Cyber Security Technologies

Session 11 – Steganography

Shawn Davis
ITMS 448 – Spring 2016

Slides contain original content from Davis, S., and Lidinsky, W.

Note

- Logon to RADISH
- Copy the “Stego In-Class Labs” folder from M:\Tools folder to Win 8.1 VM

Overview

Part I – Introduction

Part II – Cyber Stego Taxonomy

Part III – Graphic Image Carriers

Part IV – Audio Carriers

Part V – Alternate Data Streams (ADS)

Part VI – Network Carriers

Part VII – Detection of Covert Data

Part I

Introduction

Terms and Concepts

- Steganography:
  - Science of hiding messages so that the existence of the message is unknown to outsiders
- How is this different from Cryptography???
- Steganalysis:
  - Detection of steganography

Al-Qaeda

- Berlin, May 2011
- Suspected al-Qaeda member arrested
- Had in his possession a flash memory with a password-protected folder
- Password was cracked by authorities
- Within the folder was a porn video that could be viewed without difficulty
  - OK. So the guy liked to watch porn.
  - But why was it password-protected?
  - And wasn't watching porn strictly against his religion?

© 2013 W.P.Lidinsky

Al-Qaeda

- So the authorities investigated further
- Applying steganalysis to the video they found, hidden within the video, 141 text files
- These text files included
  - A list of al-Qaeda's current operations
  - Reports on the status of these operations
  - Plans for future operations
  - Mistakes made in past operations
- So how did a terrorist hide all this in a seemingly "innocent" video file?
- And how did the authorities detect and extract the hidden info?

© 2013 W.P.Lidinsky

Operation Shady RAT

- This attack had been going on unnoticed for 5 years
  - McAfee finally discovered it in mid 2011
- Stole intellectual property from over 70 corporations, government agencies and NGOs in 14 countries
- Seemingly innocuous-looking image files were used to carry commands to previously infected computers
- Examples of image files used:

© 2013 W.P.Lidinsky

Russian Moles – June 2010

- FBI arrested 11 Russian "moles" who had been living in the U.S. for years as U.S. citizens
- These moles used a Russian-devised stego tool to conceal their electronic communication with the Russian SVR (Russian foreign intelligence agency)
- Communication was done using image files
  - Covert information was hidden in carrier image file by sender
  - Carrier image file was posted to public web sites by sender
  - Later the carrier image file was copied by the receiver
  - Covert information was then extracted by the receiver
- Classical spy "drop box" scheme adapted to the cyber world

© 2013 W.P.Lidinsky

How?

- So how can one hide potentially harmful information in a seemingly innocent carrier file so that the carrier file can be rendered (viewed, played) as though it carried nothing.
- Even more, how can one
  - Examine a file to determine if it carries hidden information
  - Extract and understand the hidden information
  - Or at least remove it
- The Steganography course in Spring 2016 focuses on these issues

© 2013 W.P.Lidinsky

Legitimate Uses of Stego

- Hiding trademarks and copyrights in image files
- Hiding copyrights in digital music files
- Called ***digital watermarking***

© 2013 W.P.Lidinsky

Modern Cyber Steganography

- Today digital information can be hidden many different ways
- Two factors common to many modern cyber-stego schemes
  - A **carrier** or **overt** file
  - A **covert** file to be **hidden**

© 2013 W.P.Lidinsky

Covert Data & Carrier Files

- Steganographic software is used to hide covert data inside of an overt/carrier file
- **Any** type of covert data file can be hidden inside a carrier file
  - A covert executable file hidden inside a carrier Word file
  - A covert image file hidden inside a carrier audio file
  - Etc.
- However, some carrier file types are better than others for holding covert data
  - Each Stego tool only works with specific types of carrier files

Popular Overt / Carrier Files

- Carrier files can be of many different types
  - Document files (e.g., text, spreadsheet, word processor)
  - Image files (e.g., jpeg, bmp, gif, tiff)
  - Video files (e.g., avi, flv, H120, H261 through H265, mpeg1, mpeg4)
  - Audio files (e.g., mp3, wav, G711, G729)
  - Web page files
- Most of the above are in the public domain
- There are many more that are proprietary
  - e.g., realAudio, realVideo…

© 2013 W.P.Lidinsky

Other Possible Carrier Schemes

- Video files
- Document files such as those of MSWORD
  - e.g., Put covert info in slack space
- Web pages
  - e.g., HTML non-visible text and images
- Simple text files

© 2013 W.P.Lidinsky

Hidden / Covert Files

- The file(s) to be hidden can usually be any bit stream
- So the nature of the file actually is irrelevant
  - Don't matter if the file is text, doc, xls, bmp, avi, mp3…
  - Don't matter if the file is compressed, encrypted or both
  - Don't matter what encryption or compression algorithm is used
  - They're all **just bits** as far as the stego hiding software is concerned

© 2013 W.P.Lidinsky

Part II

Cyber Stego Taxonomy

Taxonomy Overview

- Classification is based upon
  - **Where** covert data is hidden in the carrier
  - **How** covert data is hidden in the carrier
- Methods to hide data:
  1. **Insertion**
  2. **Substitution**
  3. **Generation**

© 2013 W.P.Lidinsky

1. Insertion

- Where data is hidden
  - Locations in the overt file that the application ignores
    - Header of File
    - Slack Space
    - Ignored space in the file proper
- How data is hidden
  - Data is added to file and the application ignores it
- The file size increases when covert data is added
  - Why does the file size increase???

© 2013 W.P.Lidinsky

File Header (PNG File Example)

