

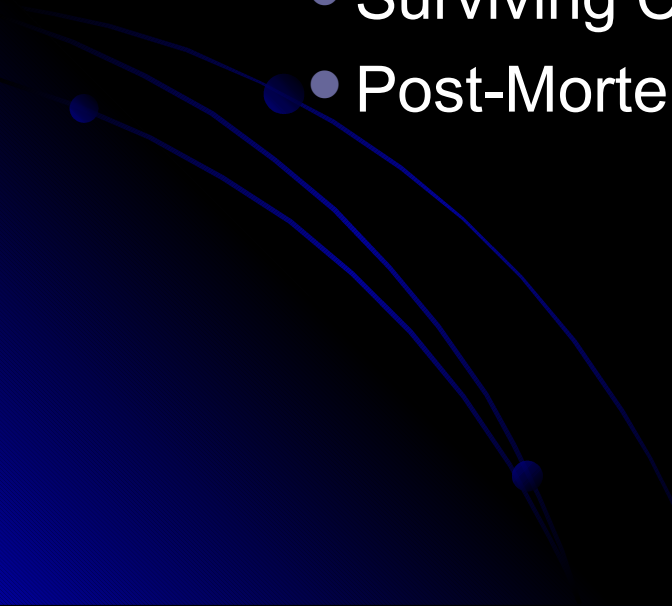
Le'go My Stego

Steganography
in the post Web 2.0 World

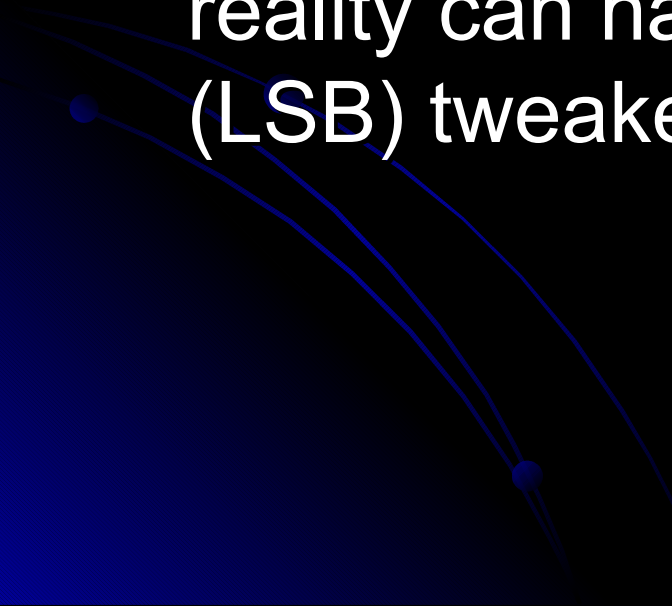
James Shewmaker © 2008
Defcon 0x10



Today's Agenda

- Today's Agenda
 - Background: Classical Stego
 - Digital Stego Techniques
 - Extending Stego concepts
 - Surviving Conversion
 - Post-Mortem Conversion Detection
- 

Classic Stego

- Old School Spy vs. Spy
 - Classified Ads
 - Microdot
 - Digital Stego - Any media file that samples reality can have its Least Significant Bit (LSB) tweaked with minor disruptions
- 

Classic Digital Stego

- Classic Digital Stego
 - Manipulating that Least Significant Bit
 - Using unused space in a host file
- Examples
 - Digital audio – fairly subtle
 - Even magnitude == zero
 - Odd magnitude == one
 - Digital image – also fairly subtle
 - Even LSB of a pixel == zero
 - Odd LSB of a pixel == one

Distributed Stego

- Many public video sites converted to flash video (FLV)
- Take your favorite viral marketing video
 - Encode to FLV before you upload
 - Store data with LSB stego using each frame/tag/box
 - (GIF/PNG/JPEG, etc.)
 - Store parity bit with each audio sample
- Classic/Simple Stego is not quite robust enough to survive video conversions
 - High redundancy might survive conversion
 - If we pick our codec well, it might survive unmolested

Phfft—who needs binary anyway?

- Whitespace in public blog comments
 - Seed arbitrary blog with keywords, then ask Google to find the blog
 - \x20 between words == zero
 - \x20\x20 between words == one
- Misspelt blog comments
 - the == zero
 - teh == one
- These techniques are compressible and subtle enough to likely be overlooked when classic stego detection tools are used

Creating a Stego Filesystem

- Previous slides could be used for data or metadata
- Pick a method to encode a structure, ie:
 - Use blog comments as metadata for a dually-linked list
 - URL to previous metadata comment
 - URL to datablock
 - URL to next metadata comment
 - Store datablock in video frame/tag/box (up to one bit per pixel)
 - Store an extra parity bit for the datablock in the audio sample

How the data survives conversion

- Small bit errors from conversion could be detected and corrected with Hamming code-like techniques to survive conversion
- RAID 10 the metadata dually-linked list
 - That is to say mirrored sets of RAID 5
- If LSB bits are lost in a single frame/tag/box-we can recover
- If the conversion taints a portion of the frame/tag/box-we can recover

Even Hamming code example

- Every power of 2 is a parity bit (4 extra bits)
- For example, store \xFF, blanks are parity
 - `__1_ 111_ 1111`
 - 1st bit checks 1, skips 1, then repeats, 5 ones is odd so we get
 - `1_1_ 111_ 1111`
 - 2nd bit checks 2 bits, skip 2...
(2, 3, 6, 7, 10, 11), 5 ones so we get
 - `111_ 111_ 1111`
 - (4, 5, 6, 7, 12), 4 ones so we get
 - `1110 111_ 1111`
 - `1110 1110 1111` –Final encoded

