab. When the melted wax was poured over the slab, the now conceal message was later revealed by the recipient when they re-melted the wax and poured it from the tablet.

From the 1st century through World War II invisible inks were often used to conceal hidden messages. At first, the inks were organic substances that oxidized when heated. The heat reaction revealed the hidden message. As time passed, compounds and substances were chosen based on desirable chemical reactions. When the recipient mixed the compounds used to write the invisible message with a reactive agent, the resulting chemical reaction revealed the hidden data. Today, some commonly used compounds are visible when placed under an ultraviolet light. In another form while Paris was under siege in 1870, messages were sent by carrier pigeon. A Parisian photographer used a microfilm technique to enable each pigeon to carry a higher volume of data.

The miniaturization of information also served to deter detection and was a precursor to the invention of the microdot. A microdot is a document or photograph reduced in size until it is as small as a pencil dot (about the size of the period at the end of this sentence). Between World War I and II Germany used microdots for steganographic messaging purposes and later many countries passed these microdot messages through insecure postal channels. With any type of hidden communication, the security of the message often lies in the secrecy of its existence and/or the secrecy of how to decode it. Cryptography often uses only a worst case approach assuming only one of these two conditions holds.

Given the historical examples above, it should be clear that if a steganographic system's key were to be discovered, the security of the system would be irrevocably broken. Simply shaving the hair off the head of everyone passing through a checkpoint, or melting the wax off of any discovered tablets reveals not only the existence of a hidden message but the message itself.

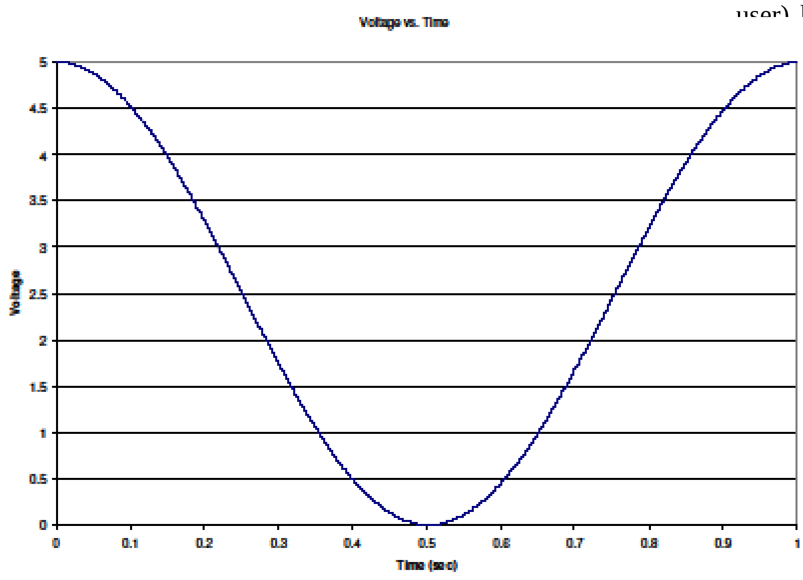
3. FUNCTIONAL OVERVIEW

Focusing the discussion on steganographic techniques used in digital media, traditional methods are employed to modify the data that defines the carrier or cover file. Modifications are made to achieve a desired pattern. The pattern used to modify the carrier defines a bit sequence that contains the hidden message or data. The basic principle of steganography ensures that modifications to the data in the cover file must have insignificant or no impact to the final presentation. Insignificant or no impact onfinal presentation means changes so minor in nature that the casual observer cannot tell that a hidden message is even present.