bankshell.png

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000000	89	50	4E	47	0D	0A	1A	0A	00	00	00	0D	49	48	44	52	‰PNG IHDR
00000010	00	00	08	FE	00	00	05	02	08	06	00	00	00	90	3D	AF	þ =¯
00000020	D1	00	00	0C	17	69	43	43	50	49	43	43	20	50	72	6F	Ñ iCCPICC Pro
00000030	66	69	6C	65	00	00	48	89	95	97	07	54	53	C9	1A	C7	file H‰•— TSÉ Ç
00000040	E7	96	14	42	42	0B	84	22	25	F4	26	48	AF	D2	BB	54	ç– BB „"%ô&H¯Ò»T
00000050	E9	60	23	24	01	42	89	21	10	54	EC	C8	A2	82	6B	17	é`#$ B‰! TìÈ¢‚k
00000060	0B	8A	8A	AE	80	28	B8	16	40	16	15	11	C5	C2	22	D8	ŠŠ®€(¸ @ ÅÂ"Ø
00000070	FB	62	41	65	65	5D	2C	D8	50	79	93	02	FA	7C	6F	CF	ûbAee],ØPy“ ú|oÏ
00000080	3B	6F	CE	99	7B	7F	F7	9B	EF	FB	EE	7F	E6	CE	DC	33	;oÎ™{ ÷›ïûî æÎÜ3
00000090	03	80	A2	03	4B	20	C8	46	95	00	C8	E1	E7	0B	A3	83	€¢ K ÈF• Èáç £ƒ
000000A0	7C	99	89	49	C9	4C	D2	1F	80	08	D4	01	03	90	81	1D	|™‰IÉLÒ € Ô
000000B0	8B	9D	27	F0	89	8A	0A	03	FF	58	DE	DD	00	88	F8	7E	‹ 'ð‰Š ÿXÞÝ ˆø~
000000C0	D5	4A	9C	EB	9F	FD	FE	6B	51	E6	70	F3	D8	00	20	51	ÕJœëŸýþkQæpóØ Q
000000D0	90	53	39	79	EC	1C	C8	47	01	C0	B5	D8	02	61	3E	00	S9yì ÈG ÀµØ a>
000000E0	84	1E	68	37	9C	93	2F	10	F3	5B	C8	AA	42	28	10	00	„ h7œ“/ ó[Èª B(
000000F0	22	59	CC	E9	52	D6	16	73	AA	94	6D	24	3E	B1	D1	7E	"YÌéRÖ sª”m$>±Ñ~
00000100	90	FD	01	20	53	59	2C	61	3A	00	0A	E2	FC	CC	02	76	ý SY,a: âüÌ v
00000110	3A	CC	A3	20	80	6C	C3	E7	F0	F8	90	77	41	F6	64	67	:Ì£ €lÃçðø wAödg
00000120	B0	38	90	FB	20	4F	CC	C9	99	0D	59	91	0A	D9	2C	F5	°8 û OÌÉ™ Y‘ Ù,õ
00000130	BB	3C	E9	FF	96	33	75	3C	27	8B	95	3E	CE	D2	BE	48	»<éÿ–3u<'‹•>ÎÒ¾H
00000140	0A	D9	9F	97	27	C8	66	CD	FB	3F	87	E3	7F	97	9C	6C	ÙŸ—'ÈfÍû?‡ã —œl
00000150	D1	D8	3B	0C	60	A5	66	08	83	A3	C5	7D	86	E3	56	93	ÑØ; `¥f ƒ£Å}†ãV“

File Header

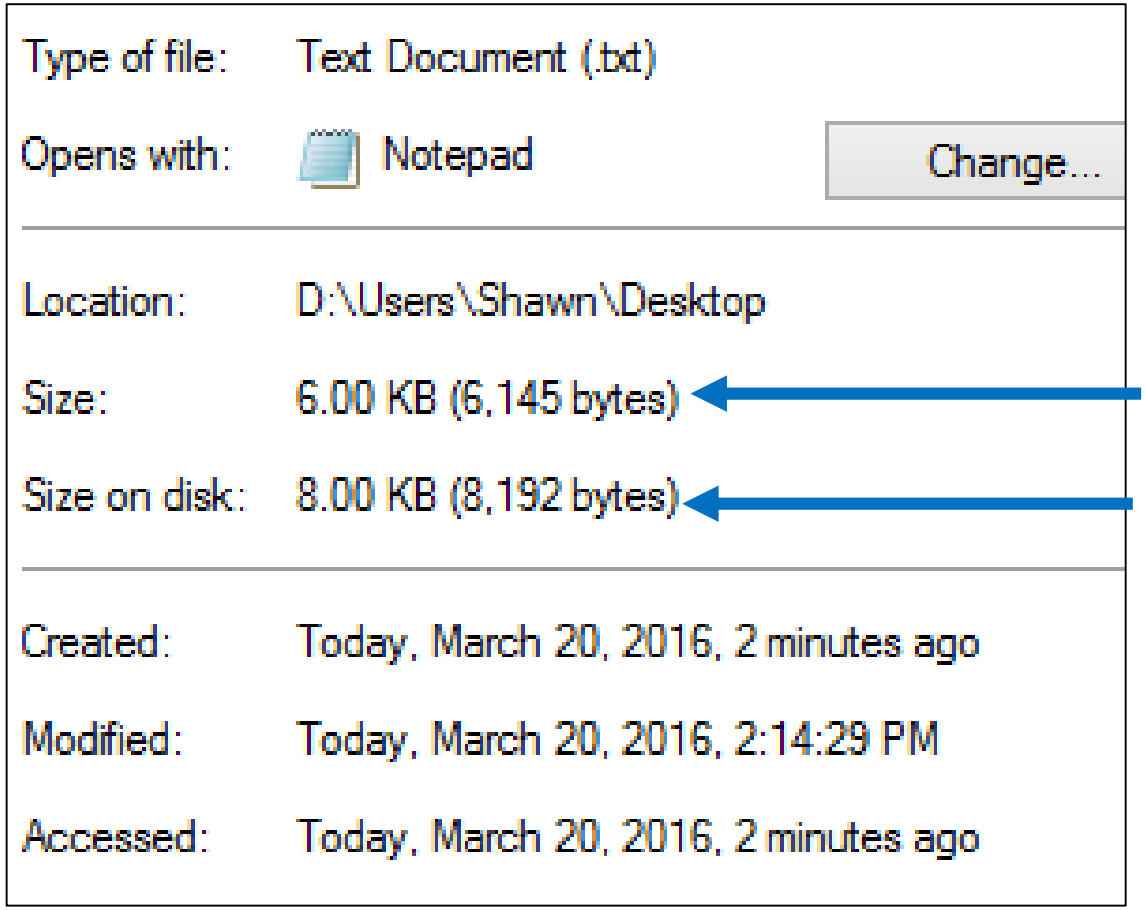
File Data

File Slack Space

- A hard drive is made up of sectors
- An operation system allocates space for files in clusters
- A Windows NTFS cluster is 4KB
- Let's say a file is 6KB
  - This is called the file's logical (actual) size
- Windows will allocate two 4KB clusters for it
  - This is called the file's physical size which is the space it takes on disk

File Slack Space (Cont.)

- I created a file with 6KB of content as an example:



Logical Size

Physical Size (Two 4KB Clusters)

File Slack Space (Cont.)

