**SY486J: Cyber Crime Investigation: Lab eight**

**Steganography**

**In class lab**

1. **Steganography overview**
2. Steganography is essentially the act of concealing a message, image, or file within another message, image or file.
3. The advantage to steganography over cryptology and other associated realms is that within steganography, the hidden message does not attract attention to itself as an object of scrutiny.
4. Since the hidden message is not plainly visible, it is more difficult to detect and break messages protected by steganography. As a result of this, it has become a popular method for hiding data – especially digital data.
5. In digital steganography, which is the concealment of messages within computer files, many electronic communications may be carrying messages hidden by stenographic code within a transport layer, such as an image, program, or document.
6. Some file types are more ideal for hiding messages due to their large size, which makes steganography difficult to detect. Media files are ideal for sending steganography transmissions for this reason. They do not attract as much attention as sending a message hidden within a text or Word file, which may give away the presence of steganography due to a much larger than normal file size.
7. Some simple forms of steganography include making text the same color as the background in text documents, emails, and forum posts, embedding pictures in video material, concealing data (such as instructions or coordinates) within random or encrypted data, or within media files.
8. **Tools for steganography**

There are multiple free tools for steganography available online. We will be looking at some of these within this tutorial. In particular, we will be looking at the following tools:

* **Spam Mimic** – Spam Mimic is a site that allows a user to hide a short message within a spam email that can then be sent to either the user or an intended party. The intended party can then enter the text from the spam email into the Spam Mimic decoder to see the message. Though it is clear that Spam Mimic is using a mimic algorithm of some type to hide the message, the full process does not seem to have been discovered as of yet. This is a lossless method of steganography, which means that the original data (message) can be perfectly reconstructed from the compressed data (the generated spam email).
* **OpenPuff** – An open-source steganography application that supports a wide variety of "carrier" formats in which to hide data, including MP3, JPG, and more. OpenPuff asks that the hidden information be protected by three passwords, though it is possible to use only one. Additionally, it supports plausibly deniable encryption. This means that even if someone knows there is hidden information in a file, it is possible to create a decoy along with the real message. The person will then decrypt the decoy message, and the original will remain hidden. This is quite possibly the newest and most up-to-date free steganographic tool available at this point in time, having been updated as recently as 2013.
* **S-Tools** – This is a lightweight steganography tool released some years ago. However, it has been used for raining purposes and general steganography learning until 2011. S-Tools, like OpenPuff, allows a user to hide a message protected by a passphrase that both the sender and receiver must have. It permits the user to hide a text or other file inside of an image or wav file. However, it does not allow for decoys, and is not nearly as robust in its encryption methods as OpenPuff.
* **TrueCrypt** – A robust program that is used largely for on-the-fly encryption. It can create a virtual encrypted disk within a file or encrypt a partition or an entire storage device (an entire hard drive or system). These volumes tend to be extremely difficult to decrypt, as no back door has been built into the system in case a password is forgotten.

1. **Why is this an important topic?**

* In many cases, the use of encryption and steganography is an effective method used by cyber criminals to hide information. While some of the older freeware programs are less useful due to the fact that the older, less sophisticated algorithms have been cracked by steganographers and cryptologists, the new tools that have come to be known (like OpenPuff and TrueCrypt) have become obstacles in the realm of digital forensics. In fact, there is one case in which a cyber-criminal used TrueCrypt to create a hidden volume, and the FBI spent over five months attempting to crack it with no success.
* In reality, the fact is that in most of these cases, it is much easier to attempt to get the password information from the criminal or an involved party than it is to spend months and hundreds of thousands of dollars attempting to crack encryption that has been placed on sensitive or potentially criminal information. However, it is important to understand the process of steganography in order to better comprehend the methods of hiding data used by cyber criminals today.
* In addition, the tools within this tutorial can be used in conjunction with other tools to potentially recover information that could lead to the decryption of encrypted data. For example, perhaps you have been monitoring a suspect’s network traffic using WireShark, and have come across a series of email bits that you have reassembled to reveal a set of passwords. If a suspect has emailed themselves or another individual information containing passwords for encrypted information, once that encrypted data is seized, it is possible to use the password information found in the analysis of network traffic to decrypt the data.
* Being aware of the various programs and services that are available for the use of steganography and encryption make it more likely that, when coming across encrypted data, you will have enough knowledge to potentially identify what program was used, and what weaknesses or back doors may be associated with the program used.

