BlackHat Europe 2010, Barcelona

Mario Vuksan, Tomislav Pericin & Brian Karney

# HIDING IN THE FAMILIAR:
## STEGANOGRAPHY AND VULNERABILITIES IN POPULAR ARCHIVES FORMATS

# Agenda

- Introduction to steganography in archives
- Introduction to file format "malformations"
  - Steganography implications
  - Vulnerability implications
- Demonstrations
  - Quick and dirty hex editing
    - Hide text and file data
    - Invent our own file format
- Introduction to NyxEngine

# Steganography

*"**Steganography** is the art and science of writing hidden messages in such a way that no one, apart from the sender and intended recipient, suspects the existence of the message, a form of security through obscurity. The word steganography is of Greek origin and means concealed writing."*

Steganography

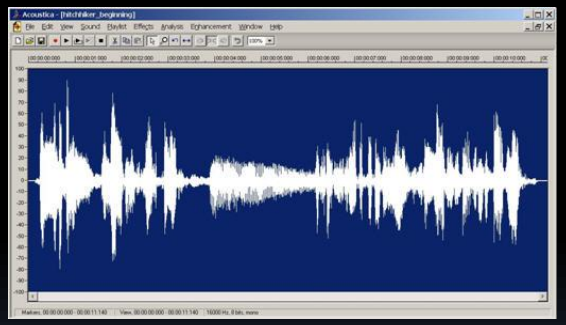
# Steganography History

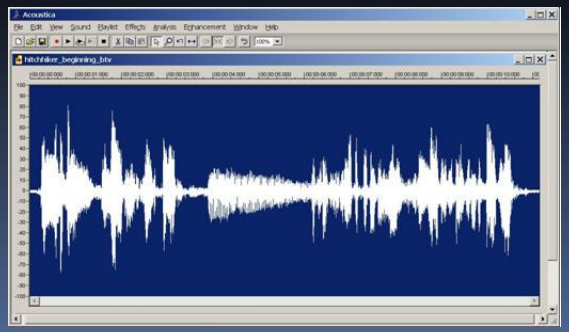
- Ancient Fascination
- Rumours & Conspiracies
  - From Pearl Harbor to Al-Qaida & eBay
- 2008 arrest
  - British Muslim, Rangzieb Ahmed used invisible ink to write down Al-Qaida telephone directory
- Difference is in the purpose
  - Malicious Uses
    - Private communication for illicit purposes, so-called Stego
  - Legitimate Uses
    - Watermarking, DRM, Movies (CAP – Coded Anti-Piracy), Medical Images Tracking

# Malicious Angle on Stego

- Types
  - Messages
  - Images
  - Media Files
- Open source projects
- 600+ different tools
- Private/commissioned tools
- Obscurity is power
- Detection
  - Stego Tool discovery
  - Brute Force







# Reality

- Why can't we find any good stories about stego in the wild?
  - It could be due to the fact it really is not that prevalent in the wild
  - It could be that analysts are not really looking so they never find it
  - That most media based approaches have many weakness and make it hard to hide large amounts of data.
  - That the best method to identify stego is to find the tools based off of Hashes

