CSCI 400 Cryptography Lab 01 – Group 4

Topic: Steganography

Jae Cho, Edward Cruz Peralta, Carl Laguerre, Akbor Uddin

1 General description

The purpose of steganography (concealed writing), as different from cryptography, is to hide the

fact that a message is even being sent. In ancient times, generals would shave a soldier’s head, write

on their scalp, and let the hair regrow. The recipient would then shave the soldiers head and read

the message. We will be doing the computer science equivalent using embedding functions, e.g. by

changing the nth bit in a digital image, audio file, or even a network protocol.

2 File with samples and results

The files are in lab-stego.zip. It contains some tools (steghide, vecna, stegdetect, stegsecret), sample

(images, audio) files, and text. Also, inside are network steganography tools using TCP/IP headers

(covert tcp).

3 Hiding/extracting information

Refer to steganography primers for background. The basic principle is to use redundant bits in

the cover medium to embed information that is not to be found. Cover media include text, images

(GIF, BMP, JPG), audio files (e.g. MP3, WAV), video files (e.g. MPEG4). The goal is to produce

output files that do not give the appearance of containing additional information.

3.1 Requirement

\_ You should use the tools provided in the file. If for some reason, you choose not to, please

justify your choice.

\_ Your output should contain files that have embedded information in them, with or without

passphrases (depending on the algorithm). If you used a passphrase, you must submit it as

well.

1

3.2 Tasks

3.2.1 Embedding information into files

\_ Create a sample image file, which will be your cover file. Embed some information in the file,

creating a separate stego file for each of the following:

1. another image

2. a sound file (e.g. a cricket chirp file)

3. a short text

4. a long text

\_ Embed into each of the items (1. image file, 2. sound file, 3. short text file, 4. long text file)

above another item (1-4). Having worked with the image file, move on to the sound file as a

cover file, and embed another item into it, again creating 4 stego files. At the end you should

have 4 cover files (1-4), and 16 stego files.

\_ How would you embed information into a video (e.g. MPEG4) file? Give an example.

\_ Prepare a report of your findings.

Example: Embedding a text file into a JPEG file. You can do this in Windows, MacOS X, Linux,

or your favorite operating system. The example we will be showing here is steghide under Kali

Linux (currently at version 2018.3), but you can use any other operating system and steganography

tool to do so.

To install steghide on Kali Linux, you will need to run:

# apt-get install steghide steghide-doc

We then proceed to the actual embedding.

$ steghide embed -cf image.jpg -ef secret.txt -sf new-image.jpg -v

Passphrase: <simplepassword>

Re-enter passphrase: <simplepasswordagain>

reading secret file "secret.txt"... done

reading cover file "image.jpg"... done

creating the graph... 193 sample values, 1073 vertices, 496096 edges

executing Static Minimum Degree Construction Heuristic... 90.6% (1.0) done

writing stego file "new-image.jpg"... done

Note that we are using two image files here. One is the source image image.jpg, which we do not

modify and preserve for further use, and the stego file new-image.jpg, which contains the embedded

secret.txt.

3.2.2 Information gathering and extraction

We can then check the file and extract information embedded in it, assuming we know what method

was used to embed the information (e.g. steghide) and the passphrase is either known to us or empty.

2

$ steghide info new-image.jpg

"new-image.jpg":

format: jpeg

capacity: 3.8KB

$ steghide extract -sf new-image.jpg -xf extracted.txt -v

Enter passphrase: <simplepassword>

reading stego file "new-image.jpg"... done

extracting data... done

checking crc32 checksum... done

writing extracted data to "extracted.txt"... done

3.2.3 Embedding information into network traffic

The program covert tcp can hide data in TCP packets. Currently it is running on Linux only. You

can run this in a Linux virtual machine guest, e.g. under VirtualBox or VMware.

\_ Create two endpoint hosts with the network steganography tool installed, or you can use the

same host with two different terminal windows

1. Document the active connection with tcpdump data (pcap)

2. Transmit the information (e.g. a small text file, remember that transmission is going to

be slow)

\_ Prepare a report of your findings.

Use the included source code from the zip file and compile it using this command:

$ cc -o covert\_tcp covert\_tcp.c

or

$ gcc -o covert\_tcp covert\_tcp.c

You can then send and receive packets with steganographic data. You can use the command

ifconfig to get the IP address of your own machine, or the one of your guest machines.

To send a file a file (secret.txt) via the IP Identification field encoding from client IP to server IP,

you can use the command:

Client/sender:

# covert\_tcp -source client\_IP -dest server\_IP -file secret.txt

Server/receiver:

# covert\_tcp -source client\_IP -server -file secret.txt

Note that you should either be root to run the program (indicated by the # prompt) or use

sudo. Here is an example of covert tcp using sudo. You should launch the server first, as it is a

best effort and not a synchronized transmission.