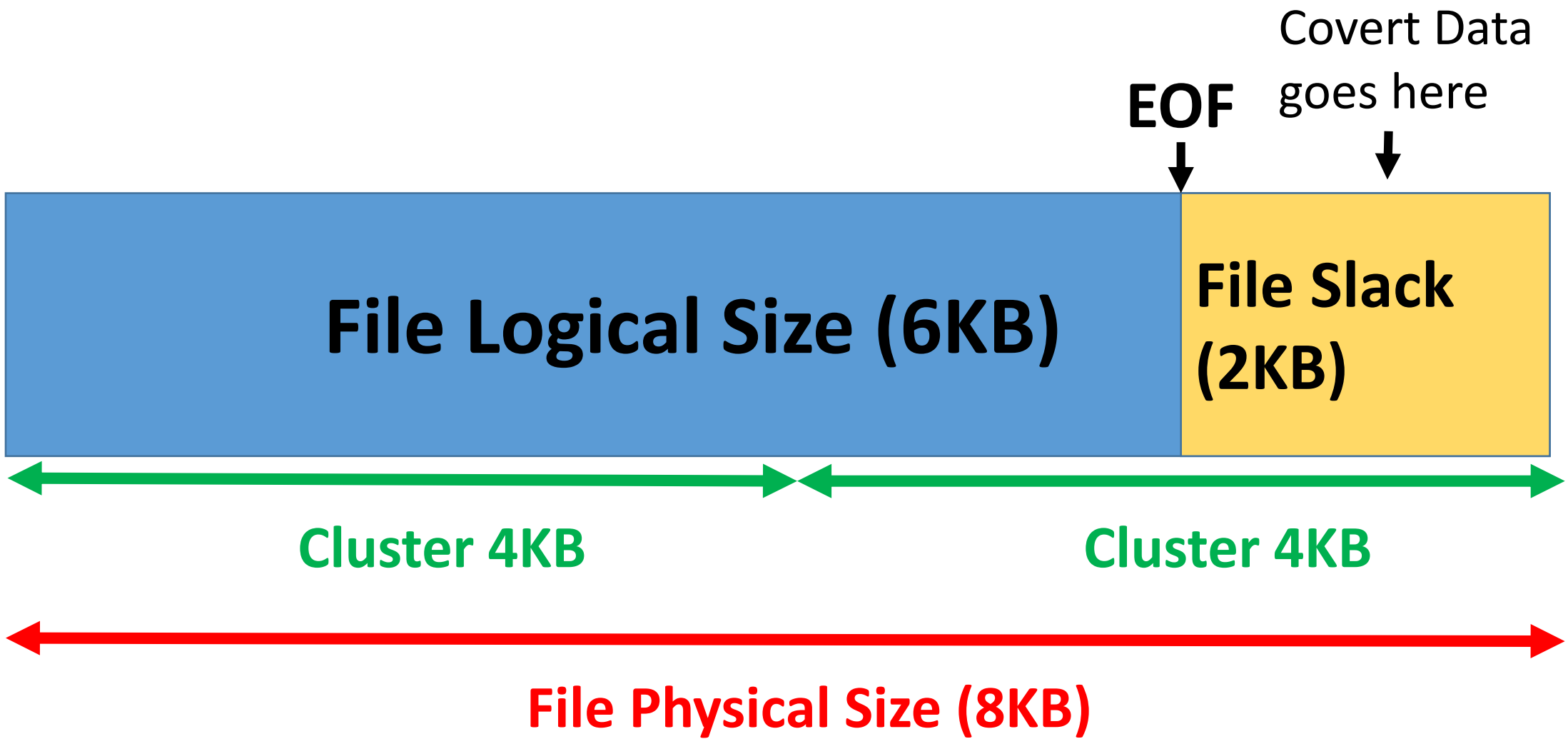
- Where the logical file ends is called the EOF (End of File)
- The space between the end of the logical file (EOF) and the end of the physical file (End of cluster) is called the File Slack

8KB Cluster Size

– 6KB Logical File Size

2KB Slack Space (for data hiding)

File Slack Space (Cont.)



Insertion Note

- Insertion is the most basic stego method and is often easy to detect since:
  - The file size increases
  - The covert data is often stored in one part of the file
- The next method (Substitution) is a much better way to hide covert data

2. Substitution

- Where data is hidden
  - In place of part of carrier file content
  - Modified carrier content
- How data is hidden
  - Overwrite carrier data in such a way that
    - It is not noticeable
    - Keeps the overt file usable by the application
- The file size generally does **not** increase when covert data is added
  - Why does the file size not increase???

© 2013 W.P.Lidinsky

Substitution Example

- Use a 16-bit .wav sound file as the carrier (overt file)
  - Each "chirp" is a instant of sound represented by a 16-bit number
  - The file is a sequence of chirps
- Distribute the hidden (covert) message in the least significant 1 or 2 bits of the chirps
  - Changing the LSb or two of each chirp will not be detectable by a human listener

© 2013 W.P.Lidinsky

3. Generation

- Where data is hidden
  - In the overt file as part of the overt file
- How data is hidden
  - Algorithmic scheme is used to hide data in the overt file
  - Covert file algorithm and values create the overt file
- The file size does **not** increase when covert data is added since the covert data is designed to be a part of the file

© 2013 W.P.Lidinsky

A Very Simple Text Example

- Send the number 57 hidden in a carrier text file.
- The carrier with 57 hidden in it is:
  - Hi Annette. I saw the 2D graphics for your project. They were great. But I suggest, since you have the knowledge of graphics, that you do a 3D version.
- Any idea how 57 is hidden in the above message???

© 2013 W.P.Lidinsky

A Very Simple Text Example

- Hints:
  - Start at the beginning of the message:
    - A period yields a “1”
    - A comma yields a “0”

    Hi Annette. I saw the 2D graphics for your project. They were great. But I suggest, since you have the knowledge of graphics, that you do a 3D version.

  - What is our binary value???

© 2013 W.P.Lidinsky

A Very Simple Text Example

- How do we get 57 from 111001???
  - $57_{10} = 111001_2$
  - Base10 Decimal to Base 2 Binary:
  - 1 1 1 0 0 1
  - 32 16 8 4 2 1
  - 32+16+8+0+0+1 = 57

Generation Example

- In the simple "Hi Annette" example that I provided earlier:
  - I knew that I needed to hide the binary number 57
  - I devised an algorithm to carry the information in an overt message
  - I then created the message to carry the hidden number

© 2013 W.P.Lidinsky

Part III

Graphic Image Carriers

Graphic Image Carriers

- Hiding a covert file inside of a carrier file that is an image
- Most covert method is to use Substitution
  - Stego application modifies the existing low order or redundant color information bits in the carrier file to contain the covert data
  - An individual viewing the original carrier and the modified carrier (with the covert data embedded) should not notice a visual difference

Taxonomy

- Graphic image formats are usually classified into one of three types:
  - Vector map
    - Images defined in terms of lines, rectangles, triangles, circles and other geometric shapes
  - Fractal maps
    - Images generated by starting with a simple geometric shape and replicating that shape but changing its size and orientation
  - Pixel map
    - Image comprised of a 2D array of pixels (picture elements) which are easily rendered on a TV, Computer Monitor, etc.

© 2013 W.P.Lidinsky

Pixel Map

- We will only cover the Pixel Map in this class.

Pixel Resolution

- Let's say we have an image with a resolution of 1024 x 768

- What does that mean?

Pixel Resolution (Cont.)