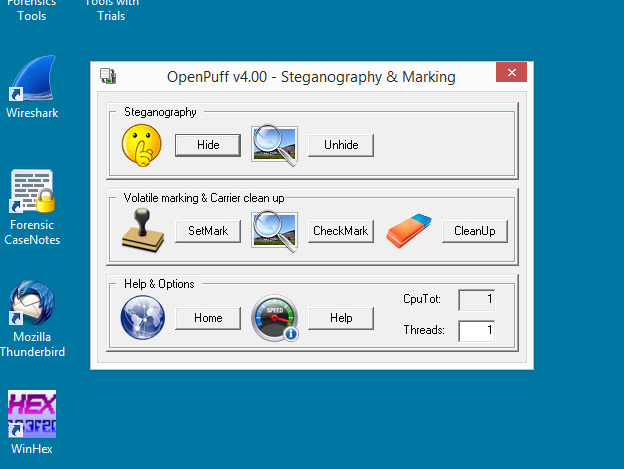
### **Lab exercise**

In this lab exercise, we will be hide data using bit-shifting and OpenPuff. Many of these tools are used by cyber criminals today. By hiding our own steganographic messages, it will become clear the variety of tools that make digital forensics investigations more difficult.

1. Hide a message within an MP4 video file using OpenPuff;
2. Use WinHex to hide data through bit-shifting

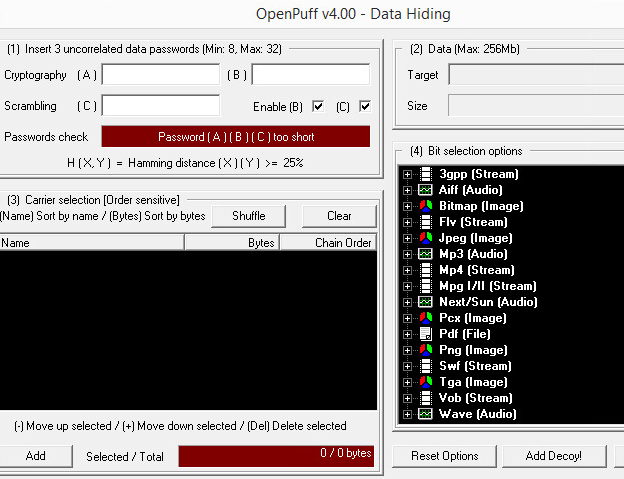
**Using OpenPuff**

1. OpenPuff is a professional freeware steganography tool with multiple features. It is a stealth/portable software, and uses 256-bit encryption methods to make data secure at a high level. It also has a decoy mode, which we will view. The website for OpenPuff can be found at: <http://embeddedsw.net/OpenPuff_Steganography_Home.html>. It has been downloaded for you and is in the Shared drive.
2. Create a folder on the Desktop called Hiding, and create another folder called Hidden Data within the Hiding folder.
3. Navigate to the **Shared** folder and download the **zipped OpenPuff, password.txt, the Cops versus Geeks Examination and Analysis … file,secretdata.txt**  into the Hiding folder.
4. Extract the zipped folder and run it. You get the screen below.



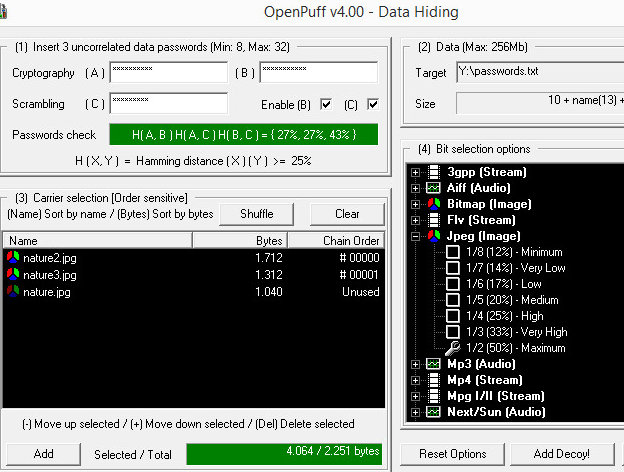
1. At the moment we will focus on hiding data, although there are various options available. Click on **Hide** in order to begin hiding data. A new window will open with an interface containing options for hiding data.

