Web-Based CovertFS Project

Ryne Flores, Kyle Gorak, David Hart, Matthew Sjoholm
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 [2]. 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 [2]. 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. More and more devices are being released to provide more convenience to users and to keep them connected and updated on various things they want to be informed about. 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 servers 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 aims to become an application that allows users to share information in a covert manner that can provide plausible denaibility 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[4]. The main concepts of this project were obtained from a paper that which invovled faculty and students from Rutgers University[2]. 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.

Yet by entering the covert realm of the internet we are allowing our project to criticized on the ethics of it development in accordance with ACM code of Ethics and Professional Conduct[1]. Just as Tor is reviewed arguments are brought up against further development[3] so to may this project draw vast attention because of what it offers or means to malicious users and intentions.

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 into.
2) Uploading images to an open online forum anonymously without altering the image.
3) Retrieving specific images from an open 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 nuetral images that were big enough to store information inside, and the site is reliable. The next two requirements are filled by sendspace.com. This had an API that our application can interface easily with, and it allowed anonymous uploading and downloading of images.

The only significant issue that we encountered while using these online resources was with Sendspace. At some point, Sendspace updated their API to require a key. We were not aware of this update and our application was completely broken until we found the bug and fixed it.

The online resources work together as follows:

1) The Cat API retrieves a random cat image for a mesage to be embedded into.
2) The cat image with the embedded message is uploaded to Sendspace. Sendspace returns and download and delete URL.
3) The cat image from step 2 can be retrieved using the download URL. This allows for the retrival of the original message.

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

We began the incorporation of steganography (steg) by researching different techniques, and decided to use the Least Significatn Bit technique due to its ease of implementation. This technique is easily detectable compared to others, but it was deemed acceptable for now. During the research, we found a Git repository that had source code for a Least Significant Bit steg module that matched our requirements. The source code could take a message, embed it into a picture, and then return the new picture. The source code could also take a picture with an embedded message and extract it. Encryption was included in the code, but we removed that functionality in order to maintain simplicity and reliability. In addition, the source code was written for Python 2. Because our application is written in Python 3, we had to modify the code, including what libraries to import, in order to for the steg to be implemented.

This first steg module had issues with reliability. In particular, the module would try decoding bytes using utf-8 encoding, and the results would vary. The size of the picture or the message length in characters seemed to have little effect on whether the bytes could be successfully decoded. We could have the decode method catch the error and only return the part of the message that was succesfully read, but this would ultimately yeild unreliable messages. We also tried having a method that would check the image to ensure that the message was properly embedded. This made the steg module cumbersome and very inefficient. The steg module could only be trusted to take very small messages (no greater than 100 characters for most images). We decided to rewrite the steg module in order to ensure reliability.

To increase the reliability, we increased the simplicity of the steganography. Instead of trying to manipulate color bit values and concern ourselves with the encoding and decoding of those bits, we tried manipulating the color decimal values. This steg module was written entirely from scratch by this team. In order to embed a message into an image, a string is concatenated to the end of the message that signals its termination. In this case, ”ENDMSG” was added to the end of all messages being embedded. Then, each character of the message is converted to its ASCII decimal value, and then encoded into the color values of each pixel. This method turned out to be very reliable. The main issue was that it was difficult to embed files into images instead of just strings.

After the application was tested using this reliable steg module, we decided to go back to using the Least Significatn Bit technique. If we could write our own module from scratch that manipulated the bits in the picture, we figured it would be easier to place the file bits into the image. By being consistent in only manipulating bits throughout the module, we could avoid the type errors and inefficiences that would arise if we used the steg method described in the previous paragraph. This new steg module is not completed yet and has not been tested. If it works, it would allow us to encode entire directories.

4 Future Implementation Plans

One of the things we would like to implement into this project is developing the covertfs into the TAILS operating system and having it run as a bootable program through the a flash drive. The reason for this is because the TAILS operating system in and of itself offers various security measures to ensure a stable and fucntioning system. We would like to develop a working prototype where our covertfs is integrated in the TAILS operating system to further allow confidentality and plausible deniability to the users.

The original concept paper[2]

4.1 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 steganalysis 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.

5 Conclusion

6 Acknowledgments

The original Web-Based Covert File System Paper [2].

References

[1] Ronald E. Anderson, Deborah G. Johnson, Donald Gotterbarn, and Judith Perrolle. ACM code of ethics and professional conduct. *Communications of the ACM*, 36(2):98–107, 1993.

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

[3] Clement Guitton. A review of the available content on Tor hidden services: The case against further development. *Computers in Human Behavior*, 29(6):2805–2815, 2013.

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