Every digital file is composed of a sequence of binary digits (0 or 1). It is also a relatively simple task to modify the content of a file by changing a single bit in the sequence. Accomplishing the modification without changing the presentation or the final form of the file is altogether a different task. For example, the binary value of the decimal number 13 consists of 4 bits (1101), changing one bit in the sequence changes the decimal value of the number it represents and ultimately changes the meaning of the value, (i.e. 1100 is the decimal equivalent of the number 12 not 13). What is required for steganography is a data set represented by large numbers of bits per datum. For illustration, an electric signal conducted on a wire can contain varying voltage levels over time. When using a single bit to sample the voltage level, we can only represent two states for any given time interval (off or on -> 0 or 1). We cannot represent a specific value such as +3.3v unless the value happens to be a boundary condition (i.e. the high voltage of this signal is +3.3v). By adding bits to the representation of the measurement we can reproduce measurements between the boundary values. Two bits can define up to four states (0, 1.1, 2.2, and 3.3v for example), three can define eight, four bits define sixteen, and so on. The level of precision used in the measurement is proportional to the number of bits used in the binary representation of the voltage level.

The downfall of using additional bits per datum is seen in the impact to the size of the stored data that represents the measured waveform. When measurements are taken over time intervals, each additional bit multiplies the size of the data file. Depending on the level of fidelity needed for the data representation, additional bits can eventually cease to contribute desirable or distinguishing information such as round off errors (4.999999999999 vs. 4.999999999998). In essence, additional bits can eventually become unnecessary when the accuracy of the waveform has been achieved. A trade off between file size and sample accuracy is often performed and the bit depth (number of bits per sample) chosen based on an amicable medium. Selection of the optimum amount of bits needed to represent the information using the smallest amount of storage space is a goal for many data formats. Using the previous example, sampling voltage level over discrete time intervals, it is also possible to graphically represent the waveform in a voltage vs. time plot. With enough bits to provide fidelity to the measurement, a close approximation of the actual signal can be reproduced. In the **following example, 8** bits will be chosen to represent a value between -5 and +5 volts with the most significant bit determining the sign (+/-) of the measurement. The remaining seven bits provide 128 discrete values for the amplitude of the sampled voltage. Thus, each discrete value is 0.04 volts. Voltage samples of the signal taken 25,000 samples per second produce 25 Kilobytes (200 Kilobits) of data over a one second time interval.

A plot of our hypothetical wave form is displayed in Figure 1. Six randomly selected samples (represented by eight binary digits) are included below simply to illustrate that the binary data changes over the time interval.



01001010, 01001011, 01001100, 01001101, 01001110, 01001111

Figure 1: Example carrier wave form and Binary Data Representing Six Distinct Point.

By modifying the least significant bit of each sample, it is possible to embed information into the waveform without having significant impact to the graphical representation of the data. In the next section the waveform above, and its associated binary data, will become the carrier or cover file for our steganographically embedded covert message.

3.1 Modifying the Carrier

Noting that by using 7 bits to represent 5 volts of amplitude, we create a relatively small division between values (0.04V). By modifying the least significant bit (LSB) of any datum we can only change its reproduced value by the same amount (0.04V). This imperceptible change means that intentional modifications to the LSB of every sample may go unnoticed and allow data to be embedded into the bit sequence. Using sequential data points to carry our message, we can inject a 25,000 bit message into the LSB for every second of data we have recorded. When viewing the waveform after modification, the difference in voltage at any datum is imperceptible to the naked eye. To illustrate 01001010, 01001011, 01001100, 01001101, 01001110, 01001111, 01010000, 01010001 … In the event we wished to inject the 8 bit message (11110000) into the data, we would modify the corresponding LSBs of the above bit stream to match our message. The resulting steganographic data stream would become 01001011, 01001011, 01001101, 01001101, 01001110, 01001110, 01010000, 01010000 … where the modified bits are in blue bold typeset. Note that while the carrier data has changed, what is represented or displayed in the final form (i.e. the form delivered to the end user) has been modified only in an imperceptible manner. hows our example waveform embedded with the ASCII message after conversion to binary: "The set you free". The existence of the embedded an only be seen in the blow-up of the first few the reproduced waveform.

