A Discussion of Covert Channels and Steganography

Mark Owens
March 19, 2002

SANS/GIAC GSEC 1.3

00000000011101000000000000000000

Abstract:

For those whose task it is to assure security, electronic communication without scrutiny that can continue between parties trying to circumvent a security policy poses a risk. This is true whether the "parties" are individuals with malicious intent or processes with different security levels passing data while running on a shared CPU. Covert communication channels have been present throughout history and continue to be developed, used and sometimes exploited by those intent on keeping their communications not only secret, but also hidden. Awareness has increased in the application of covert channels, fueled by increased demand for development in the field of digital watermarking and fingerprinting for copyright protection and infringement prosecution, and reports of increased use for illegal purposes on the Internet. The technology itself is a double-edged sword.

Although the current threat of steganographic technology appears to lag its usefulness, the diligent information systems person needs to be mindful of the security ramifications that a covert channel in their enterprise carries. Myriad techniques for secreting information flow exist; eliminating them is an impossible task. In order to protect ourselves, we must apply the dynamics that serve so well in other unpredictable information security risk venues: assess the risk, find ways to quickly detect the exploit's use, determine an appropriate response and use whatever means available to impede the perpetrator allowing time for detection and reaction mechanisms to work.

And learn from the other guys. . . .

00000000011011110000000000000000

<u>A Discussion of Covert Channels and Steganography</u>

Mark Owens
March 19, 2002

A covert channel.

The thought is certainly not comforting to anyone who is responsible for assuring availability, integrity and confidentiality of data in an information system. IS people are happiest when everything is above board, and/or accessible and understood. The idea that information may well be passing *through the very domain that it is our charge to protect*, its content at best we can only guess about, is anathema. It could keep some people from sleeping.

According to a 1985 U.S. Department of Defense publication titled "Trusted Computer System Evaluation", a covert channel is defined as:

> ". . . any communication channel that can be exploited by a process to transfer information in a manner that violates the system's security policy."

It continues, describing two categories of covert channels, storage channels and timing channels:

> "Covert storage channels include all vehicles that would allow the direct or indirect writing of a storage location by one process and the direct or indirect reading of it by another. Covert timing channels include all vehicles that would allow one process to signal information to another process by modulating its own use of system resources in such a way that the change in response time observed by the second process would provide information." [1]

Storage channels will be the focus of the remainder of this text. Not that timing channels lack importance, far from it. But for most people working in information security, storage channels pose a greater (or at least more obvious) risk. This is partly due to the comparative communication bandwidths, and the availability of tools that allow non-engineering types the opportunity to exploit covert storage channels. The increased computer industry and news media coverage that storage channels (by many names) have been getting in years past also tend to make them more interesting or immediate.

The fact that a communications channel is covert literally means that it is hidden. This implies that a "third party" does not know of *even the existence* of such a channel. One of the most common and perhaps the best vehicle for discussing the dynamics of covert communications is found in what is known as the "*prisoners' problem*". Initially posed by G.J. Simmons in 1983, it involves two prisoners and a warden [2]. The prisoners, Alice and Bob [a], who wish to devise an escape plan, need to communicate with each other. The warden, here named Wendy, oversees all inter-prisoner communications. The warden can handle her job as monitor of all prisoner communications in one of two ways:

1) She can review all messages, and pass or deny them based on what she sees (be 'passive') or,
2) She can modify the message slightly, not to change the meaning of the message itself, but to make sure it's not precisely what was sent. It is assumed that in so doing, she might thwart any attempt to pass a message that could be subtly embedded in the cover communication (be 'active') [b].

Ideally, the prisoners find a means to communicate in such a manner that suspicion on the part of the warden is never raised. But the warden, faced with the possibility that Alice and Bob *might eventually* wish to discuss something prohibited, must accept that there is a risk that some covert communication may be attempted. To counter this threat, and after accepting that *some form of covert communications channel possibly exists,* she must pose a hypotheses regarding how it might function. This alone is not cause for denying all communication; no rules have necessarily been broken, as at this stage, the covert channel *is theoretical*. The following and much more difficult step would be to determine that this covert channel was indeed being purposefully *used*.

In a practical application, for any given moment in time, prudent Information Security diligence requires that we accept the existence of covert channels in our environment [c]. Our tasks are to mitigate the risk that they introduce. The difficulty in identifying what these covert channels *are* comes in the fact that by nature, most of these communication paths are not mechanisms that were never intended to be used to convey information at all.

Methodologies

Recently, accounts have surfaced of suspected *secret, hidden conveyance* of plans or instructions via the Internet to terrorist groups operating within the U.S. It is believed by many that these encrypted digital messages were (and perhaps are still) passed by way of covert channels, embedded within other innocent-looking files [Maney 3]. For the purpose of National security, agencies of the U.S. government have amassed powerful and sophisticated systems and groups of people to intercept communications sessions, inspect content and react accordingly when suspect communications content is detected. As concerns about the risks to security that open electronic communications systems carry has increased, so has the concern for confidentiality of individual communications. A result of the latter has been a multi-fold increase in the use of data encryption. This in turn increases the struggle between the two factions desiring the proliferation of strong encryption capabilities (e-commerce and personal liberty), and the restriction of such capabilities (law enforcement and other government agencies). The FBI's *Clipper* system and *Carnivore* [d], and the NSA's *Echelon* are but three examples of how committed the government is to continuing their work in these areas. Meanwhile, an indeterminable amount of communication continues to take place undetected.