* **Section 1**: The empty fields marked “A”, “B”, and “C” are for passwords to protect the data. **To use one password only**, uncheck the **Enable** boxes for “B” and “C”. **Your password should be at least 8 characters including alphanumeric and includes special characters.**
* **Section 2**: This section on the right side of the screen is where you select the target (the message you would like to hide).
* **Section 3**: This is the carrier selection (the type and location of file that will contain the hidden data)
* **Section 4**: Bit selection options.



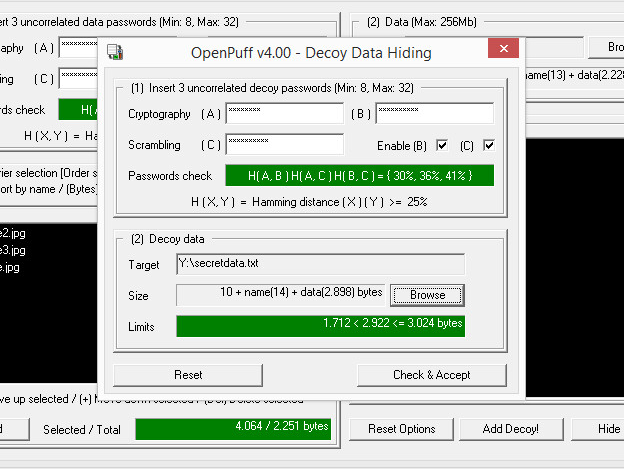
1. We are going to start fairly simple, and hide a test file with some passwords inside of a MP4 file. A text file with passwords in it has been created for you (passwords.txt) .

* **Section 1**: Enter three passwords of your choosing, but make sure you can remember them, ELSE use one. Note that if they are too similar they will not be accepted.
* **Section 2**: Select passwords.txt as the target from the Hiding folder
* **Section 3**: Select **the MP4 file**  from the Hiding folder.
* **Section 4**: Select Navigate to the **MP4** option to view the bit selection Change this to **1/2 (50%) maximum** – if it’s not set to that



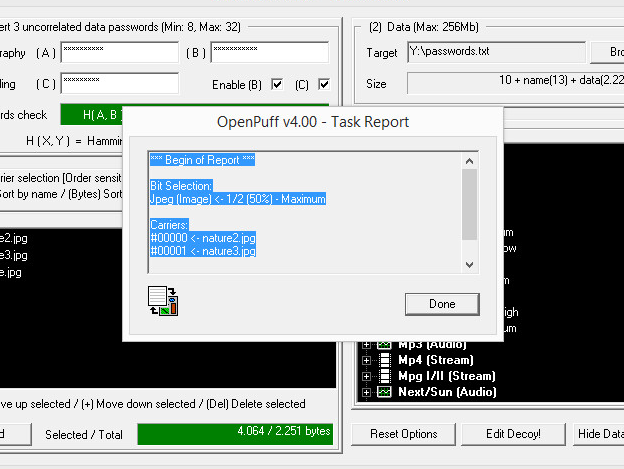
1. We are going to explore adding a decoy to increase the protection of our sensitive data. Click the **Add Decoy!** Button.
2. Create a password file named secretdata.txt – which is your decoy – save it in the Hiding folder. The secretdata.txt contains a set of passwords that are easy to remember. Select secretdata.txt, as data to hide(which contains false data to mislead people attempting to get ahold of your data) as the source. Click **Check & Accept** to add the decoy data. A new window will pop up stating that the passwords and decoy data are OK.

\*Note that the decoy data must be around the same size as the actual hidden data to avoid suspicion.



1. Click **Hide Data & Decoy!** to continue hiding the data. A window asking to save the information will appear; save the data to the folder called **Hidden Data** which is in your Hiding folder. When hiding is completed, a notice telling you that one of the carriers was not used will appear. Just note that when you decrypt the data, this carrier will not need to be loaded, since nothing is hidden in that file. A Task Report window stating what was done will appear upon completion. Here we can see that two carriers were used and contain the hidden data.