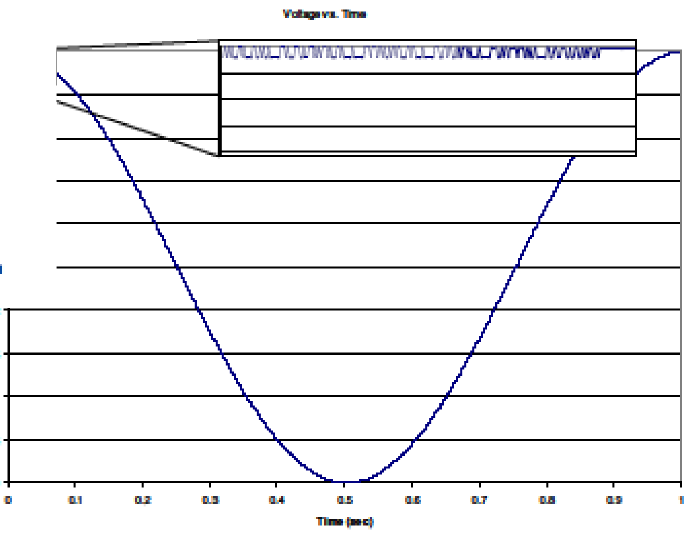


Figure 2- Steganographically Injected Wave Form Blow–up Of the Injected Data Area

3.2 Typical Carrier Data Types

Any file that requires multiple bits to reasonably quantify its message such that minor changes to the data are imperceptible when the file is presented in final form is an acceptable candidate for a carrier. Digital data types fitting this description include image, video and sound files, data can even be embedded in standard TCP/IP packet headers. The most common image formats include BMP, GIF, and JPEG. The majority of software applications designed for steganography utilize the JPEG image file format as the carrier.

3.2.1 Audio Files

The human ear can distinguish frequencies between 20 Hz and 20 kHz. By embedding a stream of data into an audio signal at frequencies above those, the effect is inaudible and cannot be detected by the human ear. Not only does the carrier's reproduction of the data sound identical to that of the original, but the only impact the added data has is an increase to the size of the file. A frequency spectrum analyzer or a calculation of the total amount of data required to produce the same audible spectrum over that time interval would be able to detect the presence of the additional information. LSB modification of the bit stream can also be used but has a noticeable detrimental impact to the carrier file when it is reproduced. Typically the reproduced audio has a higher occurrence and level of what sounds like static or hiss. As displayed in the blow-up of Figure 2, the audio signal can be expected to diversify which results in an increase in the amount and level of background noise.

3.2.2 TCP/IP Headers

Software for audio steganography is also widely available. Formats suited for injection include WAV, PCM, AVI, MIDI, MPEG, MP3, RIFF, and VOC. Finally, data hiding software titles are available to embed information in unused or hidden locations on physical drives. Through manipulation of unused space or hidden directories, data can be stored between files or at any unused area of the file system. The tools take advantage of the slack space often located between the legitimate end of a file and the start of the next cluster.

Hidden directories can be created that are not included in the allocation tables of the main operating system. Files are stored in these directories through a ghost or mirror OS directory structure that is managed by the software. By using areas of the drive unlikely to be accessed by the OS or by marking the sectors as bad or unreadable in the main OS allocation, the steganography software is able to reduce or eliminate the likelihood that the hidden data will be overwritten. By encoding or encrypting the data stored in slack space and hidden directories, this software is also able to reduce the chance that simple file scans will detect or indicate the presence of the hidden data.

The vast majority of the steganography titles incorporate the use of cryptographic protocols such as AES, 3DES, RSA, and Blowfish to either encrypt the hidden message prior to embedding, or use the protocol to randomize the injection sequence for the data. When the file containing the embedded information is provided to the recipient, only the correct password and decoding algorithm will produce the decoding sequence or decrypt the embedded file.