- 1024 pixels wide
- 768 pixels high
- 1024 * 768 = 786,432 pixels or about 0.8 Megapixels

Pixel Map

- Each pixel has visual properties
  - RGB (**r**ed, **g**reen, **b**lue)
  - CYMK (**c**yan, **y**ellow,**m**agenta, blac**k**)
  - HSI (**h**ue, **s**aturation, **i**ntensity)
  - Y,Cb,Cr (brightness, blue adjust, red adjust)
    - No gamma correction in Y
  - Y',Cb,Cr (brightness, blue adjust, red adjust)
    - Gamma correction used
- Usually each pixel consists of 8, 16, or 24 bits

© 2013 W.P.Lidinsky

Image File

- An image file is a binary file that represents the:
  - Color or light intensity of each pixel in the image
  - An image can have various bit depth:
- 24-bit:
  - Red = 256
  - Green = 256
  - Blue = 256
    - 256 x 256 x 256 = 16.7mil colors
- 8-bit:
  - Red = 8
  - Green = 8
  - Blue = 4
    - 8 x 8 x 4 = 256 colors

http://www.garykessler.net/library/steganography.html

24-Bit RBG Example

	Red	Green	Blue
Pixel 1:	10011010	01101010	10111100
Pixel 2:	01101010	11001100	10101010
Pixel 3:	11001011	10100110	00011010
	8-bits	8-bits	8-bits

Hiding Data: Substitution Method

- Now let's say we want to insert nine bits of hidden data which is 101110111.
- We will substitute one bit from each of the colors of the three pixels.
- Let's look at the 8 bits of Red for the first pixel
  - 10011010
- We need to substitute a "1" in there somewhere where it won't be noticeable.
  - Any ideas where to put it???

Hiding Data: Substitution Method

- Put it in the Least Significant Bit (LSb)!
- Let's look at the 8 bits of Red for the first pixel
  - 1001101**1**
- Why should we put the value in the LSb?
  - Least likely to show a visual change to the viewer since the change would be the most minimal

24-Bit RBG Example with Hidden Data Substitution

Hidden Data to Substitute: 101 110 111

	Red	Green	Blue
Pixel 1:	10011011	01101010	10111101
Pixel 2:	01101011	11001101	10101010
Pixel 3:	11001011	10100111	00011011
	8-bits	8-bits	8-bits

S-Tools Features

- Small, portable Stego tool
- Supports bmp, gif, and wav carrier files
- LSb substitution method to hide covert file in the carrier file.
- Lossless compression
- Symmetric key encryption for added secrecy

In-Class Lab: S-Tools

- Open the "Stego In-Class Labs" folder you previously placed on your desktop
- The Secretmessage.txt file will be our covert message
- The splash.bmp file will be our carrier file
- Notice the files sizes and open both files and look at them
- You can now close both of those files

In-Class Lab: S-Tools

- S-Tools employs LSB substitution method to hide the covert file in the carrier file.
- In the “Stego In-Class Lab” folder, open the S-Tools folder, and execute S-Tools application (not the help file)
- Select “Continue”
- Go back to Windows Explorer and back up a directory to the “Stego In-Class Labs” folder

In-Class Lab: S-Tools

- Drag the splash.bmp file into S-Tools
- How much covert data can this carrier file hold???

- We want to put the Secretmessage covert file (71,432 bytes) into this carrier file. Can we do it???
- Let's try it.

In-Class Lab: S-Tools

- Drag the Secretmessage.txt file into the splash.bmp image open in S-Tools
- The text file shows it will be added as only 5,601 bytes (not 71,432 bytes)
  - How can that happen???

In-Class Lab: S-Tools

- Set the Passphrase as test and verify it
- Keep the added encryption algorithm as IDEA
- Click “OK”
- Now, you should see the original image and the image with the hidden data.
- Do the two images look any different to you???

In-Class Lab: S-Tools

- Select the “hidden data” image, right-click on it, and select “Save-As”
- Save the file as hidden.bmp up a directory in the “Stego In-Class Labs” folder
  - Don’t forget the .bmp extension
- Open the “Stego In-Class Labs” folder and you should see the original splash and hidden files.

In-Class Lab: S-Tools

- Are they the same size?
- If so, why are they the same size?

In-Class Lab: S-Tools

- They are the same size because:
  - The data simply replaced the LSB which allows the new file (with the hidden data) to stay the same size as the original file
  - No additional data was added to the file
- Let's look at this a little further.
- Open the WinHex folder and execute WinHex.exe application (not setup.exe)
- Close any notification messages that may appear

In-Class Lab: S-Tools

- Drag the splash.bmp and hidden.bmp files from the “Stego In-Class Labs” folder into WinHex
- Look at each tab
- Do they have the exact same file size???

