



General

FA2025 • 2025-12-4

Forensics

Bryce Kurfman & Suchit Bapatla

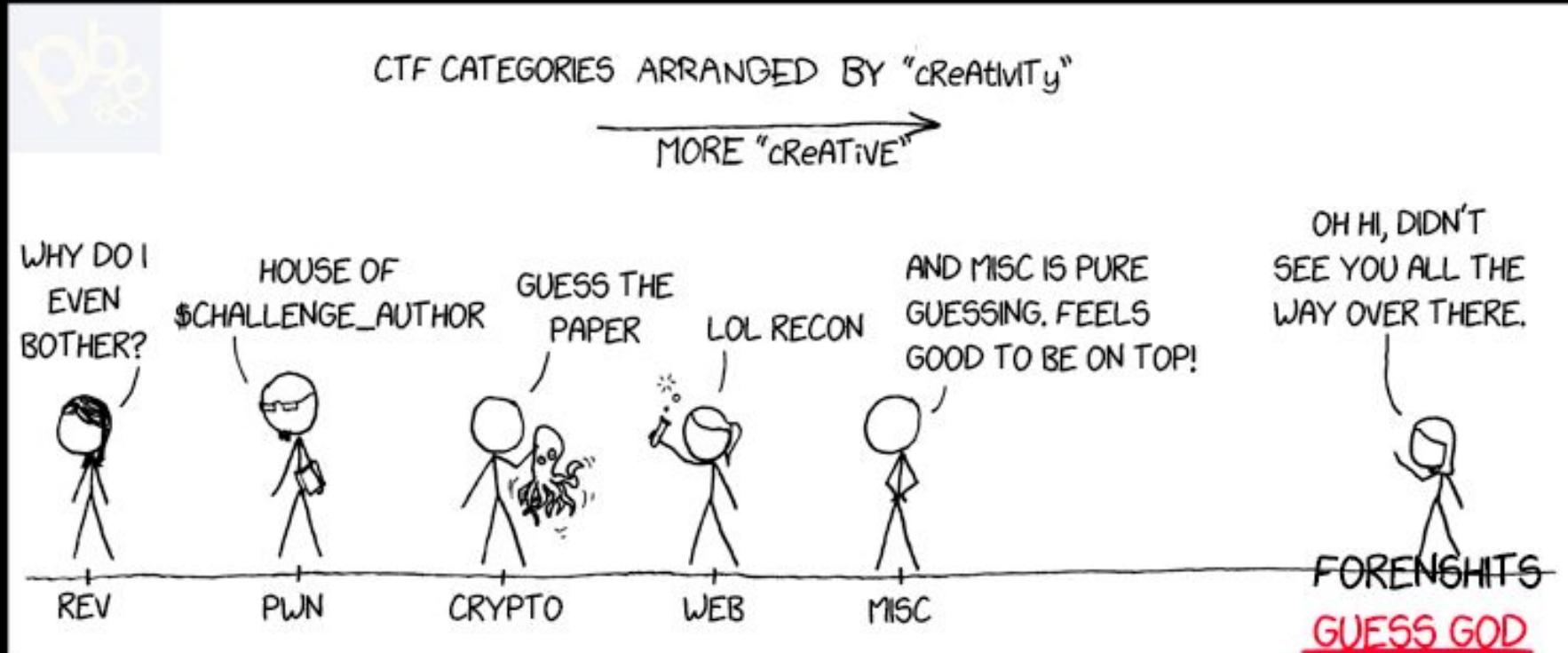
Announcements

- Minecraft/Gaming Social
 - Sunday at 5:00 pm
 - Somewhere in Siebel or DCL
- Good luck with finals!



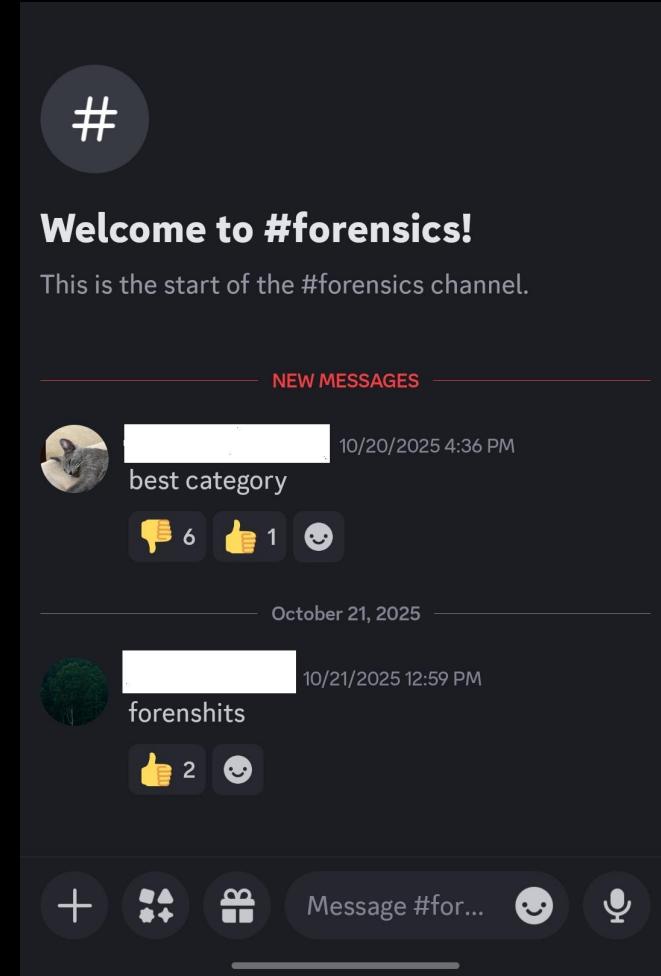
ctf.sigpwny.com

sigpwny{on_the_loose}



What is Digital Forensics?