4. COMMERCIALLY AVAILABLE SOFTWARE

Johnson & Johnson technology consultants maintains a website that contains a survey of more than 140 software titles that perform steganography using all of the various types of datafiles discussed earlier. The software includes Freeware, Shareware, and licensed versions for both individual and business users. Various titles on the site will run on Linux variants, Microsoft Windows, and Macintosh computers. Of the more than 140 titles listed on their web site, over half (85) deal with embedding information in still images. 37 of these titles can encode information into BMP files, 20 into GIF, and 15 can embed data in JPEG files. The remaining titles can use any binary input to produce encoded PCX, PICT, and PNG output files.

The second most popular type of steganography software is for plain text and HTML file types. The data is embedded through the use of character spacing, insertion of sequences of tabs and spaces at the end of the lines in the carrier file, or through production of poorly formed English sentences and poetry. Even the web site's descriptions of these types of tools indicate poor grammar and awkward word selection when they write "substitution cipher that makes text files look like a cross between adlibs and bad poetry".

Software for audio steganography is also widely available.Formats suited for injection include WAV, PCM, AVI, MIDI,MPEG, MP3, RIFF, and VOC. Finally, data hiding software titles are available to embed information in unused or hidden locations on physical drives. Through manipulation of unused space or hidden directories, data can be stored between files or at any unused area of the file system. The tools take advantage of the slack space often located between the legitimate end of a file and the start of the next cluster.

Hidden directories can be created that are not included in the allocation tables of the main operating system. Files are stored in these directories through a ghost or mirror OS directory structure that is managed by the software. By using areas of the drive unlikely to be accessed by the OS or by marking the sectors as bad or unreadable in the main OS allocation tables, the steganography software is able to reduce or eliminate the likelihood that the hidden data will be overwritten. By encoding or encrypting the data stored in slack space and hidden directories, this software is also able to reduce the chance that simple file scans will detect or indicate the presence of the hidden data.

The vast majority of the steganography titles incorporate the use of cryptographic protocols such as AES, 3DES, RSA, and Blowfish to either encrypt the hidden message prior to embedding, or use the protocol to randomize the injection sequence for the data. When the file containing the embedded information is provided to the recipient, only the correct password and decoding algorithm will produce the decoding sequence or decrypt the embedded file.

5. DETECTION AND RECOVERY METHODS

Steganalysis is the art and science behind the detection of the use of steganography by a third party. The basic function of steganalysis is to first detect or estimate the probability that hidden information is present in any given file. The detection and estimation is based only on the data presented in its observable form (i.e. nothing is known about the file prior to investigation). Because simply detecting the presence of hidden data may not be sufficient, steganalysis also covers the functions of extracting the message, disabling and/or destroying the hidden message so that it cannot be extracted, and finally, altering the hidden message such that misinformation can be sent to the intended recipient instead of the original message.

Depending on how much information is known about the embedded image, steganalysis techniques and methods closely mirror traditional cryptanalysis methods. The steganalysis attack methods can be broken into six types:

• Steganography-only attack: Only the file with the embedded data is available for analysis.

• Known-carrier attack: Both the original carrier file and the final (hidden message embedded) files are available for analysis.

• Known-message attack: The original message prior to embedding in the carrier is known.

• Chosen-steganography attack: Both the algorithm used to embed the data and the final (hidden message embedded) file are known and available for analysis.

• Chosen-message attack: The original message and the algorithm used to embed the message are available, but neither the carrier nor the final (hidden message embedded) file are. This attack is used by the analyst for comparison to future files.

• Known-steganography attack: All components of the system (the original message, the carrier message, and the algorithm) are available for analysis.

It follows that the success of any steganalysis technique is tied to the amount of information known about the file prior to Investigation. As more information about the file is known prior to investigation, the investigator can move from simply detecting to modifying or altering the hidden message before sending it on to the intended recipient. In the first category (steganography-only attack), the purpose of analysis is to simply detect the existence of a hidden message.