I believe it important here to make a distinction between *cryptography*, and one of its sub-disciplines, *steganography*. As stated in the early pages of his book Information Hiding – techniques for steganographic and digital watermarking, Fabien Petitcolas writes, "While cryptography is about protecting the content of messages, steganography is about concealing their very existence. . . . [Steganography] is usually interpreted to mean hiding information in other information"[4]. David Hughes provides a definition for a secure steganographic system as "a system where an opponent who understands the system but does not know the key can obtain no evidence (or even grounds for suspicion) that a communication has taken place" [5].

Digital steganography methods can be classified in three distinct modes: *injection, substitution* and for the third I coin the term *propagation* (otherwise known as "generating a new file"). The first two, and often the third type utilize specific *bit-locations* as the covert channel for communications. And most utilize a *stego-key,* which provides control for the hiding and recovery processes, preventing or restricting detection by those who are not aware of the key, or do not have access to it.

Injection steganography works as might be expected, in that the *payload* or *embedded data* is placed inside the original (unaltered) host *cover-text, cover-image, cover-audio* or *cover-program* file. Doing so increases the host file size, and the process must be done in such a manner as to prevent the end-processing or presentation application (word processing program, picture viewer, music player, etc.), from revealing the presence of the embedded data within the cover. Most file types are susceptible to injection steganography. The file resulting from this process, or any other steganographic methodology is often referred to as the *stego-text, stego-image (-audio, etc.)* file, or more generically, a *stego-object.*

Substitution steganography *replaces* what is viewed as an insignificant part of the cover file, but also must survive when processed by any "native" application such as those listed above. The substituted portion of an executable cover file could be a program module or segment of executable code that is rarely or never used. This method (sometimes referred to as "bit-twiddling" or "bit-tweaking") can result in file degradation such as aberrations in video or still images, audible noise in sound files or in the case of executables, processing errors or abends.

Propagation steganography most often utilizes a generation engine which when fed the payload produces an output file. (It is possible to do it manually using a lookup table when the stego-object will be text). The content of this file, sometimes referred to as a "mimic", may appear as a freeform graphic, a music file, a verbose text document, a fractal image or some other form. When reprocessed by the generation engine, with few exceptions, a given payload will yield the same stego-object file. There is no host file or cover-object involved in this method [e].

To the unaware, and those without detection mechanisms, injection steganographic methods probably pose the greatest risk. Many forms of malicious code are distributed through some variation of injection. Rita Summers in her book Secure Computing: Threats and Safeguards defines a Trojan horse as an "apparently useful program containing hidden functions that can exploit the privileges of the user, with a resulting security threat. A Trojan horse does things that the program user did not intend" [6] [f].

History

The covert channel or conduit used in a steganographic communication could in actuality be quite a variety of things. According to the writings of a Greek historian (*Herodotus (c.486-425 B.C.E.)* *Histories*), a messenger's head was shaved and tattooed with a secret message calling for revolt against the Persians. Later, after the hair had re-grown, the messenger traveled to the location of the intended receiver. The message was then revealed to the recipient as the head was again shaved [Petitcolas 7]. The message, or payload was secreted within an unseen conduit, a head of hair. An *acrostic* would likewise qualify as a covert channel, or more rightly, the positions held by first letters, middle letters or last letters (depending on the agreed-upon location of the channel in the acrostic itself). Likewise qualifying would be the physical space immediately above printed letters onto which during WWII miniscule dots of invisible ink were sometimes placed. These signified to the recipient the characters that were pertinent in the encoded message stream. (This was a derivative of a much earlier technique using pinpricks.) There are very old texts documenting a variety of message hiding schemes including the use of a musical scores (*Gaspar Schott (1608-1688)* *Schola Steganographica*), a mechanism for generating *psudo-logical texts*, which insert a word or phrase that somehow correlates to a specific letter in the original message (*Johannes Trithemius (1462-1516)* *Steganographiae*), and many others. In the recent century prisoners of war are known to have used the dots and dashes in letters such as "i & j" and "f & t" to convey Morse-coded messages [Petitcolas 7]. All these, the head of hair, the letter positions in the acrostic, the areas just above (below or otherwise) or the characteristics of specific letters-- they all constitute covert channels, by virtue of an *agreed-upon arrangement by the communicating parties*.

The "arrangement" is what makes steganography work. Would you have ever noticed the hidden images in the "Magic Eye" prints developed in the early '90s had no one been told by their creators to look at each one for a while? Without this preconditioning arrangement, you could not have known that there, in what looked much like a scrap of gift-wrapping paper, was a hidden stereogram.