- File Forensics
 - File format
 - Steganography
- OS Forensics
 - aka Incident Response
- Memory Forensics
 - Dump file of RAM of process or entire OS
- Sleuth Kit



File Forensics

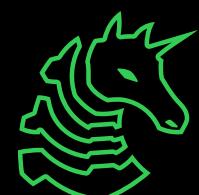


file

File extension is just a *hint* to users and OS about the file's *intended* purpose, but the content can be arbitrary.

`file` command uses "magic bytes" or a file signature to determine the format of a file (e.g. PNG files always start with `89 50 4E 47`)

```
Q: $ file unknown.txt
unknown.txt: JPEG image data, JFIF standard 1.01, aspect ratio, density 1x1,
segment length 16, progressive, precision 8, 400x400, components 3
```



binwalk

Finds files formats appended in a file (like `file` but recursive). For example, some binaries include their assets!

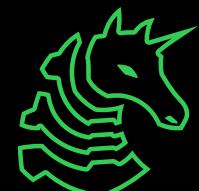
```
$ binwalk cmd.exe
```

DECIMAL	HEXADECIMAL	DESCRIPTION
0	0x0	Microsoft executable, portable (PE)
235448	0x397B8	Copyright string: "Copyright (c) Microsoft Corpora
253789	0x3DF5D	mcrypt 2.5 encrypted data, algorithm: "o_exit", ke
283472	0x45350	XML document, version: "1.0"
283534	0x4538E	Copyright string: "Copyright (c) Microsoft Corpora
294568	0x47EA8	PNG image, 256 x 256, 8-bit/color RGBA, non-interl
294609	0x47ED1	Zlib compressed data, default compression

Usage:

`binwalk <FILE>` - View a list of contained file formats

`binwalk -e <FILE>` - Extract each file format from the file



binwalk

Sometimes it's normal to find other file format in one file, as one file format may use another to encode information.

Example: Images in PDF

```
cbcicada@DESKTOP-5T74EEM:/mnt/c/Users/CBCicada/Downloads$ binwalk MP3_FA24_CP2.pdf
```

DECIMAL	HEXADECIMAL	DESCRIPTION
0	0x0	PDF document, version: "1.5"
78	0x4E	Zlib compressed data, best compression
1319	0x527	Zlib compressed data, best compression
3237	0xCA5	Zlib compressed data, best compression
4368	0x1110	Zlib compressed data, best compression
7435	0x1D0B	Zlib compressed data, best compression
10269	0x281D	Zlib compressed data, best compression
11621	0x2D65	Zlib compressed data, best compression
14478	0x388E	Zlib compressed data, best compression
17128	0x42E8	Zlib compressed data, best compression
20555	0x504B	Zlib compressed data, best compression
23832	0x5D18	Zlib compressed data, best compression
25324	0x62EC	Zlib compressed data, best compression
...



strings

Lists plaintext strings that exist in a file, useful for binary files

Usage:

strings <FILE> - print all strings of length 4 or greater

strings -n 16 <FILE> - print all strings of length 16 or greater

```
$ strings -n 16 cmd.exe
!This program cannot be run in DOS mode.
SetThreadUILanguage
Unknown exception
bad array new length
api-ms-win-core-winrt-l1-1-0.dll
ext-ms-win-branding-winbrand-l1-1-0.dll
ext-ms-win-cmd-util-l1-1-0.dll
ext-ms-win-appmodel-shellexecute-l1-1-0.dll
onecore\internal\sdk\inc\wil\opensource\wil\resource.h
WilFailureNotifyWatchers
RtlRegisterFeatureConfigurationChangeNotification
RtlUnregisterFeatureConfigurationChangeNotification
RtlNotifyFeatureUsage
NtQueryWnfStateData
NtUpdateWnfStateData
onecore\internal\sdk\inc\wil/Staging.h
CMD Internal Error %s
Null environment
APerformUnaryOperation: '%c'
APerformArithmeticOperation: '%c'
IsDebuggerPresent
SetConsoleInputExeNameW
RaiseFailFastException
RtlNtStatusToDosErrorNoTeb
RtlDllShutdownInProgress
RtlDisownModuleHeapAllocation
NtQueryInformationProcess
Copyright (c) Microsoft Corporation. All rights reserved.
onecore\base\cmd\StartShellExecServiceProvider.h
onecore\base\cmd\maxpathawarestring.cpp
```

xxd

Prints a hexdump of a file

Good to look for recognizable
hex patterns or perform
advanced hex manipulation

Usage:

`xxd <FILE>`

For hex editing: Ghex, or Hexedit

```
00025a40: d508 de91 1600 65b9 c62a 9b8b ac88 d919 .....e.*.....
00025a50: 4f3b 881d 7db3 5f44 5df8 5b50 9dca 468c 0;...}._D].[P..F.
00025a60: 8a79 24f6 ac65 f0f4 f681 8301 28cf f10e .y$..e.....(...
00025a70: 723b d2df 7339 ad74 3c54 df6a 2aa3 134a r;..s9.t<T.j*..J
00025a80: a8bf 2e00 207f 2a85 6eae 4b98 9f73 2677 .... .*..n.K..s&w
00025a90: 6587 24d7 b31d 325b 8244 8385 e76e 318a e.$...2[.D...n1.
00025aa0: 8e5b 2893 f773 c48c 0f4e 0722 856e c16e .[(.s...N.".n.n
00025ab0: e78e 4aeb 2b0f 3214 639e 8475 f635 1cc9 ..J.+.2.c..u.5..
00025ac0: 6aac d1dd 58c6 08ec 3230 6bd5 e7d0 ac2e j....X...20k.....
00025ad0: 6310 4702 c6cb cef1 eddb 3597 2784 6da4 c.G.....5.'..m.
00025ae0: 712f 9ecc 641b 8871 8c13 44af d06a 299e q/..d..q..D..j).
00025af0: 4e6d 74f6 20c9 091e 841e 950b 5858 1001 Nmt. ....XX..
00025b00: 99a3 1eb8 cd7a 4bf8 30bc a434 db42 9c60 ....zK.0..4.B.`.
00025b10: 6315 5ae3 c39f 67b9 fb3c 7961 b4b6 ec71 c.Z...g..<ya...q
00025b20: c569 cc4a a4cf 3e9f 488b cc1f 67b9 322e .i.J..>.H...g.2.
00025b30: 38ca e2a9 1d26 52a5 4b2b 919c 66bb 49ec 8....&R.K+..f.I.
00025b40: ae61 6cf9 7e62 0efd eb39 a326 4236 141d .al.~b...9.&B6..
00025b50: f230 2ab9 d8a7 4dbd 9181 1e95 a8c4 108e .0*...M.....
00025b60: fd08 6aaf 3457 a1bc cdae ac9c 6715 d288 ..j.4W.....g...
00025b70: e577 f31b 2a07 0067 a8f5 a97c eda8 487c .w.*..g...|..H|
00025b80: 03c7 d6a1 c8c9 d267 2524 f76f f248 0918 .....g%$.o.H..
00025b90: e7e5 aaf9 ff00 63f4 aec6 4f99 3681 9350 .....c...0.6..P
00025ba0: 7952 ff00 cf31 f90a a524 4b8b 3fff d9 yR...1...$K.?..
```



grep

Search for text matches within a file or recursively in files!

```
grep "text you want to find" <FILE>
```

```
grep -R "text you want to find" <DIRECTORY>
```

Combine with other utilities!

```
cat <FILE> | grep "text"
```

```
strings <FILE> | grep "text"
```

```
cbcicada@DESKTOP-5T74EEM:/mnt/c/Users/CBCicada/Downloads$ strings Goose.dmp | grep uiuctf
uiuctf{W1LD_G00S3_CH4S3_8234819284901_UNGUESSABLE_1337}
uiuctf{W1LD_G00S3_CH4S3_8234819284901_UNGUESSABLE_1337}
uiuctf{W1LD_G00S3_CH4S3_8234819284901_UNGUESSABLE_1337}
uiuctf{NO_THIS_FLAG_AIN'T_IT321}YOU\?_+GITGOOSED~i>-u
```



Steganography

It sucks



Steganography

- Hide data in other data
- Inherently guessy during CTFs
 - Try lots of ideas
 - Waste lots of time
 - Use statistical approaches if applicable

