CovertFS Implementation

Ryne Flores, Kyle Gorak, David Hart, Matthew Sjoholm, LTC Timothy Nix
Department of Electrical Engineering and Computer Science
United States Military Academy

Abstract—In 2007 Arati Baliga, Joe Kilian, and Liviu Iftode of Rutgers University presented the idea of a web based covert file system [1]. The purpose of a web based covert file system is providing confidentiality and plausible deniability regarding the existence of files. The key components of a web based covert file system is hiding files within media using steganographic techniques using online file sharing for the purpose of collaboration [1]. We created such a system, covertFS, as a prototype to develop the idea for a web based covert file system previously mentioned. Our prototype of the web based covert file system implements steganographic techniques to obfuscate an entire file system on a public file share site. By storing data in this manner, rather than within the application or on a users computer, we provide confidentiality and plausible deniability to both the users and the companies that manage these public file sharing sites.

1 Introduction

Technology continues to integrate itself more and more in our daily lives because of the convenience it provides to users. Users are becoming connected to the internet in multiple ways nearly everywhere they go. Numerous devices are being developed and released to provide more convenience to users and to keep them updated on their various interests. A user will usually have at the very least a cell phone on their person and throughout their routine on any particular day will browse email, social media sites, and various other applications. These applications themselves, regardless of the convenience they provide users, can track and offer the user the option to post their information on the user's social media site profile to share among friends and acquaintances.

The majority of people interact with various forms of technology throughout each day. Depending on what type of technology we interact with there may be some type of data collected from our interaction. Also the use of social media is wide spread and it is easier to find various types of information about people because of this.

The means to store and share information in a covert manner not only serves multiple purposes, but also has various implications depending on how those means are used. There will be innovations designed with the good intentions, yet there will also usually be clever modifications of those designs for malicious intentions.

This project is an application that allows users to share information in a covert manner that can provide plausible deniability to not only the users but also to the file sharing or social media site that is being used as a medium where the information is stored.

2 Related Works

There are other projects and systems that have been released which offer users covert methods of sharing information. Such examples include Tor and StegFS[3]. The main concepts of this project were obtained from a paper that which involved faculty and students from Rutgers University[1]. There has been no other implementation of the application described in the Rutger's paper, and this project is our take at creating a working prototype which could later be used to develop a successful application or implementation of the covert file system.

3 Design Overview

In this section we will explain how we decided to implement a prototype of this project.

3.1 Web Connection

Our application's interface with the internet needs to serve three purposes:

1) Retrieve images to use for embedding a message.
2) Upload images to a public online forum anonymously without altering the image.
3) Retrieve specific images from a public online forum anonymously without altering the image.

After some research, we discovered a site that could satisfy the first purpose. The Cat API is a site that can offer random cat images. This is suitable because they are content neutral images that were big enough to store information inside, and the site is reliable. A downside to this approach is that statistical steg analysis can be done to compare the uploaded images to the originals. The possibility of using local unique images is out of the scope of our project. The next two requirements are fulfilled by Sendspace.com. This site allowed our application to easily interface with to upload and download images anonymously.

The only significant issue that we encountered while using these online resources was with Sendspace.

The online resources work together as follows:

1) The Cat API retrieves a random cat image for a message to be embedded into.
2) Embeds message into image.
3) This new image is uploaded to Sendspace. Sendspace returns and download and delete URL.
4) The new image from step 3 can be retrieved from Sendspace using the download URL.

3.2 FUSE

What FUSE is, what is provides for us, and why we choose it.

3.3 Mounting the File System

Subsection that goes more in-depth of our use of FUSE and our file system.

3.4 MEMFUSE Module

3.5 COVERTFS Module

3.6 Mapping File System Data to Photos

How is the data stored in our file system, and how it is constructed.

3.7 Steganography and CovertFS

The primary purpose in a web based covert file system is confidentiality and plausible deniability. Using steganography can provide both confidentiality for users as well as plausible deniability. However, steganography has risks that make it vulnerable to both. Statistical analysis and visible alterations (anything else? – research) can make steganography detectable, thus removing the aspect of both confidentiality and plausible deniability. However, there are ways to reduce the risks of using steganography.

3.7.1 Risks Using Steganography

The greatest risk to using steganography, particularly our steganography module, is that an embedded message can be discovered using statistical analysis. Since the original images exist on Tumblr and can be retrieved using the Cat API, our images that get uploaded to Sendspace can be compared to the originals. Statistical analysis between the original and the modified images will indicate that a message has been embeded. There are alternative steganograhpy techniques, but their are still several steganalysis tools to detect them (cite http://www.computing.surrey.ac.uk/personal/pt/H.Schaathun/projects/Past/steganalysis.pdf). Not only can steganalysis detected changes, but they can be visually obvious as well. Figure A shows both an original image and an encoded image. The image

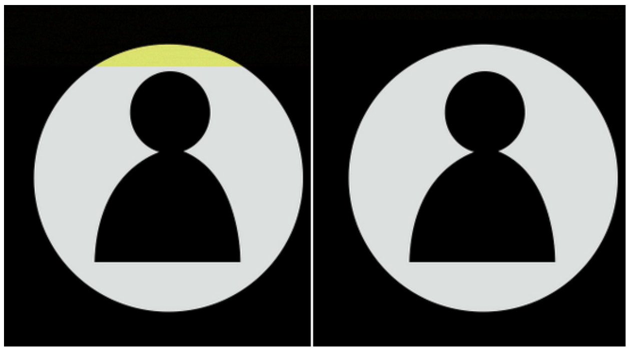

Fig. 1. Figure A

on the left is the one with a message encoded into it. The image on the right is the original. Notice the yellow in the image on the left - that is the encoded message. Because the message was encoded into white pixels, the pixels' value changes noticeably.