Consider a two-dimensional barcode image. In theory, recent symbol designs can store as many as three thousand characters in a block much smaller than a postage stamp. One current developer of 2-D barcode claims that error correction techniques allow for full data restoration when up to 50% of the image is destroyed. That requires a fair amount of overhead. Suppose we assume that rarely would a full restoration be necessary from half the image. So instead of having *quite* so many bits dedicated to redundancy, (and knowing the restoration algorithm of course), we could "borrow" a select number of bit positions and insert other data. This would probably work because the normal scanning software would likely be written to ignore the redundant parity bits unless needed. The embedded data would be readable only by someone who had software altered to to read the selected "parity" data as a discrete data stream.

TCP/IP Stego

Similar schemes exist for secreting data within the header of a TCP/IP packet. Basic TCP/IP packet archetecture allows for a number of covert chanel options by way of numerous locations within packets which are normally unused or optional. By prior arrangement between Alice and Bob, Bob's computer (referred to as Bob for this example) could receive packets from Alice's computer (Alice) that looked quite normal unless very carefully scrutinized. By extracting predefined bits or blocks of bits from specific locations within a series of packets, Bob could easily reassemble a hidden ASCII message from Alice. (Although the numbers of packets needing to be sent to convey a given message using *stego-TCP* might need to be large due to its low bandwidth, the time required is still in the multi-millisecond range.)

Manipulating the 32-bit sequence number is a popular option due to its field size allowing for over four billion numbers. Deriving an initial sequence number (ISN), from an ASCII base-component allows for the receiving port, assuming it's listening and aware of the mechanism, to extract the encoded character (or multiple characters, depending on the ISN generating algorithm). In this scenario the SYN sent by the originating computer as the first part of a socket set-up sequence is the only packet in the session to carry embedded data. Each message character sent will likewise be part of a new handshake sequence, as that is when new ISNs are exchanged.

In this simple example, Alice wants to send the message "*tonight*" to Bob. The exchange would start as all TCP/IP communications do, with a SYN sent to one of Bob's listening ports to begin setting up a socket, and beginning a new session.

- 1st "encoded" SYN packet - Using an ISN of, say, *7602176*, a three-way handshake is started by Alice. Bob's receiving port (which is listening and "aware" of the embedding scheme) decodes the encoded ISN by dividing the ISN by a *prearranged divisor of 65,536* – yielding the ASCII value of 116 (or '*t*'). The handshake would continue, either: a) establishing the socket by Bob sending a SYN/ACK and getting an ACK from Alice (with the two perhaps employing some superficial data transfer until the session is closed cleanly); or b) being aborted by Alice sending Bob an RST after receiving Bob's SYN/ACK.
- 2nd encoded SYN packet - Alice sends another SYN to Bob, this time with an ISN of *7274496*, which Bob decodes as an ASCII value of 111 (or '*o*'). The sequence continues as above.
- 3rd through 7th encoded SYN packets - The remaining five initialization SYNs from Alice, each following the previous scenario, have the ISNs of *7208960* ('*n*'), *6881280* ('*i*'), *6750208* ('*g*'), *6815744* ('*h*') and *7602176* ('*t*'), respectively.

To a trained observer, there have been seven sockets set up and torn down between Alice and Bob, but if no data was transmitted it could be considered strange. However, sending innocuous data like a seven-part exchange planning a birthday party for Wendy the wonderful warden could have easily masked the interchange. Two lingering clues in the seven-sequence exchange are still evident though. The first is the clustering of the “random” initial sequence numbers. With ASCII codes for lowercase text running from decimal 97 to 122, the ISNs would always fall between 6356992 and 7995392. The second clue is the fact that the first socket setup and the seventh used the same ISN. The visibility of both of these clues could be lessened by some application of encryption in the cleartext string to be transmitted, thus better “randomizing” the ISNs over a range of approximately 125 as opposed to 26 [g].

An additional layer of protection for this type of TCP-stego would be to obscure the source of the communication. A means of accomplishing this would be to “bounce” the packets off of an unsuspecting server to the destination computer by spoofing the source IP address in the SYN. Using the address of the destination computer in the source IP address field of the encoded SYN packet would cause the “bounce server” to send the SYN/ACK to the intended destination workstation (not to the originating computer), with the originator’s sending sequence number, plus one (ISN+1). (Obviously, the destination computer must be predisposed to accept this packet outside standard protocol.) With that information the receiving station could then decode the embedded ASCII character by decrementing the SYN/ACK packet’s ISN and dividing the result by the agreed-upon divisor. Of course, this leaves the bounce server’s port expecting an ACK from the destination machine, which may either be closed by a RST from the destination machine or just left to time out [Rowland 8].

Audio & Video Steganograms

The form of steganography that’s getting the most press recently is that of imbedding data in digital audio, video or still images. The suspected terrorist cell communications mentioned earlier was supposedly by these means. And although debate still continues regarding whether or not these communiqués were in fact transmitted via stego, and for that matter whether steganographic communications are evident at all on the Internet, after spending some time perusing some ‘bit-bandit’ newsgroups, I suggest it is prudent accept stego on the Internet conceptually as a reality. (The real concern should be what about it poses a risk for us.)