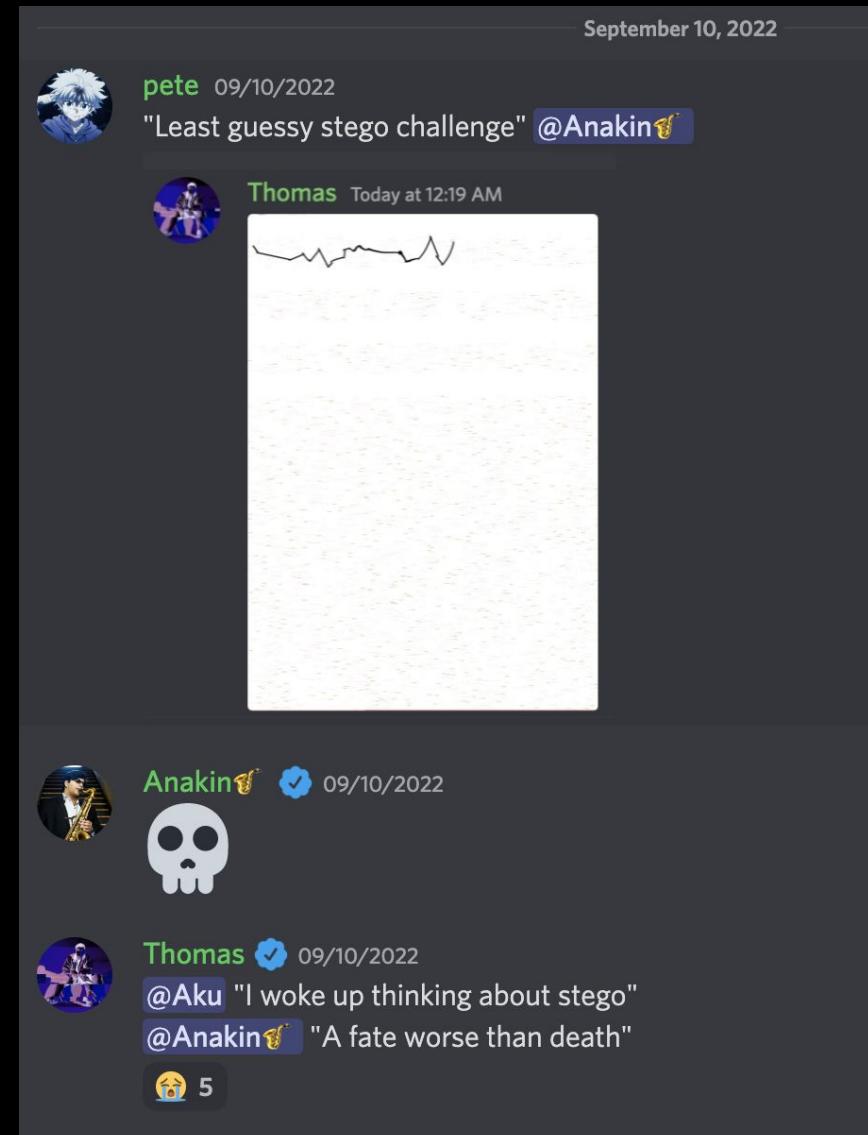


Image Stego - LSB

- LSB (least significant bit) encoding
- Not really useful in the real world, but CTFs love it
- Take the least significant bit (last bit) of each color byte and concatenate all of them to form a message
- Image is mostly visibly unchanged

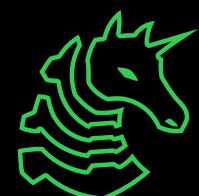
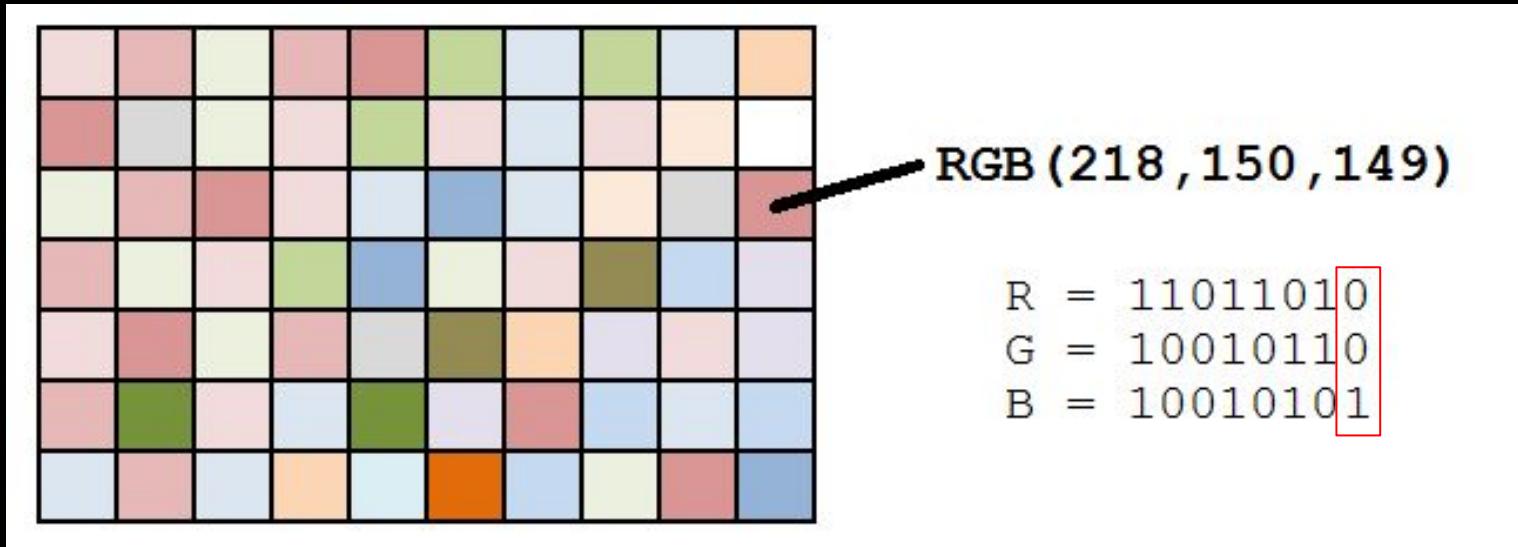
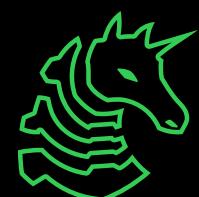