Without prior knowledge of the encoding mechanism, key, or data contained within the message, recovery of the contents using this method while possible, can take an excessive amount of time. With access to the original carrier and the final file with the embedded content (known-carrier), the purpose of analysis can move toward recovering the embedded message by comparing the differences between the two *files. If the algorithm* is known and the file with the hidden message embedded is also available (chosen-steganography attack), the analyst may have the ability to reverse the embedding to recover the hidden message and can easily alter or destroy the hidden contents.

Finally, if the analyst has the algorithm and a message prior to embedding (chosen-message attack), they can move towards identifying possible (hidden message embedded) files to attempt to recover the original carrier. If the carrier can be recovered or closely reproduced, the ability to insert alternate messages in lieu of the original message is possible.

The steganography-only attack can be accomplished through the use of statistical analysis performed on the final medium. In the following example, the color contents of JPEG images are examined. A modification to each coefficient's LSB produces variations in the data that results in deviations to the histogram for the given file. If the deviations are large enough to produce noticeable aberrations, the embedded file's histogram can identify the existence of the hidden message. Likewise, LSB modifications to palette-based images (GIF, etc.) cause duplications of the colors in the palette with identical or nearly-identical colors appearing. This duplication of colors can also serve as an indicator pointing to the existence of hidden data.

When examining the grayscale histograms for an original and a steganographically embedded JPEG (such as in Figure 3), slight deviations in the histograms are noticeable. The grayscale histogram provides a cumulative value for all three color channels (red, green, and blue) at each brightness level (0-255). As such the value displayed in the graph for brightness level 100, would be the total number of pixels in the image with a value of 100 in grayscale brightness. By modifying the original palette LSBs or the LSBs of the DCT coefficients, the histogram values shift to reflect the change in the number of pixels containing that specific value. To demonstrate this phenomenon,

Figure 3 compares the same photograph in its original form (containing 42,784 colors) to an embedded version of the file (containing 42,886 colors)

Figure 3: Original & Steganographic(right) image

The arrows in the embedded histogram indicate two obvious differences in the waveform (the introduction of pixels near brightness level 64 and the reduction of pixels near level 175). Steganalysis takes this phenomenon one step further by comparing the normalized distribution of colors against a predicted value. For palette based images, a normal distribution of color frequency is likely. A scalable standard bell curve can be assumed as the comparison benchmark against the suspect file. As seen previously, changes to the LSBs for any given pixel can create duplicate (or near duplicate) colors in the image's color palette. The duplicate colors increase the frequency for that value and can create a spike in the distribution exceeding the benchmark reference. Any large deviations from the benchmark can be an indicator of anomalies or modifications to the contents of the file. The process for JPEGs can be a bit more complicated. Because the JPEG format does not use a palette based encoding algorithm, a second step is necessary to compare DCT frequency to a benchmark. Recall that DCTs are reference points based on the quantized value for each color channel in the 8 *8 grid . As references, they are small by nature and plotting the frequency of a grid's coefficient values to another without compensating for the quantization reference is pointless. Further, the value of any given coefficient only affects a small percentage of the total number of pixels in the image. When tallied individually, the histogram for the DCTs will only tell whether the image contains elements of high contrast or not. (i.e. a photo of the blue sky vs. a picture of the international balloon fiesta in Albuquerque, NM) . The coefficients for the blue sky should have less variance than the coefficients for the photos of a colorful balloon.

Algorithms that sequentially modify the DCT coefficients in JPEG files tend to cause distortions in the histogram that flatten out the frequency values of adjacent DCTs. To compensate for this issue, newer algorithms do not sequentially embed the data but rather use a password or key to generate a random order for DCT or LSB modifications. Some readily available software titles for steganography detection include StegDetect, Stego Watch, and Steg Spy. Each of these titles use some form of statistical analysis on the target image to predict the existence of a hidden message. Westfield and Pfitzmann used a X^2 test to predict the probability that an image contained steganographic content by comparing the expected distribution (the null hypothesis) against the sampled values. If the measured value produced a deviation from the expected, then the amplitude of the deviation was proportional to the probability of steganographic content at that point in the file. Because their algorithm ran on sequential bytes with an increasing sample size for each calculation, when the probability dropped, the size of the hidden message was often revealed as well.