Explain to Dr.Shumba what you did: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

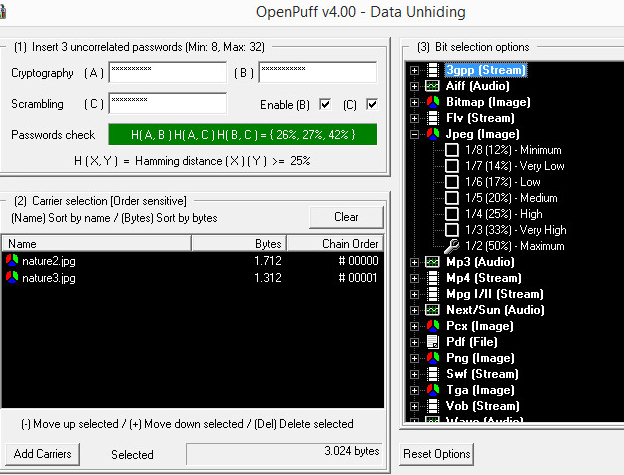


1. You will see the files in which the data was hidden.
2. Close the Hide Data window – this will return you to the Main Menu. We will now attempt to unhide our data. Click on **Unhide**. A window that looks similar to the ‘Hide’ window will appear. Enter the original set of passwords you created, and be sure to select the carriers that were saved the **Hidden Data** folder. Click **Unhide** to unhide the passwords.txt file. Save it to the Hidden Data folder.

Try playing the MP4. Does it play with hidden data? \_\_\_\_\_\_\_\_

This method reveals the original file. However, using the three passwords set for the decoy will unhide the decoy secretdata.txt file. Do the same thing as above, but enter the information for the decoy file.

Essentially, the point of creating a decoy is to create deniability of hiding sensitive data. The decoy data makes up some of the white noise that is created when encrypting data, allowing you to claim that while you were hiding data, you were not hiding illegal or sensitive data.



Explain to Dr.Shumba what you have done

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

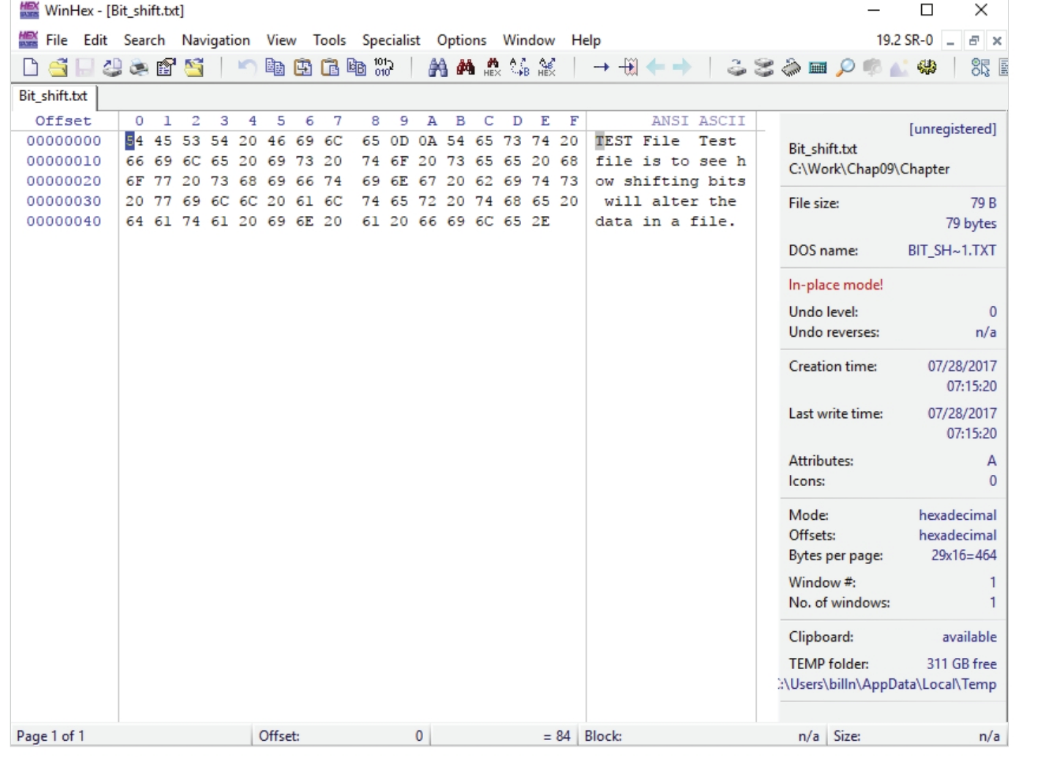
**Bit-Shifting**

A related, and well-known, technique for hiding data is shifting bit patterns to alter the byte values of data.