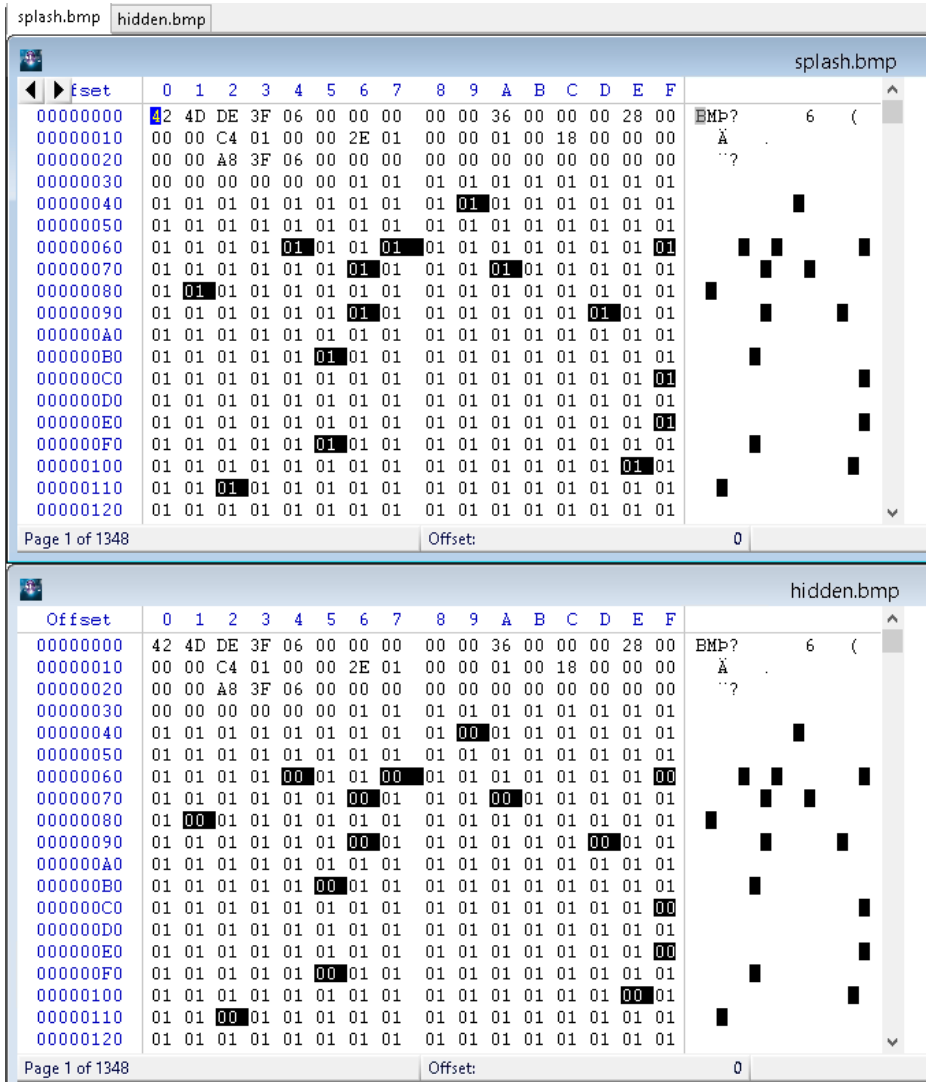
In-Class Lab: S-Tools

- Now let's compare the two files
- Make sure both files are at the top where you see the BMP header

- In WinHex, select View and place a check in Synchronize & Compare
- Select "OK"

In-Class Lab: S-Tools

- You should now see this:



In-Class Lab: S-Tools

- WinHex is showing what representation of these two files???
- The black highlighting shows anywhere that the two files differ
- If there is not black highlighting, the files are the same

In-Class Lab: S-Tools

- On the splash.bmp file in WinHex, scroll down (which should cause both file windows to scroll)
- You can see that the covert file has been distributed evenly across the LSB in the carrier file.
- If you copied all of the ASCII text value differences from the hidden file, could those values be combined to instantly display the text of the original Secretmessage covert file???

In-Class Lab: S-Tools

- They could not because S-Tools used compression as well as encryption to further obscure the hidden message.
- Go ahead and close WinHex
- Go back to the S-Tools window and select the hidden.bmp file, right-click, and select “Reveal”
- Enter the passphrase as test in both spots and select “OK”

In-Class Lab: S-Tools

- You should see the “Revealed Archive” window showing the extracted covert file
- Select the file, right-click, and select “Save-As”
- Save the file in the “Stego In-Class Labs” folder as extractedmsg.txt
- Now open the extractedmsg.txt file from the “Stego In-Class Labs” folder

In-Class Lab: S-Tools

- Does the extracted message look the same as the original???
- You can close S-Tools and Windows Explorer

Invisible Secrets Exploratory Lab

- In the Stego In-Class Labs Folder, install invsecr43 with defaults
- Use .\student and student for UAC
- Your goals are to:
  1. Hide the Secretmessage.txt file covert file inside of the splash.bmp carrier file with Invisible Secrets and create a new carrier called splashnew.bmp
  2. Extract and view the covert file from the splashnew.bmp file
  3. Determine if Invisible Secrets uses Insertion or Substitution

Part IV

Audio Carriers

Common Stego Schemes – Audio File Carrier

- Modify the carrier audio file slightly to contain a hidden text message, image or another sound file
  - Put covert info into unused bits in the audio file format
  - Modify low order bits of audio samples
  - Include covert file in audio that is inaudible to humans or is not in the frequencies reproduced by the audio playback
- A person listening to the carrier image cannot detect that a hidden message exists

© 2013 W.P.Lidinsky

Audio Carriers

- Common formats:
  - wav
  - mp3

Some wav File Format Info

- ***wav*** files store audio a sequence of digitized audio samples
- Each sample is a instant of sound represented by a digital value
- Several formats and variants
- Most common format consists of
  - Audio sample wrapper describing details about the sampling & number of channels for rendering codecs
  - Sequence of samples are created using uncompressed Linear Pulse Code Modulation (LPCM)
  - 16 bits per sample, 44,100 samples per second

© 2013 W.P.Lidinsky

Some wav File Format Info

- ***wav*** files can be very large
  - 88,200 Bytes per second of audio uncompressed
    - 44100 samples/second x 2 bytes/sample
  - Smaller samples, slower sample rates and compression can bring this down to as low as 3,000 Bytes per second of audio
    - Fidelity is lower. OK for speech but not for music
- But large size allows much covert information to be hidden in a wav carrier
  - e.g., (2 bits / sample) x (44,100 samples / second)
  - = 88,200 bits per second ≅ 11 KBytes per second of audio

© 2013 W.P.Lidinsky

wav File Stego

- Distribute the hidden (covert) message in the least significant 1 or 2 bits of the sample
- Changing the LSb or two of each sample
  - Will not be detectable by a human listener
  - Will not increase the file size
  - Can store a lot of covert data
  - e.g., (2 bits / sample) x (44,100 samples / second)
  - = 88,200 bits per second ≅ 11 KBytes per second of audio