# New Paradigms for Forensics

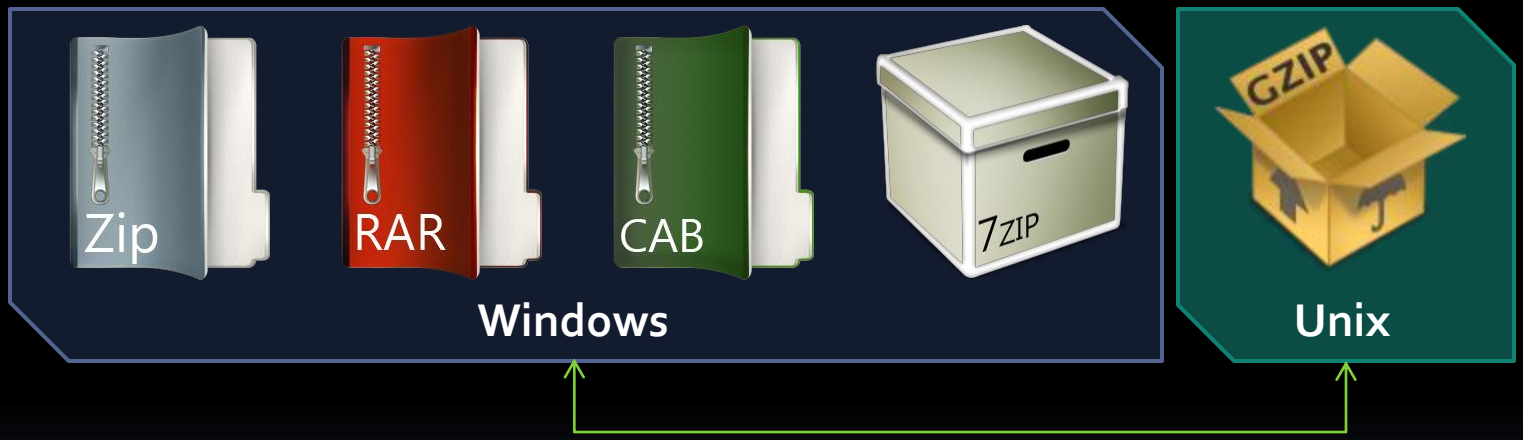
- Traditional Steganography
  - Typical stego is thought of embedding data into media files (audio files, JPG, BMP, GIF, PNG )
- New paradigm for Stego: Shift away from media
  - to archive files (zip,cab..)
  - other approaches such as SFS (Stego File System)
  - Other novel approaches

# Investigating Stego in Archives

- Why it is relevant from an investigative perspective?
  - Easier way to hide larger payloads in plain sight
  - Not easy to identify using existing methods
    - blind anomaly-based approach
    - image analysis using image filters
    - audio analyzer
    - Signature analysis (substitution)
  - Using hashes to identify tools is pointless
  - Makes you always question what is inside the archive

# Archive formats

- Most common file formats found in every Microsoft Windows, Unix and Mac OS system



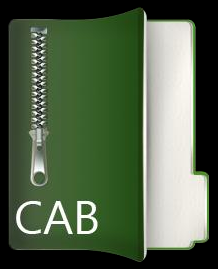

**File formats are not binded to operating system**


# ZIP file format

- Most common archive file format in use today
- The format was originally created in **1986** by Phil Katz for PKZIP
- Format is fully documented by PKWARE (32k line text file)
- The PKZIP format is now supported by many software utilities :
  - Microsoft Windows has included built-in ZIP support
  - **WinZIP** (most popular ZIP archiver program) – *www.winzip.com*
  - **PowerArchiver** - *www.powerarchiver.com*
  - **WinRAR** – *www.rarlab.com*
  - **7ZIP** - *www.7-zip.org*
- Format supports:
  - Error recovery, multi-disk spanning, encryption and SFX
  - Multiple compression algorithms in use (DEFLATE)


# RAR file format

- Very popular archive file format
- The format was as developed by Eugene Roshal
- Format is partially documented by developer (TechNote)
- The RAR format is now supported by many software utilities :
  - RAR format ships with a free decompressor library (SDK)
  - **WinRAR** – *www.rarlab.com*
  - **WinZIP** – *www.winzip.com*
  - **PowerArchiver** - *www.powerarchiver.com*
  - **7ZIP** - *www.7-zip.org*
- Format supports:
  - Error recovery, multi-disk spanning, encryption and SFX
  - Compression algorithms <u>based</u> on **LZ** and **PPMd**




# CAB file format

- Common installer file format (rarely used by users)
- CAB is the Microsoft Windows native compressed archive format
- Format is fully documented by Microsoft (20 page PDF)
- The cabinet format is now supported by many software utilities :
  - Microsoft Windows has included built-in CAB support
  - **PowerArchiver** (can compress) - *www.powerarchiver.com*
  - **WinZIP** – *www.winzip.com*
  - **WinRAR** – *www.rarlab.com*
  - **7ZIP** - *www.7-zip.org*
- Format supports:
  - Multi-disk spanning, digital signing and SFX
  - Uses LZX, DEFLATE, Quantum and MsZIP compression