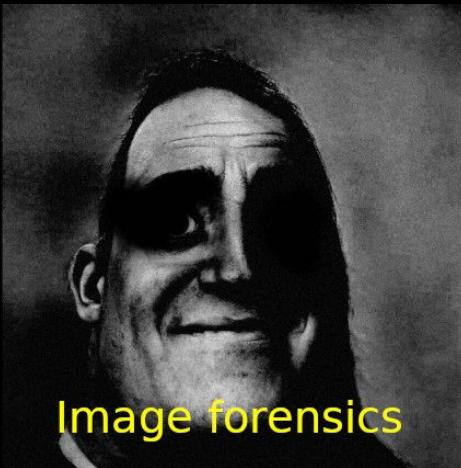
Image Stego - LSB



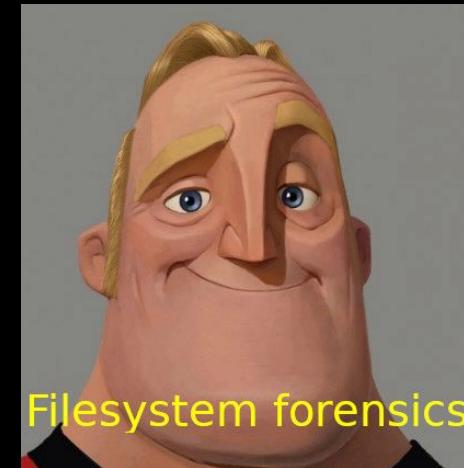
Message = (R & 1) || (G & 1) || (B & 1) = 001...



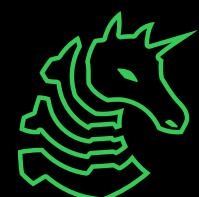
Can you tell the difference?



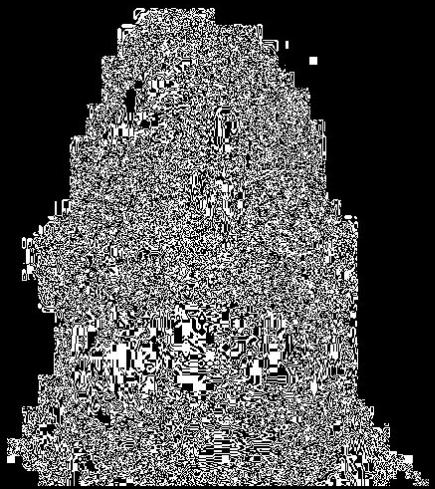
Original



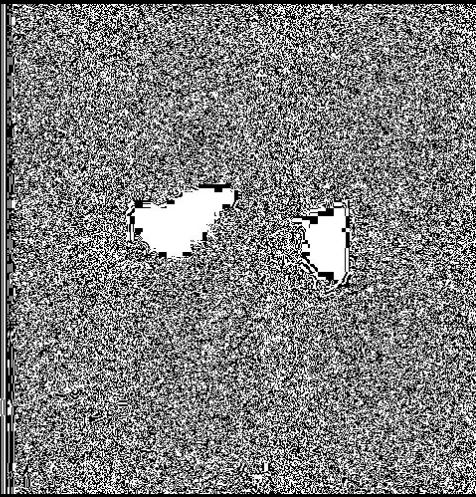
LSB Encoded



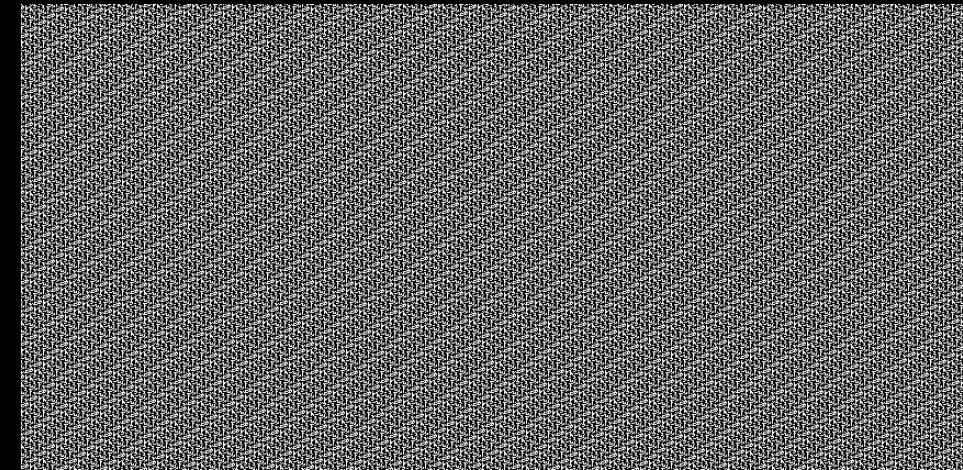
Can you tell the difference?