I will use pixilated color images as an example, as they are what the majority of readily available steganography software employs as a cover object. Stego in graphics is not limited to color images; it’s just that the additional data they carry to describe a particular pixel’s color provides additional options over monochrome graphics. The devil is in the details so it is first necessary to review some color image basics.

In computer circles the best known *color model* or *color space* for describing how a pixel is to be displayed or printed is known as red-green-blue, or RGB. There are a number others. In offset printing the common color space is cyan-magenta-yellow-black or CMYK, as these are the most widely used ink colors for rendering color from dots on paper. RGB is the standard with computer imaging because of the characteristics of computer monitors and all other color cathode ray tube (CRT) devices. They all contain phosphors laid in a matte against the back of the glass front of the picture tube, and contain three different types of phosphors. These are arranged in small clusters of one each of the three (defining a single pixel) —a red-producing phosphor, a green-producing phosphor, and a blue-producing phosphor. Each can be independently “excited” in a range of intensities by being struck by a variable-strength beam of electrons. A specific color in the RGB color space is defined in terms of what “quantity” of each color is present in a given cluster of the three phosphors (based on how intensely illuminated each one is) [h].

This is where the number of bits in a color scheme or *bit depth* comes into play. The least number of bits in current color schemes is eight, meaning that there are eight bits used to describe the color of a pixel. This creates an interesting situation. Since there are eight bits available (providing 256 possible choices), and three color elements in RGB, they cannot equally share in the number of intensity levels (256/3=85.3333). I’ll digress a bit here to talk about the color spectrum.

As we all know, the visible color spectrum is a continuum from red (just above infrared) to violet (just below ultraviolet). Mathematically, there are an infinite number of points along this spectrum. Color monitors are analog devices so the “number” of colors available is a factor of the digital graphics chipsets that drive them. There are 64-bit and 32-bit chipsets, but let’s use 24 bits to keep the numbers workable in this example. 24-bit color allows for each of the three RGB elements to have eight bits to represent its “range of presence”, or intensity if you choose. So there are a possible 255 different quantities of red (and blue, and green) in a given pixel. That’s a total of 16,777,215 combinations for an individual pixel’s color. For comparison, a quality offset printing press can print around 4,000 colors, a photograph can contain somewhere around 6,000,000 colors and the human eye can discern somewhere around 10,000,000 colors. (So we’re already behind the CRT—why 64-bit color? Probably like so many other things—because we can.)

Back to the point, a change of three in decimal is the equivalent of changing the two least significant bits (LSB) from a ‘1-1’ to a ‘0-0’. The opposite would be increasing a color element’s value’s two LSBs from ‘0-0’ to ‘1-1’. So “doing the math” as they say, if I give myself free reign over the zero-order and one-order bits, each of a pixel’s individual color components can be altered by a maximum of decimal 3. So for a given pixel, there are 3x3x3 or 27 *possible* bit-values, leaving 26 possible variations from the original in a range of more than sixteen million. (Consider again that the <u>color</u> of the pixel is what your eye perceives when specific *percentages* of red, blue and green are present. So in situations where all three colors bits were changed the same way, up or down, there would no color change per-sé, it would appear more as a change in the pixel’s overall <u>color-intensity</u> or “brightness” [i].) So is there a chance your eye will notice? If the images is computer-generated, possibly. Or if you’re using magnification to compare with the original pictures a stego-picture of a polar bear in a snowstorm, or a pink kite against a sky of solid blue, maybe. But if it’s a picture that’s considered “busy”, with a lot of detail, or noise, it's not likely at all. Consider again that your eye can only discern about 65% of the sixteen-million-plus colors in the 24-bit gamut anyway. (It would even be less perceptible if the photo had had a form of "dithering" applied.) So the trick to hiding a payload from a human eye, is 1) choose the right type of image- add noise if you can; 2) use the largest reasonable color gamut available (32-bit vs. 8-bit) and, 3) don't get greedy—the fewer of the LSBs stolen (single LSB as opposed to the last four LSBs) the less obvious the hidden package will appear. Think about it in the extreme- in my example, if I used *twelve* bits per-pixel for stego, and was hiding another image, I'd easily see that there were two images there. (A good example on the Internet of this effect using various numbers of LSBs of two merging images can be found at: http://www.cl.cam.ac.uk/~fapp2/steganography/image_downgrading .)

So how broad is this channel in which I can secret my own data? Ultimately, it depends on two other factors, resolution (or more succinctly, pixels-per-inch/millimeter (PPI)) and the geography involved—exactly *how big is it?* First, assuming that the target image is one that is captured for printing, the rule of thumb is to scan it at one-and-one-half the resolution at which it will be printed. So if this picture were for a newspaper printing images at 1200 dpi, we'd scan it at 2400 dpi, which is the next highest scanner setting. Assuming the image is 4"x5" that's 2400 dots-per-linear-inch, squared, or 5,760,000 dots (or pixels) per-square-inch. Multiply that by the number of square inches in the picture or 20 (4x5). That's a total of 11,520,000 pixels. Stealing six bits per pixel (2-each for red, blue & green), that's 69,120,000 bits or about 67KB. Granted, the original image would be on the order of 17MB, and that's not something you'd normally download over the Internet, but these are not unrealistic numbers for high-resolution images on your network. Could someone hide and move without notice over 60KB of contraband data on your network? (By the way, proportionally, doubling an image's resolution quadruples its number of pixels. Thus, halving the resolution of an image reduces its file size by a factor of four.)