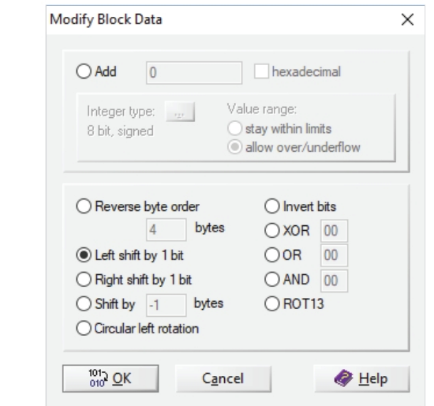
Bit-shifting changes data from readable code to data that looks like binary executable code.

WinHex and Hex Workshop include a feature for shifting bits and altering byte patterns of entire files or specified data. To shift bits in a text file, follow these steps:

1. Create a folder with whatever name on your Desktop.
2. Start Notepad, and in a text document, type **TEST FILE. Test file is to see how shifting bits will alter the data in a file**.
3. Save the file as **Bit\_shift.txt** in your work folder, and exit Notepad.
4. Start WinHex, using the Run as administrator option.
5. Click File, Open from the menu. Navigate to your work folder, and then double-click Bit\_shift.txt. You should see something close to the below screen.
6. To set up WinHex for bit-shifting, click Options, Edit Mode from the menu. Click Default Edit Mode (=editable), if necessary, and then click OK.



1. Highlight all the data in the file by clicking Edit, Select All from the menu.
2. Click Edit, Modify Data from the menu. In the Modify Block Data dialog box, click the Left shift by 1 bit option button, and then click OK. If you get a user.txt error, please uninstall WinHex and reinstall.



1. Click File, Save As from the menu, and save the file as **Bit\_shift\_left. txt** in your work folder. The text has changed to random values.
2. Exit and then restart WinHex as administrator.
3. To return the file to its original configuration, you need to bit-shift it back to the right. Make sure the data is highlighted, and then click Edit, Modify Data from the menu.
4. In the Modify Block Data dialog box, click Right shift by 1 bit, and then click OK.
5. Save the file as **Bit\_shift\_right.txt** in your work folder, and leave this file open in WinHex for the next activity.

**Dr.Shumba’s signature**

Now you can use WinHex to find the MD5 hash values for these three files and determine whether Bit\_shift.txt is different from Bit\_shift\_right.txt and Bit\_shift\_left.txt. To check the MD5 values in WinHex, follow these steps:

1. With Bit\_shift\_right.txt open in WinHex, click File, Open to open Bit\_shift. txt, and then repeat to open Bit\_shift\_left.txt.
2. Click the Bit\_shift.txt tab to make it the active file, and highlight the file’s content by clicking Edit, Select All.
3. Click Tools, Compute Hash from the menu to open the Compute hash dialog box. In the list box, click MD5 (128 bit), if necessary, and then click OK. Copy the MD5 hash value of Bit\_shift.txt, and paste it in a new text document in Notepad.
4. Click Close in the MD5 (128 bit) dialog box.
5. Repeat Steps 2 and 3 for Bit\_shift\_left.txt and Bit\_shift\_right.txt, pasting their hash values in the same Notepad text file.
6. Compare the MD5 hash values to determine whether the files are different. When you’re finished, exit WinHex. In Notepad, save the text file as Bit\_shift\_recovery.txt in your work folder, and exit Notepad.
7. If your output is correct, the Bit\_shift.txt and Bit\_shift\_right.txt files should have the same MD5 hash values.
8. Show Dr.Shumba your hashes and the three open files.

Dr.Shumba \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Exit WinHex and Notepad.

**Read this carefully:**

Typically, antivirus tools run hashes on potential malware files, but some advanced malware uses bit-shifting as a way to hide its malicious code from antivirus tools. With the bit-shifting functions in WinHex, however, you can inspect potential malicious code manually. In addition, some malware that attacks Microsoft Office files consists of executable code that’s embedded at the end of document files and hidden with bit-shifting.

When an Office document is opened, the malware reverses the bit-shifting on the executable code and then runs it.

Show Dr.Shumba the hashes for the 3 files