# 7Zip file format

- Very common archive file format used today
- The format was created in 2000 and is developed by Igor Pavlov
- Format processor is free and open source (LGPL license)
- Format is fully documented by developer (series of text files)
- The 7Zip format is now supported by many software utilities :
  - **7ZIP** - *www.7-zip.org*
  - **WinZIP** – *www.winzip.com*
  - **PowerArchiver** - *www.powerarchiver.com*
  - **WinRAR** – *www.rarlab.com*
- Format supports:
  - Multi-disk spanning, encryption and SFX


# GZip file format

- Most common archive file format in use today (on Unix)
- Gzip was created by Jean-Loup Gailly and Mark Adler in 1992
- Format is fully documented in RFC 1952 (few pages from 1996)
- The Gzip format is now supported by many software utilities :
  - **WinZIP** (most popular ZIP archiver program) – *www.winzip.com*
  - **PowerArchiver** - *www.powerarchiver.com*
  - **WinRAR** – *www.rarlab.com*
  - **7ZIP** - *www.7-zip.org*
- Format supports:
  - Single file compression (commonly used with TAR)
  - Uses DEFLATE compression algorithm

# File format malformations

- All files present on any system are binary files


- **Malformation goals:**
  - **Steganography**
    - Hide file(s) or any other message from view
    - Steganography process must be **<u>reversible</u>**
  - **Vulnerability exploiting**
    - Don't hide anything but break archive processors
    - Fuzzing doesn't apply to this scenario

# File format malformations

- Malformation is achieved by:
  - In-depth knowledge of file format specification
  - Loose use of file format specification
  - Usage of rarely used file fields
  - "Weird" file hybrid method
  - Try-and-error method
- Steganography is achieved by:
  - All of the above
  - Injecting data


# Previous work…

- Archive malformation tests
  - Last set of tests performed in 2004 by iDefense
    - **Implications:**
      - *"The vulnerability was caused by the fact that some archive compression/decompression software (including WinZip) incorrectly handles compressed files with deliberately damaged header fields, thus, in-fact, allowing creation of the damaged archive files, that could be automatically repaired on the victims computer without notifying the user."* - ESET


# ReversingLabs|Testing

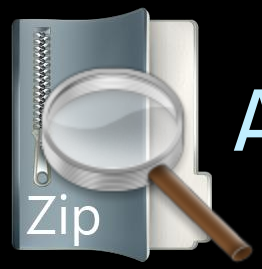
- ReversingLabs archive inspection tests:
  1. File format identification
     - Optimization: *Fastest and most accurate methods*
  2. File format validation
     - Package validation: *Archive data corruption*
     - Vulnerabilities
  3. Steganography
     - Interesting data detection
     - Data self-destruction?


# ReversingLabs|Results

- ReversingLabs archive inspection test results:
  - Steganography standpoint:
    - Multiple ways to hide file(s) and data in all formats
  - Vulnerability standpoint:
    - High probability of malware detection evasion
      - Anti-Malware scanners
        - 15 reported vulnerabilities *(more pending)*
      - Gateway scanners
      - IPS appliances

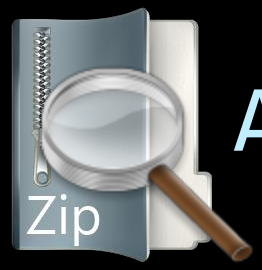
**Low impact on protected endpoints**

REVERSING LABS | Reverse Engineering & Software Protection



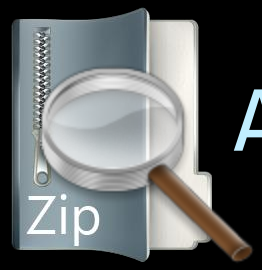

# Archive steganography|ZIP

- Steganography is achieved by:
  - Compressed file name modification (NULL byte)
  - Changes to internal ZIP structures
    - Number of packed files decrementing
    - Data camouflage by extra fields utilization
    - Moving the central directory
    - Injecting data