Putting ourselves in the shoes of our friend Warden Wendy, and faced with the potential of large blocks of potential covert data moving about, we must decide how best to protect our individual domains. Is it imperative that we discern *what* information is in a suspected stego-gram? Should we be active wardens and try to thwart suspected communications by manipulating the file, or is it enough just to play the passive role and block the communication? Would doing either be creating more grief than it would be alleviating a possible threat? Obviously, it depends primarily on your specific circumstances. If you're protecting your enterprise using a protect-detect-inhibit approach, successfully blocking unauthorized communications may be enough. If your approach includes a prosecution component, knowing the content may be critically important.

It's interesting to consider that although traditional law enforcement is keenly interested in intercepting and decoding hidden communications, it's been several other "law enforcement" groups driving the efforts to create "unbreakable" steganographic techniques.

Tool or Threat?

Used as a tool, “stego” technology currently provides the basis for *digital watermarking*, a tool for protecting copyrights in a variety of digital audio, video and software entities. Properly applied, it can also provide a means of authentication, certification validation and a standard for non-repudiation. And it is big business.

According to the Business Software Alliance (BSA), reported software revenue losses due to copyright infringement in 1999 totalled more than $12 billion worldwide and more than $59 billion over the last 5 years [j]. At about the same time, the Recording Institute Association Of America stated that the losses to the music industry were near $4.5 billion. The Motion Picture Association Of America suggested motion picture industry losses to be around $250 million. I wondered for a time what the real value to the producers would be to be able to watermark their digital wares.

Then while reading Secrets and Lies, a book by Bruce Schneier [9], I came across a discussion of watermarking and *digital fingerprinting*. Either or both might possibly be applied electronically at the time or purchase. He points out that while a copy of a film by Disney Studions might have a watermark of “Property of Disney”, the digital fingerprint might say “Purchased by Alice, 1/1/01.” The authorities would love to find that kind of evidence on pirated material. The book goes on of course to discuss that like every other digital protection, it will ultimately be compromised, but then the cycle continues, doesn’t it? So you can expect that developers of unbreakable watermarks and fingerprints for software, still and motion images and audio to be busy making their digital imprinting technologies more robust.

One can expect that for probably the next hundred years (considering the current duration of copyright), companies will continue work on better, stealthier techniques for watermarking, while the “Black Hats” continue to do the same for secreting other types of data-- both often benefitting from the others’ labor. A bit of irony is that within the BSA’s press kit you’ll read that a major concern for them regarding Internet-related software piracy is the unwanted application of the very technology secreting watermarks within legitimage files—the steganographic hiding of pirated software within email messages and newsgroup postings. [BSA 10]

Reading through some of the newsgroups of those interested in compromising systems’ security and penetrating computers by way of the Internet, you’ll see that stego is a fairly popular topic. The reason for the interest appears not to involve secreting Trojans, cracking systems and stealing software or data per-sé, but more for protecting themselves in the event they’re discovered! They can use stego against “the Feds” to hide their tools and evidence of their activities in innocuous looking files.

Steganalysis

This brings up the important topic of *steganalysis*, or the process of investigation to determine the presence of a steganographic payload. Suppose the Feds suspect that the computer of one of these guys is believed to contain stolen data and hacking tools. Suppose further that surprisingly, after their raid they find there's very little on the computer aside from the operating system and some pictures of this guy's grandmother at her 80th birthday party. Did he delete the evidence? Not necessarily.

One of the defensive uses of steganography is that of creating *plausible deniability*. Simply encrypting files on his computer would have raised suspicion. Making the encrypted files "hidden" using file atributes may have increased the suspicion. But what of seemingly random "trash" out on the disk with no sector allocation referencing it as a file at all [k]? So what is it? The perpetrator (or more likely, his lawyer) says he doesn't know. How'd it get there? Same answer.

In the preceding case, a large block of data has been discovered, but is indecipherable. One might assume that this is not one single encrypted file, but where does the cover data stop and the block of embedded data begin?

If the object of investigation is an audio or video file in which data is suspected to be hidden, the job isn't a lot easier. How do investigators go about detecting the presence of information within other information when the cover file itself would be expected to be full of noise? Well, in the case of injection and substitution stego, getting your hands on the original cover file would help immensely! Using file comparing tools or utilities such as Unix 'diff' or Microsoft 'fc' make the task fairly straightforward. (Note that postings in the bit-bandit newsgroups advocate destroying the original cover file after hiding warez and such in a stego-object…) Not having the original cover makes it a different matter entirely.

Not unlike comparing baseline network traffic patterns to current utilization to flag possible inappropriate network usage, statistical analysis of the digital content of suspected steganograms provides the best means of detection. The *statistical distribution* of bits within an image file is an example. The objective would be to determine if an image's statistical properties depart substantially enough from a "norm" to make it suspicious. But what *is* normal in a digital image?