Statistical steganalysis has been made more difficult recently because some steganography algorithms specifically take measures to preserve the carrier file's first-order statistics to avoid this type of detection. Further, encrypting the content of the embedded message makes detection even harder because encrypted data generally has a high degree of randomness associated with it . After detection of hidden content with a carrier file, the next step is recovery of the hidden message itself. For known-carrier and chosen-steganography attacks (where the algorithm used to embed the data is known) some of the same detection tools have been extended to make use of brute force message recovery to also break the key used to embed or encrypt the data. With respect to JPEG files, there are several software titles that hide information using these variations of LSB insertion. "JSteg sequentially embeds the hidden data in least significant bits, JP Hide&Seek uses a random process to select least significant bits, F5 uses a matrix encoding based on a Hamming code, and Out Guess preserves first-order statistics." If intercepted en-route, after the hidden message is recovered (by breaking the encryption and embedding key or otherwise) the same carrier file can be used to embed an alternate message prior to sending it on to its final destination. Because modifications to the data comprising the carrier file are made without incorporating a mapping back to the original values, recovery of the original carrier file is difficult and sometimes impossible.

6. DENYING STEGANOGRAPHY

Far from the technical challenges facing the detection and recovery of hidden data, altering steganographically embedded information for common carrier types is relatively easy. System Owners and administrators seeking to disrupt the communications channel provided by steganography can implement file transformations in the communication channel to accomplish this goal. Recalling that the most common data types are image, video, and sound files, one simple approach is to simply change the format of (transform) the data by re-encoding it into an alternate format. The use of a guard processor at the entry and exit point(s) of the systems network could accomplish this task. For example, Figure 4 displays a photograph of the India Gate. The photo on the left is the original photo in a bitmap format, the photo on the right has a Microsoft Office Excel spreadsheet (64 Kb in size) steganographically embedded in it. Again, the two photographs are indistinguishable to the human eye. Proving that the files contain differences can be done through the use of a cryptographic hashing algorithm that verifies differences indeed exist.

Figure 4: Original & Steganographic Bitmap

An MD5 128-bit hash provides a high degree of confidence that different inputs produce different hash outputs. Thus, differences in MD5 hashes provide a high level of certainty that the given inputs (the binary contents composing the two photos) contain differences. While the properties of the files (image size, number of pixels, etc) remained identical during the embedding process.

Table1-File properties

		MD5	
File Name			
	Size(byte)	Pixel	Depth(bit)
New_orig.bmp	c1b865197b559747be78a86bfa106b16		
	401,910	448*299	24
New_steg.bmp	a39fb606650363bd064d5d76b0af3c10		
	401,910	448*299	24
New_steg.bmp	a03448ae1050d4bece4be38615253fac		
	29,413	448*299	24
New_recov.bmp	1e55e9e65645892af9fe24e195e4dd53		
	401,910	448*299	24

The photograph with the embedded data (New_steg.bmp) is the same size, contains the same number of pixels, and the same depth as the original but the binary contents of the file are different than the original (New_orig.bmp). Visually, the two files appear to be identical but the MD5 sum provides credible evidence that that is not the case.

To illustrate how to defeat the steganographic mechanism, the final file (New_steg.bmp) was converted into a JPEG by opening it in Microsoft Paint and using the "save as" feature to save it in the JPEG format. Note that the MD5 sum of New_steg.jpg does not match either the original or the embedded version of the photograph. An expected and noticeable reduction in file size is achieved when using JPEG compression. In this case, once the final file is converted into a new format, the embedded message is destroyed and the covert steganographic channel is effectively denied.