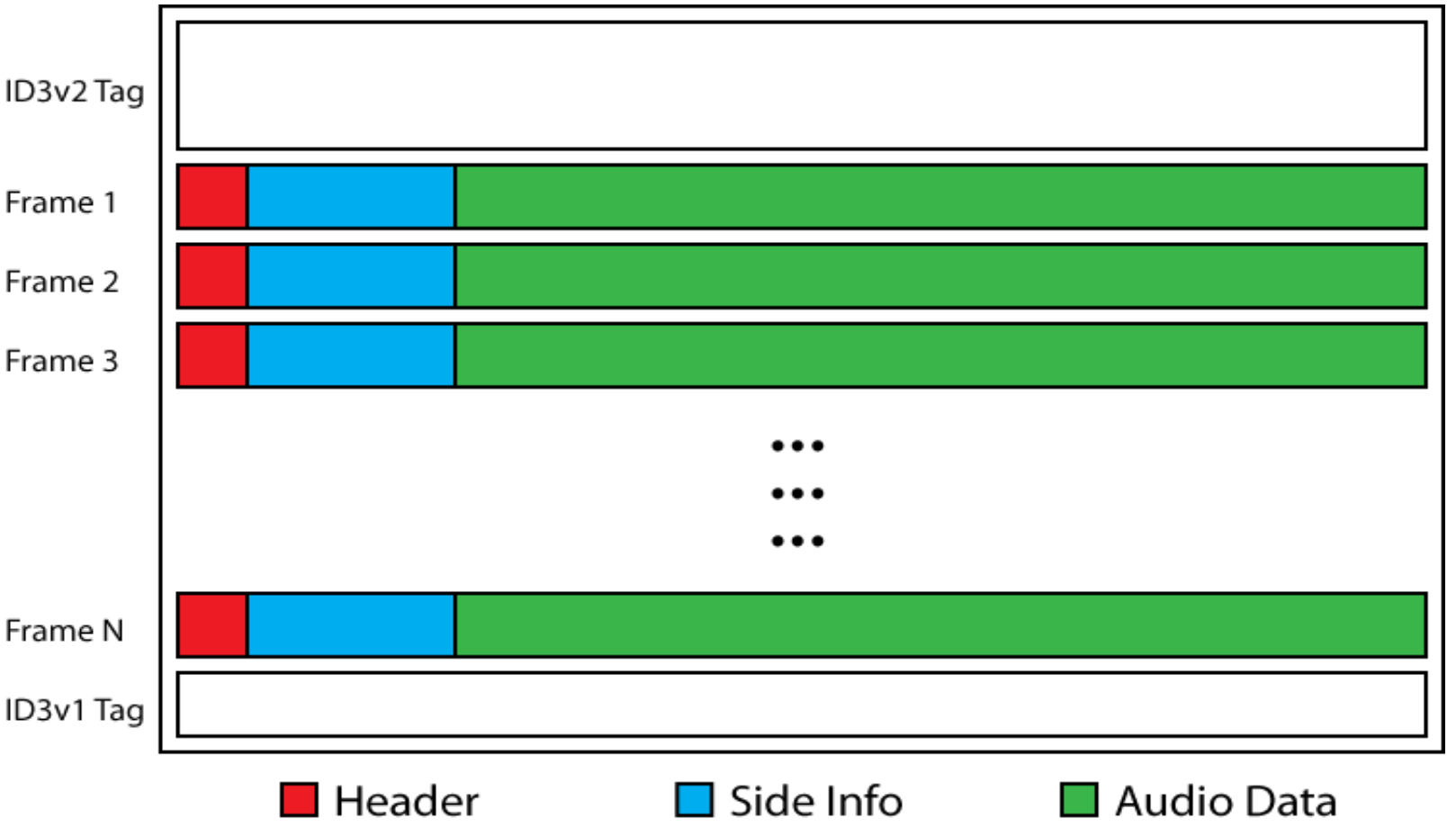
© 2013 W.P.Lidinsky

mp3

- Unlike *wav*, *mp3* uses lossy compression to achieve small file sizes
- Although the compression is lossy (discards some fidelity), rendering mp3 files results in excellent audio

© 2013 W.P.Lidinsky

Some mp3 File Format Info

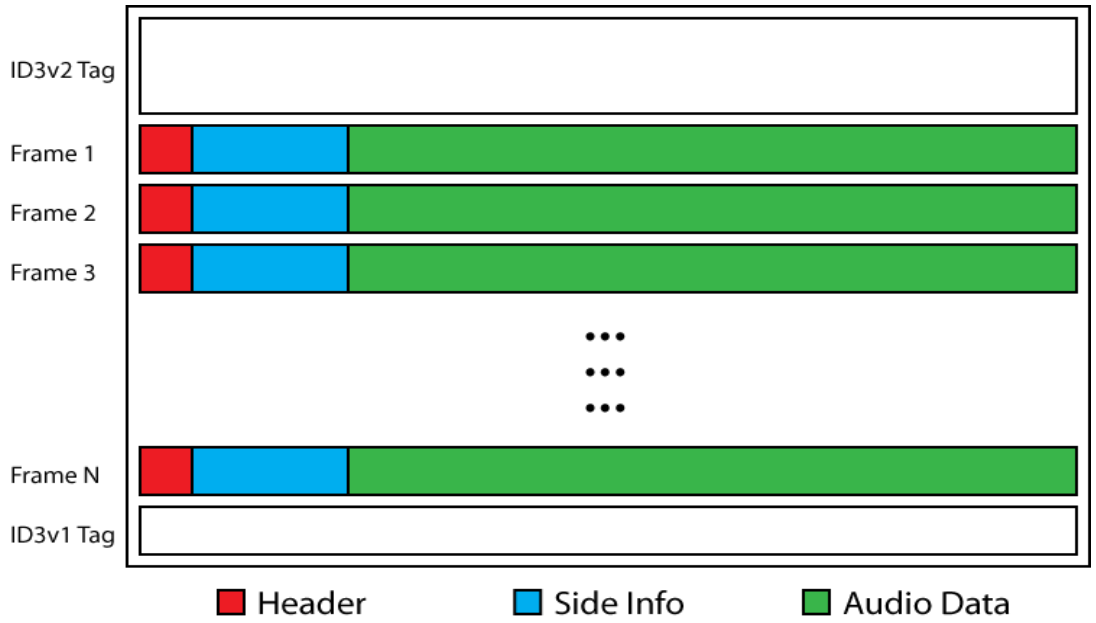


Ref: Mikhail Zaturenskiy, *MP3 Files As A Steganography Medium*

© 2013 W.P.Lidinsky

mp3 File Stego

- So where can covert files be hidden in mp3 files?
  - In the **Audio Data** like wav files?
  - In the **ID Tags**?
  - In the **Frame Headers**?
  - In the **Side Information**?
    - Difference between left and right channels
  - In **Empty Frames**?



© 2013 W.P.Lidinsky

Some mp3 Stego Tools

- Mp3stego
  - Fabien A.P. Petitcolas
    - Cambridge (1998)
    - Inserts data during compression process
    - Must start with a *wav* file
- Mp3stego (java)
  - Lukasz Maciak, Micheal Ponniah, Renu Sharma
    - Montclair State University (2005)
    - Padding byte stuffing attempt

© 2013 W.P.Lidinsky

Some mp3 Stego Tools

- Mp3stegz
  - Achmad Zaenuri
    - Diponegoro University (2008)
    - Uses empty frames
- Hiderman
  - Puts the covert file at the end of the mp3 file after the EoF
- mp3 Stegazaurus*
  - Uses all possible parts of mp3 file format except for Audio Data
  - The best mp3 stego tool yet
  - Student ForenSecure project

© 2013 W.P.Lidinsky

Part V

Alternate Data Streams

Alternate Data Streams (ADS)

- Introduced in Mac HFS and then Windows NTFS file systems
- File data stored in two forks:
  - Data Fork
    - Holds Data
  - Resource Fork
    - Tells OS how to use the Data
    - Example: IE adds information about where a file was downloaded
    - Data can be hidden in here!
- ADS are a great way to hide data!!

http://nest.unm.edu/index.php/download_file/view/47/156/

Alternate Data Streams (ADS) In-Class Lab

- Open a Windows Command Prompt
- Change to your Documents directory
- Enter:
  - **echo "This is a normal file" > file1.txt**
  - **echo "This is a secret file" > file1.txt:hidden.txt**
- Now, use the **dir** command and which file do you see and how large is it???
  - file1.txt which is 26 bytes

Alternate Data Streams (ADS) In-Class Lab

- Let's view both of the files.
- Enter:
  - **more file1.txt**
  - **more < file1.txt:hidden.txt**
    - Reveals hidden.txt message hidden within file1.txt

Alternate Data Streams (ADS) In-Class Lab

- Now let's open both of the files in notepad.
- Enter:
  - **notepad file1.txt**
  - **notepad file1.txt:hidden.txt**
- In the file1.txt:hidden file that is open in notepad, add a sentence or two and save the file.
- Run **dir** and did the file size of file1.txt change???
  - It did not

Alternate Data Streams (ADS) In-Class Lab

- Now, type a few sentences in the file1.txt file and save it. Did the file size of this file change???
  - It did
- This is an interesting way to hide data, because the carrier file will not show a size change when hidden data is added

Alternate Data Streams (ADS) In-Class Lab

- Even though the carrier file size doesn't not change when adding text to the covert file, there is one change that may be noticed.
- Take some time to determine what change occurs.
- Remember, you can use the up arrow to bring up your previous commands.
- What changed?