Like character usage in a given language, there are many examples of common or standard distribution of elements. As stated earlier digital audio, video and still image files each contain a certain amount of noise, that is data which can be altered or eliminated without appreciable degradation noticable by a human observer. So each type of file in original form, when statistically analyized, yields to some degree a predictable distribution of bits sometimes called a footprint. They vary of course, depending on the file's specifics, but accounting for the content, they are each *somewhat* predictable, either in their expected bit-randomness or entropy, or expected pattern.

When used as a cover, the color palette (or map) in 8-bit images suffer changes in the color sequence. This is due to a palette's colors (256 colors which appear in the image), being sequentially numbered while the selected colors of the palette elements are not progressive (following the color shift of the standard spectrum). In order to change a pixel color by a one- or two-value, the palette must sorted progressively so that a 1-bit change in pixels color value doesn't result in the pixel changing from something like pale blue to red. A sorted color pallete in an image file is an immediate giveaway.

Non-palettized images are a much greater challenge. This is where the mathemeticians come to the front of the room, and the real statistical analysis begins. I won't pretend to understand how this works, but if you're interested, many of the mathematical details are laid out in a paper by Neils Provos and Peter Honeyman titled Detecting Steganographic Content on the Internet [11]. The paper describes their detective work following a February 2000 USA Today article about terrorists using pictures on eBay to pass information. Using Stegdetect and searching for signatures derived from two popular steganographic systems, Jsteg and JPHide, they analyzed two-million JPEG images they downloaded from eBay. They determined that there were none containing steganographic content. The search then went on to Usenet archives and in reviewing a million images there, they found some twenty-thousand they considered suspect. They attempted to extract hidden content from these using the Jsteg amd JPHide programs. As each requires a stego-key, they each image file was subjected to a dictionary attack containing a million-point-eight words and phrases. They were unable to find a single image which gave up hidden content. The Usenet methodology and findings are available on the Internet [12].

Neils Provos' Stegdetect is believed to be one of the best (publically available?) detection programs around. So are we to believe stego isn't really out there?

Regarding the eBay search, I think it likely that before posting submitted photos, eBay routinely resamples high-resolution pictures down to a resolution suitable for screen preview, facilitating faster transport and browser response. I know I would. In my testing this broke stego objects using Stools-4. (This could account for the lack of suspicious files as compared to the Usenet search.) Also corrupting were conversion of image formats from GIF to JPEG and back to GIF, and changing the color space from RGB to CMYK and back to RGB. I suspect cropping images are hazardous to many forms of stego too.

Some stego approaches are better than others for both indetectability and survivability. For example, some programs look for specific areas in a photo that have sufficient detail to permit higher data-substitution rates. Some break a photo into segments into which different payloads or multiple copies of one payload might be embedded. According to the program descriptions on his web site, in Provos' own stego program OutGuess: "for JPEG images, OutGuess preserves statistics based on frequency counts. As a result, no known statistical test is able to detect the presence of steganographic content. Before embedding data into an image, OutGuess can determine the maximum message size that can be hidden while still being able to maintain statistics based on [bit] frequency counts. . . . OutGuess tries to find a sequence of bits that minimizes the number of changes in the data that have to be made."

I find it interesting that the author of a stego program that "no known statistical test is able to detect the presence" of is believed by many to have determined through the use of statistical testing that stego is basically not in use on the Internet. Closer reading probably shows some license being taken with his findings. As I stated before, steganalysis use by both sides is what is driving the need for bigger and better steganographic methods, and the use of modern robust encryption methodologies and secure passwords assures that there will continue to be a lot for developers to do. Again, like with most technological developments which foster a counter-technology (like missle systems and missle-defense systems), the race is on.

What's a mother to do?

Stego and other covert channel mechanisms can be used to accomplish two things, leak data or provide a means to conceal it. In terms of confidentiality, integrity and availability, it seems that the threats discussed are most often viewed in terms of their effect first on confidentiality, being used as a means of theft or unauthorized transmission. Secondarily, regarding the cover files, data integrity is obviously at stake. Availiability attacks don't immediately seem obvious, but a clever mind could devise an application I'm sure.

As offensive weapons, these techniques have their places in the arsenal, but are not to date seen generally as major threats, partly due to their bandwidths limiting the practicality for most attackers. A considerable threat to the information security community though is that the lack of curent use also puts stego threats "under the radar" of many systems people, therefore enhancing the potential risk of them. Think about it; the first time ever an attack is used is when it is most potent. There are untolled examples in the history of physical conflict, one of the latest notable being the means of attack on the World Trade Center.