3.7.2 Mitigating Risks of Steganography

These risks are mitigated by the fact that there is no universal standard for steganography methods. A user can easily replace the steganography module that we offer with a module of their own. That way, even if someone knows an image is not the original, that person will still not know who encoded it, or what kind of module was using for the encoding. Also, an encryption module can easily be added to our application and used to encrypt messages before they are encoded into an image. So even if someone knows what kind of steganography module is being used to encode messages into images, that person will also have the added challenge of decrypting the information. Although, there is no research to indicate that encryption helps. Finally, since the original images exist on the internet, it will limit the plausible deniability of the user. It can be called into question as to why a certain user is uploading images that do not belong to that user to an online image database.

3.8 Encoding a File System

The covert file system uses a drop in steganography module that takes in a bytearray object and returns a URL to an image. We used "least significant bit" steganography for this application because of its simplicity and reliability.

3.8.1 Steganography Implementation

Least significant bit (LSB) steganography is a Substitution type of steganography [2] that replaces the least significant bits in the image's pixels. Our steganography module is not true LSB steganography because we replace multiple bits in each pixel allowing us to encode one byte of data into every pixel. We designed our steganography technique this way to decrease the amount of images required to encode large file systems at the cost of only minimal discoloring.

First, we break every byte of the message into three segments. Two segments of the byte will contain three bits, and the last segment will contain two bits. Next, we replace the three least significant bits of the red component with the first three bit segment. The green component follows, with the replacement of its least significant bits with the second three bit segment. Lastly, we replace the two least significant bits of the blue component with the remaining two bit segment of data.

There are obvious drawbacks to our implementation of LSB steganography, primarily that the image may appear distorted as seen in Figure X and CovertFS images are X percent easier to detect using standard statistical analysis techniques compared to true LSB steganography as seen in figure X. However, this implementation enables us to store more bytes than other implementations of LSB steganography which decreases the latency in uploading and downloading images in large file systems [? Need evidence].

3.8.2 File System to Images

The first step in encoding a file system is retrieving an image. We use ”The Cat API” which retrieves a random cat picture from Tumblr [?]. Once we retrieve the image, we determine the number of bytes we can encode in the image by multiplying the height and width of the image, in pixels. Since we encode one byte per pixel, the result of height times width results in the total number of bytes, n, of data we can encode in the image. For example, a 640x480 pixel image can hold 307,200 bytes of data. Next, we take the data we are going to encode and append a special end of file byte encoding. Then, we break the data into two sets, one containing the last n bytes and the other the remaining bytes.

Next, we encode the data in the first set into the image and upload the image. We retrieve the URL of the uploaded image and append a URL identifier byte encoding along with the URL itself to the remaining data. Then, we repeat the encoding steps by appending the end of file byte encoding, splitting the data, and uploading until there is no data left to encode. At this point, we have encoded all the data and have a ”linked list” of encoded images containing the data. Finally, we can return the URL to the head image of data.

4 Tails

Tails will be completed prior to publication. TBD.

5 Future Work

The original concept paper[1].

5.1 Encryption and Advanced Steganography Techniques

Another future implementation worth looking into is the use of a more advanced steganographic technique. The purpose of this is to ensure the confidentiality and plausible deniability of the users by:

1) Ensuring that pictures being used to hide the information do not give tell-tale signs of modification
2) To increase the difficulty of steganographic tools discovering

the encrypted pictures within the sea of publicly shared images within the file sharing site or social media site that the system it using to store the file system.

There are many different kinds of steganographic techniques that assist in reducing the effectiveness of statistical analysis, recovery of data, and other methods of detection. [?] Other implementations of steganography do not use images at all, but rather encode data within other forms of media such as audio files or PDFs. Using a better steganographic technique in addition to encryption would increase confidentiality and plausible deniability [?].

6 Conclusion

7 Acknowledgments

The original Web-Based Covert File System Paper [1].

References

[1] Arati Baliga, Joe Kilian, and Liviu Iftode. A web based covert file system. *HotOS*, 2007.

[2] Masoud Nosrati, Ronak Karimi, and Mehdi Hariri. An introduction into steganography methods. *World Applied Programming*, 1(3):191–195, 2011.

[3] Kian-lee Tan. StegFS : A Steganographic File System Laboratories for Information Technology Department of Computer Science. i:657–667, 2003.