Original (Red,
Bit 0)



LSB Encoded
(Red, Bit 0)

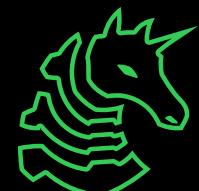


Stegoanalysis tool: <https://stegonline.georgeom.net/image>



Image Stego - Malicious Purpose

- Imagine in a red team scenario, I already have control over a machine, but I need to download and execute a binary file.
- The network may be heavily monitored with Deep Packet Inspection, and the file may be inspected by Antivirus before executed
- Plain shellcode will get detected immediately
- Instead, downloading an image that has the secret binary encoded with LSB is way less suspicious
- [POC](#)



Other Stego

- Audio stego
 - 90% of the time it is a spectrogram
 - The other 10% is either
 - SSTV
 - Some frequency modulation
 - Some other guessy home-brewed bullshit
- Most audio stego can be solved with [tools](#)
- You can kidnap an ECE major to solve these stegos



Starting Point

- What kind of file is it?
 - Use a command like `file` or `binwalk` to identify what a file is using magic bytes
 - If it's an image/video/audio, it likely contains hidden information in the form of steganography
 - If it's a document, there might be hidden information in the file format (did you know that a .docx file is just a ZIP archive?)
 - Also if it's doc, docm, it might contain macros
 - If it's plain text ASCII, what does it contain? Are there any patterns?
 - If it's a binary file, what readable strings are there? Are there patterns in the hexadecimal representation?
- What metadata does it have?
 - Images: location, camera model, encoding
 - Documents: username of creator, directory where saved
 - Everything else



OS Forensics

Log analysis, filesystem/disk



Logs

- Logs provide valuable information about what happened on the system
- You can construct a list of events that occur to determine entry point of an attack and what an attacker did

Linux:

- /var/log/http/access.log
- /var/log/syslog
- other various service logs
- ~/.bash_history or ~/.viminfo if they're really dumb

Windows:

- Event Viewer (Including PS script block logging)
- PSReadline Module
 - (Get-PSReadlineOption).HistorySavePath

See [Windows Forensics](#) and [Linux Forensics](#) slides from Purple Team for more



Starting Point

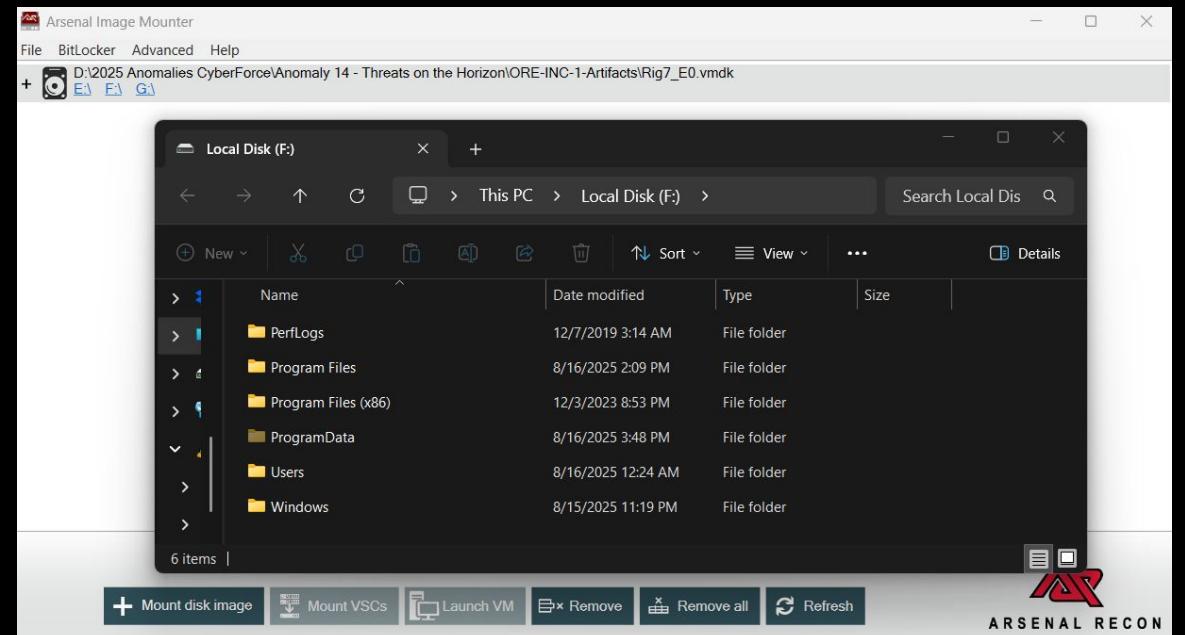
What we are given:

- An archive/zip of the full/partial filesystem
 - We can inspect logs, user files, command history, etc.
- A disk image file (.dd, .iso, .vhd, .vmdk)
 - Follows a filesystem format (e.g. NTFS, ext4)
 - Deleted files can be present and ignored by the filesystem format
 - We can mount it to a live system to examine the contents
- Live access to a system
 - Extremely difficult to perform forensics on a live system while maintaining integrity of evidence since it is volatile
 - Obtain a disk image after attempting to preserve live system evidence



Mounting File Systems

- Mounting
 - Placing the disk image in a usable and accessible state
- Autopsy
- On Windows
 - [Arsenal Image Mounter](#) is easy to set up and use (free version too)
- On Linux
 - Use a [loop device and mount](#) to mount disk images



Autopsy

- A GUI-based digital forensics platform built on top of The Sleuth Kit (TSK)
- Free, open-source, and runs on Windows, Linux, and macOS
- Key Capabilities
 - Disk & File System Analysis (NTFS, FAT, EXT, APFS, etc.)
 - Timeline Generation of system and user activity
 - Keyword Search and file content analysis
 - Registry, Browser, and Email Artifact Extraction
 - Hash-Based Filtering using NSRL or custom hashsets
 - Image Support: RAW, E01, VHD/VMDK, AFF, and more