# Archive steganography|ZIP

- Steganography implications:
  - Data can be hidden in ZIP archives
  - Data can also be hidden in OOXML file format
  - Data self-destruction:
    - Steganography data can be <u>removed</u> by user actions




# Archive steganography|ZIP

- Steganography implementations:
  - **Zipped Steganography** by Corinna John (CPOL)
    - Can hide multiple files which are stored before central dir
    - Can encrypt the hidden files with a password
  - **ZJMask** by Vincent Chu (freeware)
    - Can hide only one file and it is pre-pended to the archive
    - Can encrypt the hidden file with a password

# Discovered vulnerabilities disclosure


# Archive vulnerabilities|ZIP

- Discovered vulnerabilities:
  - RLC_VSA_001 – Extensive header modification
    - **Vulnerability:**
      - Reversible steganography implementation
      - Central ZIP directory fields used to store information
      - Intentionally damaged local ZIP directory
      - Replaced file name first letter with zero
    - **Implication:**
      - Some scanners stopped scanning on hidden file


# Archive vulnerabilities|ZIP

- Discovered vulnerabilities:
  - RLC_VSA_002 – Password only for the first file
    - **Implication:**
      - Some scanners stopped scanning at that point assuming that the whole archive was password protected


# Archive vulnerabilities|ZIP

- Discovered vulnerabilities:
  - RLC_VSA_006 – ZIP appended to ZIP SFX
    - **Vulnerability:**
      - File is compressed and converted to ZIP SFX
      - Another ZIP file is appended and aligned to it
    - **Implication:**
      - Some scanners inspected only appended file


# Archive vulnerabilities|ZIP

- Discovered vulnerabilities:
  - RLC_VSA_011 – Utilization of extra field
    - **Vulnerability:**
      - Use of documented extra ZIP fields (2 variations)
      - Improper use but still format valid
    - **Implication:**
      - Some scanners stopped processing when they found extra fields in the central ZIP directory


# Archive vulnerabilities|ZIP

- Discovered vulnerabilities:
  - RLC_VSA_012 – Fake ZIP64 archive
    - **Vulnerability:**
      - Appended following data to central directory:
        - Zip64 End of central directory record structure
        - Zip64 End of central directory locator structure
    - **Implications:**
      - Some scanners failed to scan the archive because it was identified as ZIP64 format which wasn't supported by the vendor


# Archive vulnerabilities|ZIP

- Discovered vulnerabilities:
  - RLC_VSA_013 – File "realigned" to 0x40
    - **Vulnerability:**
      - Pre-pended 0x40 NULL bytes to ZIP archive
      - Even though archive is invalid it is extracted generically via local ZIP directory data
    - **Implications:**
      - Some scanners identified the file as broken and their generic scanners failed to detect local ZIP directory


# Archive vulnerabilities|ZIP

- Discovered vulnerabilities:
  - RLC_VSA_014 – Utilization of FileComment field
    - **Vulnerability:**
      - Use of documented ZIP comment fields
    - **Implication:**
      - Some scanners stopped processing when they found extra comment field in the central ZIP directory


# Archive vulnerabilities|ZIP

- Discovered vulnerabilities:
  - RLC_VSA_015 – Bad compression algorithm
    - **Vulnerability:**
      - Specially crafted ZipX file to which the additional file is added by any archiver program other than WinZIP
      - Utilization of new JPEG compression algorithm
    - **Implications:**
      - Some scanners didn't process the whole archive when the unsupported compression algorithm was found


# Archive vulnerabilities|RAR

- Discovered vulnerabilities:
  - RLC_VSA_003 – HEAD_FLAGS tampering
    - **Vulnerability:**
      - First RAR file block is declared as "temporary" block
    - **Implications:**
      - Some scanners failed to identify and/or decompress files whose first block was a temporary block
      - Side-effect: File which has a temporary header block is write protected. Adding files to such archive corrupts it.