3

$ cd test/

$ ifconfig

eth0 Link encap:Ethernet HWaddr 00:11:22:33:44:55

inet addr:146.111.34.195 Bcast:146.111.34.255 Mask:255.255.255.0

inet6 addr: fe80::001:9cef:ee34:6007/64 Scope:Link

UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1

RX packets:21696 errors:0 dropped:0 overruns:0 frame:0

TX packets:12790 errors:0 dropped:0 overruns:0 carrier:0

collisions:0 txqueuelen:1000

RX bytes:26146254 (26.1 MB) TX bytes:1696140 (1.6 MB)

Interrupt:27 Base address:0xa000

lo Link encap:Local Loopback

inet addr:127.0.0.1 Mask:255.0.0.0

inet6 addr: ::1/128 Scope:Host

UP LOOPBACK RUNNING MTU:16436 Metric:1

RX packets:802 errors:0 dropped:0 overruns:0 frame:0

TX packets:802 errors:0 dropped:0 overruns:0 carrier:0

collisions:0 txqueuelen:0

RX bytes:32400 (32.4 KB) TX bytes:32400 (32.4 KB)

$ sudo ./covert\_tcp -source 146.111.34.195 -dest\_port -server -file received-secret.txt

[sudo] password for testuser:

Covert TCP 1.0 (c)1996 Craig H. Rowland (crowland@psionic.com)

Not for commercial use without permission.

Listening for data from IP: 146.111.34.195

Listening for data bound for local port: 80

Decoded Filename: secret.txt

Decoding Type Is: IP packet ID

Server Mode: Listening for data.

Sending Data: J

Sending Data: o

Sending Data: h

Sending Data: n

Sending Data:

Sending Data: J

Sending Data: a

Sending Data: y

Sending Data:

Sending Data: S

Sending Data: e

Sending Data: c

Sending Data: r

Sending Data: e

4

Sending Data: t

...

In another terminal window, you can then run this command:

$ sudo ./covert\_tcp -source 146.111.34.195 -dest 146.111.34.195 -dest\_port 80 -file secret.txt

Covert TCP 1.0 (c)1996 Craig H. Rowland (crowland@psionic.com)

Not for commercial use without permission.

Destination Host: 146.111.34.195

Source Host : 146.111.34.195

Originating Port: random

Destination Port: 80

Encoded Filename: secret.txt

Encoding Type : IP ID

Client Mode: Sending data.

Sending Data: J

Sending Data: o

Sending Data: h

Sending Data: n

Sending Data:

Sending Data: J

Sending Data: a

Sending Data: y

Sending Data:

Sending Data: S

Sending Data: e

Sending Data: c

Sending Data: r

Sending Data: e

Sending Data: t

...

You will have to interrupt the server to end the transmission. You can then compare the two

files. The difference should come up to be empty:

$ diff -c secret.txt received-secret.txt

$

Note: If you only have one host running, you can do this using the loopback interface lo and its

associated IP address. You should record the traffic between the two hosts, or the two processes,

e.g. using tcpdump on the interface eth0 with a snaplen of 1500, and no DNS lookups:

# tcpdump -i eth0 -s 1500 -n -w covert-data.pcap

You can then visualize the traffic streams either using tcpdump directly, or using a tool such

as Wireshark. Here is the command to extract to the screen all traffic going to port 80, without

making any DNS lookups:

5

# tcpdump -n -r covert-data.pcap -X -v dst port 80

Can you spot the embedded traffic? If so, point it out in your answers. If not, explain why.

Repeat the experiment with alternative encoding options in covert tcp.

4 Digging deeper into the steganography tools

The final part of this lab consists of two sections: measuring the capacity of each file, that is how

much can be embedded and finding possible embedding in existing files.

4.1 Capacity of the cover files

Depending on the tool in use, you will be given an estimate on how much information (size in bytes)

can be embedded into the destination file. Express that in percentage of the file in use, and specify

the type of file used (image, audio, video, etc.)

4.2 Chaining the techniques

Try to embed more information in an already encoded file (effectively chaining the steganographic

technique).

Prepare a report of your observations in the format requested under Deliverables (section 6).

5 Word Problems

1. Summarize the embedding techniques used by the tools.

2. How would you detect the presence of steganography?

3. To what extent are the embedding techniques composable?

4. How would you thwart steganographic efforts if you could be in the middle of the transmission,

i.e. you take the role of an active warden and modify traffic in transit?

6 Deliverables

1. A report describing all your findings above.

2. A zip file containing:

\_ The source and embedded files, including the particular technique and passphrases used,

if any.

\_ Answers to all word problems in Part 5.

3. Submit by the beginning of the lecture on September 5, 2019.

6

7 Grading

Points will be subtracted if any of the pieces of the deliverables are missing or incomplete.

7