Alternate Data Streams (ADS) In-Class Lab

- It is also possible to embed an application within an application.
- Great way to hide malware!

- Any idea how to detect if a hidden stream is in a folder???
  - Windows Vista+ added **dir /r**
  - Can also use the Sysinternals Streams application

Part VI

Network Carriers

Covert Communications

- So far we have been hiding covert files within a carrier file
- We can also hide messages within TCP/IP headers using an application called Covert_TCP
- This is a potential way for attackers to hide their transmissions outside of a breached network and could potentially be used for C2 communications

Covert_TCP Lab

- Open a browser in Kali
- Browse to: bit.ly/1puX4MA
- Hit the Download button
- Select Save File / OK
- Open a terminal
- Change to the Downloads directory
- Verify the covert_tcp file is now in that directory

Covert_TCP Lab

- Covert_TCP can be used as a receiver as well as the sender
- Both the receiver and sender would need to have the application on their computers
- For ease of the demo, we are going to do the receiving and sending both in Kali on the same system

Covert_TCP Lab

- Exploratory option for advanced students
  - Don't pay attention to the next slides
- Create a covert file with a short message.
- Use 3 terminals
  - Use covert_tcp as a listener server on port 7777 and receive the message to a file
  - Setup a tcpdump listener on localhost port 7777 and use the ASCII output print to screen option
  - Use covert_tcp to send your covert message to the listener
- Determine which byte in the header contains the hidden message characters

Covert_TCP Lab

- In Kali, go back to your terminal which should have the Downloads directory open
- Is covert_tcp executable?

- **chmod u+x covert_tcp**

Covert_TCP Lab

- Now we are going to set up the receiving listener

./covert_tcp –dest 127.0.0.1 –source 127.0.0.1 –source_port 7777 –dest_port 8888 –server –file receivedmsg.txt

- Hit Enter and the receiver will start listening

Covert_TCP Lab

- Open another terminal that will use tcpdump to monitor the received traffic on the interface

  tcpdump –nvvX port 7777 –i lo

Covert_TCP Lab

- Open a third terminal which will be the sender of the covert message
- Change directories to the Downloads directory if not already in there
- Create this secret message:

echo “Proceed” > transmit.txt

Covert_TCP Lab

- Put all three terminals where you can see them
- Run the command below to send the file covertly to the receiving listener (first window)

./covert_tcp –dest 127.0.0.1 –source 127.0.0.1 –source_port 8888 –dest_port 7777 –file transmit.txt

- Watch all three windows after hitting Enter

Covert_TCP Lab

```
root@KLY-IR105:~/Desktop# echo "Proceed" > transmit.txt
root@KLY-IR105:~/Desktop# ./covert_tcp -dest 127.0.0.1 -source 127.0.0.1 -source
_port 8888 -dest_port 7777 -file transmit.txt
Covert TCP 1.0 (c)1996 Craig H. Rowland (crowland@psionic.com)
Not for commercial use without permission.
Destination Host: 127.0.0.1
Source Host     : 127.0.0.1
Originating Port: 8888
Destination Port: 7777
Encoded Filename: transmit.txt
Encoding Type   : IP ID

Client Mode: Sending data.

Sending Data: P
Sending Data: r
Sending Data: o
Sending Data: c
Sending Data: e
Sending Data: e
Sending Data: d
Sending Data:
```

```
root@KLY-IR105:~# cd Desktop/
root@KLY-IR105:~/Desktop# ./covert_tcp -dest 127.0.0.1 -source 127.0.0.1 -source
_port 7777 -dest_port 8888 -server -file receivedmsg.txt
Covert TCP 1.0 (c)1996 Craig H. Rowland (crowland@psionic.com)
Not for commercial use without permission.
Listening for data from IP: 127.0.0.1
Listening for data bound for local port: 7777
Decoded Filename: receivedmsg.txt
Decoding Type Is: IP packet ID

Server Mode: Listening for data.

Receiving Data: P
Receiving Data: r
Receiving Data: o
Receiving Data: c
Receiving Data: e
Receiving Data: e
Receiving Data: d
Receiving Data:
```

Covert_TCP Lab

- Hit CTRL-C on the first receiver window to stop the listener.

  cat receivedmsg.txt
- Check out the tcpdump window
- CTRL-C on the tcpdump window as well
- The ASCII representation of the data is shown on the right side of each packet
- In what byte is “Proceed” being transferred???

Covert_TCP Lab

```
root@KLY-IR105:~# tcpdump -nvvX port 7777 -i lo
tcpdump: listening on lo, link-type EN10MB (Ethernet), capture size 65535 bytes
04:36:12.208314 IP (tos 0x0, ttl 64, id 20480, offset 0, flags [none], proto TCP
 (6), length 40)
    127.0.0.1.8888 > 127.0.0.1.7777: Flags [SW], cksum 0x6a2b (correct), seq 689
43872, win 512, length 0
        0x0000:  4500 0028 5000 0000 4006 2cce 7f00 0001  E..(P...@.,.....
        0x0010:  7f00 0001 22b8 1e61 041c 0000 0000 0000  ....".."a........
        0x0020:  5082 0200 6a2b 0000                      P...j+..
04:36:12.208355 IP (tos 0x0, ttl 64, id 17128, offset 0, flags [DF], proto TCP (
6), length 40)
    127.0.0.1.7777 > 127.0.0.1.8888: Flags [R.], cksum 0x6c98 (correct), seq 0,
ack 68943873, win 0, length 0
        0x0000:  4500 0028 42e8 4000 4006 f9e5 7f00 0001  E..(B.@.@.......
        0x0010:  7f00 0001 1e61 22b8 0000 0000 041c 0001  .....a".........
        0x0020:  5014 0000 6c98 0000                      P...l...
04:36:13.209745 IP (tos 0x0, ttl 64, id 29184, offset 0, flags [none], proto TCP
 (6), length 40)
    127.0.0.1.8888 > 127.0.0.1.7777: Flags [SW], cksum 0x223c (correct), seq 127
5789312, win 512, length 0
        0x0000:  4500 0028 7200 0000 4006 0ace 7f00 0001  E..(r...@.......
        0x0010:  7f00 0001 22b8 1e61 4c0b 0000 0000 0000  ....".."aL.......
        0x0020:  5082 0200 223c 0000                      P..."<..
04:36:13.209769 IP (tos 0x0, ttl 64, id 17256, offset 0, flags [DF], proto TCP (
```