Obviously, in the big picture, even just looking at the use of stego to protect the guilty by obscuring evidence, we need to be aware of its presence, as it is working against us. Aware too that an innovative application of some steganographic technique for which there is no current protection could be devastating. But where does this put us besides feeling vulnerable? How can this be put in some quantifyable context so that one can take appropriate measures for their own infrastructure? When in doubt, I often turn to my highlighter-striped copy of Winn Schwartau's Time Based Security – Measuring Security and Defensive Strategies in a Networked Environment [13]. A small book perhaps, but in my opinion, one with large implications. (In just the first fourty pages, this book changed my entire perspective about the application of security measures.) Consider the following:

1) All security is time-based. The bad guys in any "field of endevor" have always known that. They don't want to get caught, so one of the first things they consider is how quickly their work is going to have to be performed, be they robbing a bank, stealing a car, copying the answers off of your test paper or stealing all the credit card numbers from the database of GiantCapitalist.com.

2) The “fortress mentality” as a defense has proven to be indefinable in any meaningful quantitative way, because risk is not static. And no matter how deep-and-wide the moat, even given the alligators, with enough time, resources and resolve, the raiders can eventually get inside the wall.

3) I now think of basic security in terms of three components, protection, detection and reaction. In Mr. Schwartau’s words the concept is this:

   > Pt > Dt + Rt
   > “The amount of time offered by the Protection device or system ‘P-sub-t’, must be greater than the amount of time it takes to detect the attack ‘D-sub-t’, plus the amount of time it takes to react to the detection, ‘R-sub-t’. That’s it.” [13]

What could the detection and reaction mechanism be for stego given that we can’t detect most channels directly? Not surprisingly, the answer is pretty much the same as always. Monitor those things you wish to protect. My position is that the key element in any security system is detection, fast detection. Yes, you need the vault door and the police to protect the bank. But without detection, the former is a delay, and the latter is unnecessary.

In dealing with my clients, my approach is as follows:
1) Determine what information and processes really need to be protected.
2) Create and enforce a thorough security policy limiting activities which put these important assets at risk.
3) Determine the best and fastest means of detecting violations and raising an alarm.
4) Devise a means of quickly reacting to an incident alert indicating an attempt on these assets.
5) Create a means of making an attack on these assets take long enough that they can be detected and thwarted.

In the case of protecting against exploits using covert channels, no single application of technology such as a firewall will eliminate them. An application proxy or gateway could be used against TCP-stego as an example, but unless that was an obvious threat, the penalty in performance would make it a poor idea. A device which analyzes binary content looking for possible steganographic content takes a lot of computing power (read: time – Provos’ describes one of his test systems as running at 87Gflops). It’ll never be done in real-time like networkbased intrusion detection. We literally cannot protect ourselves against any but a small few forms of stego.

So, continue to closely monitor the valuable assets you have, and constantly review access control logs and mechanisms. Other means of detection may include paying attention to increases in network traffic like transport of high-resolution images. Why would 24-bit 2400 dpi graphic files be going out through email unless they're being sent to a publisher?

If the possibility of data being secreted in images is a concern, consider using a graphics program like Photoshop or Hijaack to play the part of the active warden and convert images' color space or resolution. (I'm certain there are audio and video counterparts that would likewise provide a means to break many varieties stego in those formats.) Another possibility in businesses which use high-resolution images in any quantity, would be to consider implementing an Open Prepress Interface (OPI) server approach. This allows pages to be created and proofed using lower-resolution preview files which will both restrict access to the high-res files, and could substantially improve productivity at the workstation level by reduced processing overhead and network bandwidth utilization [m]. Finally, periodicaly scan your systems for the presence of stego tools. An extensive list is available at the URL listed under "Additional Resources" below.

Lastly, consider the use of the defensive possibilities of stego. Taking a lesson from the bit-bandits, how might a little security-by-obscurity be of value? Could it be beneficial at the end of the day to wrap sensitive data files in an additional protective layer by hiding them in picture files of your company picnic; the old fake-tomato-soup-can aproach? What about using stego file systems on laptops to mask critical data while in transit. In certain cases, just the increase in the file size of important data by embedding it in another file can be valuable.

In Time Based Security, Schwartau suggests *data padding* as something to consider when an attack is likely across a channel with a limited bandwidth. Here, the formula looks like:

> $F/BW=T$
>
> "If the attackers goal is theft of information, the size of the critical target files, 'F' divided by the maximum bandwidth of the communications path 'BW' determines the amount of unhampered attack time required, 'T' and thus is one measurement of risk." [13]

Obviously, you'd want to detect and react in less time than 'T' in order to prevent a loss, and risk increases as BW becomes greater. Have you ever considered reducing a legitimate access channel's bandwidth to *increase* security?

The conclusion? Steganogaphy in its electronic forms is a young technology. The first acedemic conference on stego was only held in 1996, and its development is being driven by some players with a lot at stake. It will only increase in importance in the security community. And although I can't offer specific tried and true countermeasures against stego and its cousins directly, I can state that there are applicable protective strategies already available that are fairly effective.

1) Stay informed.
2) Consider the element of time in your defensive strategy.
3) Continue to apply offensive weaponry in defensive ways.
4) Document your experiences.

5) Inform the community when you encounter a new threat.
6) Don't consider any method of adding protection to your enterprise too pedestrian.

After all, the bad guys are at this moment doing these very same things.