008 - Autopsy 4.15.0

Case View Tools Window Help

Add Data Source Images/Videos Communications Geolocation Timeline File Discovery Generate Report Close Case

Data Sources

- Mantooth.E01
- Views
- File Types
- Deleted Files
- MB File Size
- Results
- Extracted Content
- EXIF Metadata (8)
- Encryption Detected (2)
- Extension Mismatch Detected (9)
- Installed Programs (40)
- Operating System Information (2)
- Operating System User Account (8)
- Recent Documents (109)
- Recycle Bin (9)
- Remote Drive (1)
- Shell Bags (110)
- USB Device Attached (30)
- User Content Suspected (8)
- Web Cookies (50)
- Web History (374)
- Web Search (130)
- Keyword Hits
- Single Literal Keyword Search (0)
- Single Regular Expression Search (0)
- Email Addresses (448)
- Hashset Hits
- E-Mail Messages
- Default ([Default])
- Interesting Items
- Accounts
- Email
- Tags
- Reports

Listing

Data Sources

Table Thumbnail

Name	Type	Size (Bytes)	Sector Size (Bytes)	Timezone	Device ID
Mantooth.E01	Image	128450048	512	Asia/Calcutta	7a1c6fb8-4b18-4921-9ac2-cca49ebc88a1

File Metadata

Hex Text Application Message File Metadata Context Results Annotations Other Occurrences

Page: 1 of 7840 Page Go to Page: Jump to Offset 0 Launch in HxD

```

0x00000000: 33 C0 8E D0 BC 00 7C FB 50 07 50 1F FC BE 1B 7C 3....|.P.P....|
0x00000010: BF 1B 06 50 57 B9 E5 01 F3 A4 CB BE BE 07 B1 04 ...PW.....
0x00000020: 38 2C 7C 09 75 15 83 C6 10 E2 F5 CD 18 8B 14 88 8,|..u.....
0x00000030: EE 83 C6 10 49 74 16 38 2C 74 F6 BE 10 07 4E AC ....It.8,t....N.
0x00000040: 3C 00 74 FA BB 07 00 B4 0E CD 10 EB F2 89 46 25 <.t.....F%
0x00000050: 96 8A 46 04 B4 06 3C 0E 74 11 B4 0B 3C 0C 74 05 ..F...<t...<t.
0x00000060: 3A C4 75 2B 40 C6 46 25 06 75 24 BB AA 55 50 B4 :.u+@.F%.u$.UP.
0x00000070: 41 CD 13 58 72 16 81 FB 55 AA 75 10 F6 C1 01 74 A..Xr...U.u...t
0x00000080: 0B 8A E0 88 56 24 C7 06 A1 06 EB 1E 88 66 04 BF ...V$.....f.
0x00000090: 0A 00 88 01 02 8B DC 33 C9 83 FF 05 7F 03 8B 4E .....3.....N
0x000000a0: 25 03 4E 02 CD 13 72 29 BE 46 07 81 3E FE 7D 55 %,N..r).F..>.)U
0x000000b0: AA 74 5A 83 EF 05 7F DA 85 F6 75 83 BE 27 07 EB .tZ.....u...
0x000000c0: 8A 98 91 52 99 03 46 08 13 56 0A E8 12 00 5A EB ..R..F..V..Z.
0x000000d0: D5 4F 74 E4 33 CO CD 13 EB B8 00 00 00 00 00 .Ot.3......
0x000000e0: 56 33 F6 56 56 52 00 06 53 51 BE 10 00 56 8B F4 V3.VVRP.SQ..V..
0x000000f0: 50 52 B8 00 42 8A 56 24 CD 13 5A 58 8D 64 10 72 PR..B..V$..ZX.d.r
0x00000100: 0A 40 75 01 42 80 C7 02 E2 F7 F8 5E C3 EB 74 49 .@.u.B.....^.tI
0x00000110: 6E 76 61 6C 69 64 20 70 61 72 74 69 74 69 6F 6E invalid partition
0x00000120: 20 74 61 62 6C 65 00 45 72 72 6F 72 20 6C 6F 61 table.Error loa

```

Credit: [Geeks for Geeks](#)



Memory Forensics



Memory Forensics

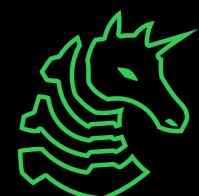
Instead of being given a traditional filesystem, you are given the contents of memory or RAM in the form of a dump file

- Dump files are usually created when a program crashes or your OS crashes for debugging purposes (or by acquisition tools)
- They can contain sensitive information that was located in memory at the time of crashing or acquisition
 - Passwords in your password manager
 - Clipboard content at time of crash
 - Network connections (possibly to C2 servers)
 - \$MFT
 - Loaded DLLs
 - lsass.exe



Volatility

- Analyze memory dump files
- There are two versions of Volatility:
 - Volatility 2.6
 - Is older, but works great when it works
 - Is limited to older OS versions (e.g. doesn't have latest Windows 10 symbols)
 - Intended to be used standalone
 - Volatility 3
 - Complete framework rewrite, under development, and missing some features
 - Is way faster and has better OS version profiling and support
 - Intended to be used as a library with standalone support
 - Annoyingly enough, also has versions such as 2.4.1