Fixing a bad bit

- 1110 1110 1111 –Final Encoded
- 1110 1110 1011 –Damaged
- ^ ^ - Lies!
- 2 + 8 =10 -bit 10 is bad!
- 1110 1110 1111 -Corrected!
- This will detect 2 bit errors, but correcting more than 1 error requires wrapping all of this parity with more checks

Hold on for a second

- What do we have so far?
 - We can hide data inside of other data
 - We can store 8 bits of data and use 4 bits to detect and correct
- If we do this for every sample (pixel), it is easier to detect
 - Many near duplicate colors
 - Compressibility changes
- So let's use sparse encoding inside

Not Just for Stego

- What use is sparsely encoded data?
 - Covert storage
 - Classic-Just stash you bits inside of other data
 - Covert metadata
 - Stash data about your covertly stashed data
 - Covert communications channel
 - Use the stashed data as a signal
 - For example, is a decoding algorithm or key
 - Watermarking
 - Stash a unique serial number to track the host data

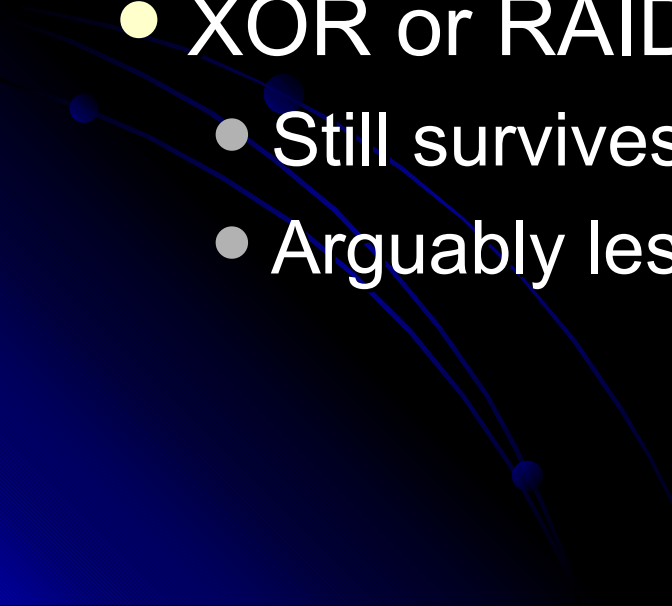
Ok, so now what?

- Making a frame survive conversion isn't everything
- What happens when video is resampled?
- This entire frame might be skipped or merged with the next frame
- But we can use another parity code across frames
- By adding this extra dimension, we can survive dropped frames

Adding redundant datablocks

- Easy to mirror the datablock for three continuous frames
 - If one frame is dropped, no problem
 - If two frames are merged, no problem
 - Just need to be able to identify a datablock
 - Might be merged with copies of itself (same five frames)
 - Might be merged with next datablock
 - Likely downsampled and threw out frames

Patterns affect compressibility

- Encoding datablocks with Hamming code not too obvious
 - Mirroring three frames is more obvious
 - So what we have so far has obvious patterns
 - XOR or RAID 5 the frames?
 - Still survives frame drops
 - Arguably less obvious
- 

Using three frames to encode xFF

- If we mirror the first two frames, we can add an XOR'd version of the byte in the third frame

1110	1110	0001
1110	1110	0001
1111	1111	0000

- For second byte we could do the same or add the XOR'd version from the previous

So what do we have now?

- An FLV video-each frame is an image
- Each frame is encoding one bit per pixel by choosing either even or odd
 - Increases near-duplicate colors if we apply blindly-becomes easy to detect
 - Chances are we will lose near duplicate colors during FLV to FLV conversion
 - Largely prevented by choosing colors well (say only encode green pixels)

So what do we have now? (2)

- Each frame contains a number of Hamming encoding bytes
- The next frame encodes the same bytes in the same way
- The third frame XORs the two previous frames with the XOR result from the previous XOR'd frame
- So we have stego byte correction and frame correction.

Automating this in reality

- These structures could hold anything
- Put the structures in arbitrary places
 - Some sites mirror
 - Some thieves plagiarize (almost as good as a mirror)
- Ask Google to find them when needed
- “Drive Maintenance” – periodically look up with Google, upload any necessary pieces (to keep redundancy from getting weak)

Alpha Implementation

- StegoFS
 - You've seen gmailfs—same idea
 - Originally written in Perl with older FLV
 - Rewritten using Python (py-fusefs, py-game, pymedia)
 - POC only, no intention of maintaining
 - Planning to release by August 2008 at <http://bluenotch.com/resources/>

Bonus Round

- We have only talked about LSB stego, what about using a relationship to encode bits?
- Can we build a pattern out of key frames? (key frames used to seek)
 - Two close keyframes = zero, two sparse = one
- FLV's metadata info frames
 - Store more stego
 - Store a hash/signature to identify datablock and/or datablock tampering
- I'm not Dan Kaminsky, but if I was I'd stash an index in somebody else's DNS ...

Ready for the Paranoia?

- In testing, it became obvious that the fault tolerance built into StegoFS revealed patterns
- That is, I could tell how the file was mangled, and could often rebuild it
- How do you know that your videos are not already watermarked?
- **They** might be able to see where you got it from, but more importantly how you got it
- Relationships are no longer safe

References / For More Info

- FLV- <http://www.adobe.com/devnet/flv/>
- Hamming code-
http://en.wikipedia.org/wiki/Hamming_code#General_algorithm