Additional resources

For additional information and links to sites on the subject see:
http://www.jjtc.com/Steganography/.htm

For an extensive descriptive list of steganographic software, on the same site see:
http://www.jjtc.com/Steganography/toolmatrix.htm

References

[1] U.S. Department of Defense. Trusted Computer System Evaluation "The Orange Book". Publication DoD 5200.28-STD. Washington: GPO 1985 http://www.radium.ncsc.mil/tpep/library/rainbow/5200.28-STD.html

[2] Simmons, Gustavus J. Prisoners' Problem and the Subliminal Channel (The), CRYPTO83 - Advances in Cryptology, August 22-24. 1984. pp. 51-67.

[3] Maney, Kevin. "Bin Laden's Messages Could Be Hiding In Plain Sight." USA Today December 19, 2001 http://www.usatoday.com/life/cyber/ccarch/2001/12/19/maney.htm

[4] Katzenbeisser, Stephan and Fabien Petitcolas, ed. Information Hiding – techniques for steganographic and digital watermarking, Norwood, MA: Artech House, 2000

[5] Hughes, Dave; Steganography and Watermarking, University of New South Wales, School of Computer Science and Engineering. Sydney, Australia: 2000 http://www.cse.unsw.edu.au/~cs4012/hughes.ps

[6] Summers, Rita C.; Secure Computing Threats and Safeguards, McGraw Hill, 1997

[7] Petitcolas, Fabien A., Ross J. Anderson and Markus G. Kahn, Information hiding - a Survey part of IEEE special issue on protection of multimedia content 7/99 http://www.cl.cam.ac.uk/~fapp2/publications/ieee99-infohiding.pdf

[8] Rowland, Craig H. "Covert Channels In the TCP Protocol Suite." First Monday 2.5 (1997) http://www.firstmonday.dk/issues/issue2_5/rowland/index.html

[9] Schneier, Bruce. Secrets & Lies. New York: Wiley & Sons, 2000

[10] Business Software Alliance. Vehicles for Theft – The Forms of Internet Software Piracy. Web Site http://www.bsa.org/usa/netpiracy/piracykit/VehiclesforTheft.pdf

[11] Provos, Neils and Peter Honeyman. Detecting Steganographic Content on the Internet. University Of Michigan Center for Information Technology Integration. 8/31/2001 http://www.citi.umich.edu/techreports/reports/citi-tr-01-11.pdf

[12] Provos, Neils and Peter Honeyman. Scanning USENET for Steganography. University Of Michigan 12/2001 http://www.citi.umich.edu/u/provos/stego/usenet.php

[13] Schwartau, Winn. Time Based Security / Measuring Security and Defensive Strategies In A Networked Environment. Seminole, FL: Interpact Press, 2001

Notes

[a] A common manner of referencing relationships – A (Alice) communicates to B (Bob), monitored by W (Wendy the warden). Other parings might include labels such as D (David) is signaling E (Edna) clocked by T (Tom the timekeeper), etc.

[b] In actuality, a third option exists, that being *malicious*. Action maliciously, Wendy can indiscriminately alter messages between prisoners and/or create complete messages and send them on as though they originated from either prisoner.

[c] Our "environment" may be any sized entity from a simple peer-to-peer network to an enterprise. In fact, the threat actually begins with individual processes with different security levels running on the same CPU, and builds in layers of complexity.

[d] Documents obtained by the Electronic Privacy Information Center (EPIC) released through the Freedom Of Information Act can be reviewed at: http://www.epic.org/open_gov/foia/secrets.html

[e] The book Disappearing Cryptography: Being and Nothingness on the Net by Peter Wayner AP Professional, San Diego, 1996, provides a great deal of background on propagation stego including the Pascal code for a program that generates context-free grammar that appears to be a voiceover from a baseball game.

[f] One could argue that this is not "pure" steganography, as transporting and executing malicious code goes beyond mere "storage" in a covert channel. It is also clearly outside the bounds of the prisoners' problem unless we make Wendy the victim. I must add though that during my research I've found nowhere an indication that to fit the definition there must not be a means to trigger an executable steganographic payload without formal extraction.

[g] Considering the distribution of characters in English language communications, the ISN clustering would be even more evident, as without encryption, they would follow directly the distribution of characters in the text string.

[h] Time for a disclaimer. Color generation and perception are neither linear nor objective. This text is not intending to be a treatise on color, and will present a selective number of technical details that relate to steganography and steganalysis.

[i] Of course, in the context of the "unique" colors available in a device's color gamut, each individual bit change defines a new color. Here I'm dealing mostly with "detectability", and changing a pixel's elements equally maintains the color's "hue", but affects the "saturation", a much harder change for the human eye to discern. There is at least one stego program that exploits the brightness or more correctly saturation of pixels alone as opposed to manipulating the color bits per-sé. The effect or reduced saturation would be that a picture's colors might appear "flatter".

[j] The U.S. and Canada shared about 26% of those losses.

[k] There are programs which can create a steganographic file system. To further obscure data within such a file system, a program caled Stealth, used with PGP, filters the identifying header information to make the data stream less identifyable and thus better for steganographic use.

[m] For details, see http://www.adobe.com/support/techdocs/99b6.htm