Setting up Volatility

```
git clone https://github.com/volatilityfoundation/volatility3.git
cd volatility3
pip install -r requirements.txt
python3 vol.py -f <FILE> <PLUGIN_NAME> (<PLUGIN_OPTION>)
```

\$ python3 vol.py -f MEMORY.DMP windows.pslist Volatility 3 Framework 2.4.1 Progress: 100.00													
PID	PPID	ImageFileName	Offset(V)	Threads	Handles	SessionId	Wow64	CreateTime	ExitTime	File output			
4	0	System	0xb50a3627b040	137	-	N/A	False	2022-11-17 23:06:49.000000	N/A	Disabled	88	4	Disabled
260	4	smss.exe	0xb50a39093040	2	-	N/A	False	2022-11-17 23:06:49.000000	N/A	Disabled			
388	376	csrss.exe	0xb50a36fb8140	12	-	0	False	2022-11-17 23:06:54.000000	N/A	Disabled			
460	376	wininit.exe	0xb50a39790080	1	-	0	False	2022-11-17 23:06:54.000000	N/A	Disabled			
468	452	csrss.exe	0xb50a397f7140	14	-	1	False	2022-11-17 23:06:54.000000	N/A	Disabled			
556	452	winlogon.exe	0xb50a39e4b080	6	-	1	False	2022-11-17 23:06:55.000000	N/A	Disabled			
596	460	services.exe	0xb50a39e4a140	8	-	0	False	2022-11-17 23:06:55.000000	N/A	Disabled			
612	460	lsass.exe	0xb50a39e5d0c0	11	-	0	False	2022-11-17 23:06:55.000000	N/A	Disabled			
708	596	svchost.exe	0xb50a39ecb280	1	-	0	False	2022-11-17 23:06:56.000000	N/A	Disabled			
732	556	fontdrvhost.ex	0xb50a39f07180	5	-	1	False	2022-11-17 23:06:56.000000	N/A	Disabled			
740	460	fontdrvhost.ex	0xb50a39f09180	5	-	0	False	2022-11-17 23:06:56.000000	N/A	Disabled			
748	596	svchost.exe	0xb50a39f0b280	24	-	0	False	2022-11-17 23:06:56.000000	N/A	Disabled			
852	596	svchost.exe	0xb50a39fc4300	15	-	0	False	2022-11-17 23:06:56.000000	N/A	Disabled			



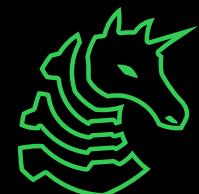
Symbol Tables

- Symbols
 - Volatility must reconstruct kernel data structures from a raw memory image
 - To do this, it needs two critical pieces of information for each kernel object:
 - Offset: Where the structure is located in memory
 - Template / Layout: The structure definition (fields, sizes, types)
- Symbol Table
 - A collection of named kernel objects, each defined by:
 - Symbol name → offset + structure layout
 - In Volatility 3, these come in ISF (Intermediate Symbol File) format (JSON).



Symbol Tables Cont.

- How Volatility Uses Symbols
 - Map virtual → physical memory using Volatility's layered model
 - Instantiate kernel objects (tasks, vm areas, lists, pointers, etc.)
 - Plugins can now:
 - Walk kernel structures
 - Follow pointers
 - Read and cast arbitrary memory
 - Enumerate processes, sockets, modules, etc.



Windows vs. Linux Symbols

- Windows
 - Easy: Microsoft publishes Program Database files (PDBs) with full kernel data.
 - Volatility automatically downloads and parses them.
- Linux
 - Hard: Too many distros, versions, architectures, and configs.
 - Kernel images are stripped by default → no DWARF data.
 - Can use remotely hosted symbols tables as to not download/create a profile for each new kernel/distribution you want to analyze
 - If it is a more niche kernel, you will need to create a new symbol table

```
python3 vol.py --remote-isf-url  
'https://github.com/Abyss-W4tcher/volatility3-symbols/raw/master/banners/banners.json' -f <memory_dump> <plugin>
```



Common Volatility Plugins

- windows.pstree: Show parent/child
- relationships
- windows.psxview: Cross-view process detection (rootkits)
- windows.netscan: Scan for sockets, TCP/UDP connections
- windows.registry.printkey: View registry keys/values

- Good [Windows Starter \(Docs\)](#)
- [Linux Documentation](#)



Or... just use basic commands

A lot of memory is simply stored as strings!

```
strings <FILE> | grep "text"
```

```
strings --encoding=l <FILE> | grep "text"
```

If you are looking for a specific file format, you can also try grep the file header / signature.



Resources

Windows filesystem artifacts:

<https://www.sans.org/posters/windows-forensic-analysis/>

(Mirror) <https://tinyurl.com/sanswindowsposter>



Next Meetings

2025-12-7 - Minecraft Social

- Woo hoo!

2025-12-19 - Winter Break

- Enjoy your break!



ctf.sigpwny.com

sigpwny{on_the_loose}

**Meeting content can be found at
[sigpwny.com/meetings.](http://sigpwny.com/meetings)**