Covert_TCP Lab

- We had used this command:

  tcpdump –nvvX port 7777 –i lo

- What do the following options mean? (Hint, use the man page for tcpdump)
  - -n
  - -vv
  - -X
  - port 7777
  - -i lo

Covert_TCP

- If you are interested in checking out the source code of covert_tcp, it can be found here:
- http://www-scf.usc.edu/~csci530l/downloads/covert_tcp.c
- You could then save it as covert_tcp.c and compile it with:

gcc –o covert_tcp covert_tcp.c

Part VII

Detection of Covert Data (Steganalysis)

Detection

- Called "***steganalysis***"
- Detection of overt (carrier) files with hidden (covert) information is often very difficult
- Extraction of the covert info can be even more difficult
- Detection is much easier if you can compare the suspect overt file to the original overt file that is known not to be a carrier

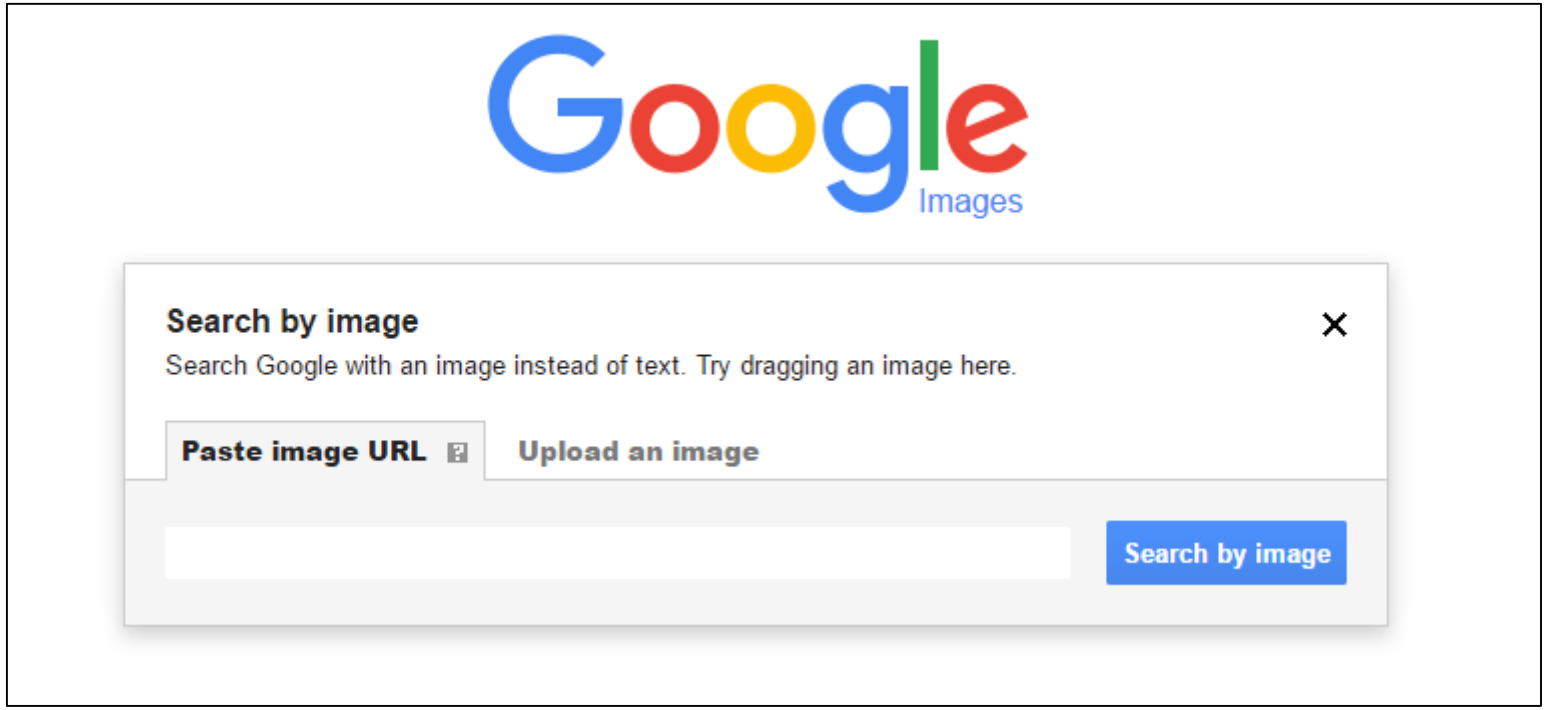
© 2013 W.P.Lidinsky

Detection (Cont.)

- Let's say you were given an image suspected of containing stego
- How might you go about finding the original carrier/overt image (prior to covert data being added)???

Detection (Cont.)

- Computer forensic examination of drive for original overt file
- Google Image Search



Detection (Cont.)

- Note: Google image search may show you several different resolutions or edits of the carrier file which may make comparison against the covert file difficult

- Any thoughts on how to detect Stego??

General Steganalysis Detection Techniques

- Statistical detection of changes in patterns of the carrier file (LSB, etc.)
- Could be visible or audible differences
  - Could use histogram to look for patterns
- Take a hash of original known good file and compare it against the potential stego file
- Size differences in file
- Look for signature that stego program uses
  - Hiderman places “CDN” at end of file (footer)

Additional Resources to Check Out

- http://www.wetstonetech.com/technicalpapers.html
- http://www.jjtc.com/Steganography/tools.html

- You will have a Stego lab this week

ITMS 539

- This lecture was just a brief dive into the basics of Steganography.
- If you are interested in this topic, ITMS 539 is a full course on Steganography offered by Bill Lidinsky in the Spring semesters.

Checkpoint

- What is the difference between an overt/carrier file and a covert file?
- Can any type of covert file usually be embedded into a carrier file?
- How do the following methods of getting covert info into carrier files work and do they increase the file size?
  - Insertion
  - Substitution
  - Generation

Checkpoint

- Where did we substitute data for the Pixel Map example?
- Which hiding method did S-Tools and Invisible Secrets use?
- Where can covert info be hidden in mp3 files?
- How do ADS work?
- How did Covert_TCP hide data?

Homework

- See end of Firewall lecture which is next.