# Archive vulnerabilities|RAR

- Discovered vulnerabilities:
  - RLC_VSA_005 – Password only for the first file
    - **Implication:**
      - Some scanners stopped scanning at that point assuming that the whole archive was password protected


# Archive vulnerabilities|RAR

- Discovered vulnerabilities:
  - RLC_VSA_008 – Bad extract version requirements
    - **Vulnerability:**
      - RAR decompression algorithm requirements set to version 25.0 (which doesn't exist)
    - **Implications:**
      - Some scanners failed to process the whole archive and stopped at file whose extract requirements weren't meet


# Archive vulnerabilities|CAB

- Discovered vulnerabilities:
  - RLC_VSA_004 – Incorrect decompressed size
    - **Vulnerability:**
      - Modification of the uncompressed size field
      - Effectively an archive bomb and detected as such by some scanners
    - **Implications:**
      - Extraction of such archive took large amount of time as some scanners tried to allocate the whole 4GB file first. Some skipped over the file due to its size.


# Archive vulnerabilities|GZIP

- Discovered vulnerabilities:
  - RLC_VSA_007 – Adding documented extra fields
    - **Vulnerability:**
      - Manual addition of documented and valid extra fields
    - **Implications:**
      - Some scanners failed to locate start of compressed data and skipped the file inspection

# Archive vulnerabilities|7Zip

- Discovered vulnerabilities:
  - RLC_VSA_009 – Incorrect start header CRC
    - **Vulnerability:**
      - Checksum of the first block set to 0xFFFFFFFF
    - **Implications:**
      - Some scanners failed to scan archives with invalid header checksum

# Archive vulnerabilities|7Zip

- Discovered vulnerabilities:
  - RLC_VSA_010 – Null out first header block
    - **Vulnerability:**
      - Resetting the following values in first header block:
        - StartHeaderCRC, NextHeaderOffset, NextHeaderSize and NextHeaderCRC to NULL
    - **Implications:**
      - Some scanners failed to scan archives this specific but format valid archive header


# Test|Conclusions

- ReversingLabs archive inspection test conclusions:
  1. Files could still be malformed to carry hidden payload
  2. Malformed files can be automatically fixed which making them valid on endpoint PCs
  3. Files could be "malformed" to carry stegano content
  4. Content hidden by steganography principles can have a self-destruct button


# DEMO|Steganography

- Demonstration #1:
  - Hex editing:
    - Hiding existing file(s) inside ZIP archive
    - Inserting hidden message into ZIP archive
    - Inventing file formats
  - Tool:
    - ZIPInsider

# NyxEngine

REVERSING LABS | Reverse Engineering & Software Protection

# NyxEngine|Introduction

- Introduction to the **NyxEngine**
  - **Who is Nyx?**
  - **What does it do?**
    - Does archive pre-processing
    - Inspects archive for viable hidden data
    - Recovers broken and/or hidden files
    - Acts like an exploit shield
  - **How can I use it?**
    - Nyx is a free library and it comes with its SDK
    - NyxConsole, example of SDK implementation
    - Plugin for TotalCommander and PowerArchiver

# NyxEngine|Functionality

- **NyxEngine** functional groups:
  - Archive identification
    - Supports: ZIP, RAR, CAB and GZIP
  - Packed content browsing
    - Transverse the packed content one file at the time
    - Retrieve information about packed content
    - Extract selected file slice
  - Archive validation
    - Checks if the archive is corrupted beyond recovering
  - Archive inspection
    - Search for steganography content
  - Recover salvageable corrupted content

# NyxEngine|Exploit shield

- **NyxEngine** exploit shield
  - Archive pre-processing protects from:
    - Stored file name length and content
    - Suspicious compression ratio (archive bombs)
    - Extract algorithm requirements
    - Checksum tampering
    - Multi-disk tampering
    - File entry duplication
    - … and other miscellaneous header data checks
  - Description & ReversingLabs VSA for every exploit

# NyxEngine|DEMO

- **NyxEngine** demo
  - NyxConsole tested on ReversingLabs VSA
  - NyxConsole tested on ZIP stegano solutions
  - NyxEngine corrupted file recovery

# Questions?

## (What Would You Like to Know)