The final step to proving this is the case was to reconvert the JPEG image back into a bitmap. Again Microsoft Paint was used to open the JPEG image and the "save as" feature was used to save it in the bitmap format. Note that the recovered image (New_recov.bmp) has identical properties to the original and steganographic files, but contains a different MD5 sum. The recovered image no longer contains the hidden message and it is not the same file as the original. The modifications to the original file when the Microsoft Office Excel spreadsheet was embedded made irrecoverable changes to the bits defining each pixel.

For video and audio files the process outlined above remains the same. Convert the file to another format that requires a conversion, such as a lossy compression or expansion routine, and the embedded data will be destroyed in the process. With the exception of high compression data formats, the resulting "cleaned" reproduction of the file should show no noticeable deviations from the original. For text based denial techniques, the process can be a bit more complicated. Removal and/or addition of carriage returns and white space (such as adding an additional space after every period in the text) can shift the placement of characters which can break the character mapping decryption keys rely on.

Techniques like this can also alter the spacing of characters in a stepped character approach. Character space shifting approaches often require that the final document, or at a minimum the individual character, is an image instead of text. These steganographic insertions can be defeated using standard original character recognition software to rebuild the original file from the OCR output. Documents that are not image based (such as this report in its PDF format) can have the text copied and pasted into another document.

Synonyms can also be used to replace the awkward text often found when words are substituted in stepped character routines. This approach not only denies the steganographic channel, but leaves the intended message in the carrier intact and can make the document more pleasant to read.

The injection of bits into the headers of TCP/IP packets does not modify the content of the payload in any way. Steganographic covert channels utilizing techniques such as this are easily defeated through the use of monitoring features at the switch or router level. Malformed packets can be screened out or modified to conform to a specific rule set. Consider packets with the do not fragment (DF) bit manipulated so that the packets carry a covert message. A history or state based rule set could trigger on packets going to the same destination under the same protocol but having inconsistent DF bits.

Other network steganography denial techniques could include a security specification stating that the DF bit on every packet leaving the switch/router should have a value of one and all packets entering should have a value of zero. At a more rudimentary level (knowing that it could be detrimental to some fragment sensitive applications) network security could be achieved by forcing the above conditions and modifying the flags.

7. Conclusion

Computer forensic professionals need to be aware of the difficulties in identifying the use of steganography in any investigation. As with many digital age technologies, steganography techniques are becoming increasingly more sophisticated and difficult to reliably detect. Once use is detected or discovered, obtaining the ability to recover the embedded content is becoming difficult as well. Acquiring knowledge of current steganographic techniques, along with their associated data types, can provide a critical advantage to an investigator by adding valuable tools to their forensic toolkit.

Finally, due to the relatively simple techniques capable of denying the exploitation of a covert steganographic channel, companies may wish to take precautionary measures. By enacting measures discussed in this paper, they can ensure their proprietary and trade secret information is not being shoplifted inside of the daily podcast, shared in family photos, or distributed via the latest YouTube video.

REFERENCE

1. Herodotus, “The Histories”, Penguin Classics; Reprint edition, September 1, 1996
2. R. Krenn, “Steganography: Implementation & Detection”, found online at <http://www.krenn.nl/univ/cry/steg/presentation/2004-01-21-presentation-steganography.pdf>
3. R. Rivest, “The MD5 Message-Digest Algorithm”, MIT Laboratory for Computer Science and RSA Data Security, Inc, April 1992, .can be found online at < http://www.faqs.org/rfcs/rfc1321.html>
4. N. Johnson, “Digital Image Steganography and Digital Watermarking Tool Table”, found online at <http://www.jjtc.com/Steganography/toolmatrix.htm>
5. R. Krenn, “Steganography: Implementation & Detection”, found online at <http://www.krenn.nl/univ/cry/steg/presentation/ 2004-01-21-presentation-steganography.pdf>
6. N. Provos and P. Honeyman, “Hide and Seek: An Introduction to Steganography”