FA2022 Week 12

Forensics

Minh

Slide Styling Guidelines

- Remove this slide once you have read the guidelines
- Do not put "SIGPwny" or "Meeting" or "Seminar" (or synonyms) in the title
  - Unless it is for info meetings or the Recursive Meeting :)
- Use dashes ("-") for bullet points
- Use straight quotes (""), not smart quotes (“”)
- Avoid moving text boxes for titles and headings
  - Unless they are all consistently moved!
- Stick to SIGPwny theme colors in the color picker
- Do not make text too small (font size 20 is the limit)
- Reference Brand Guidelines here:
  - https://docs.google.com/document/d/1SioiiGVKIwm0sn56YOr_ESFkqx1HeAv7JfERbeI_AoM/edit

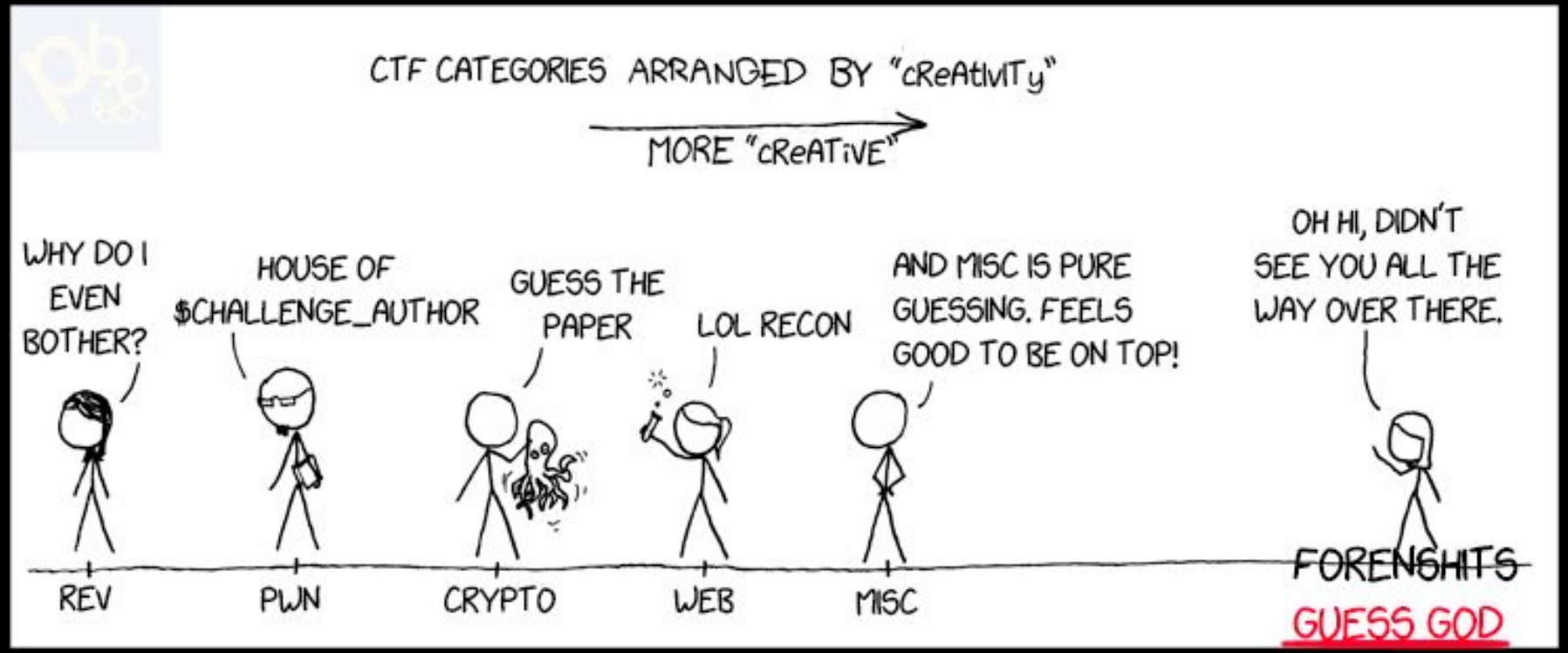
Announcements

- Sunshine CTF 2022
  - Starts this Saturday at 9AM
  - Completely virtual because of fall break
  - We are trying to get top 50 in the world on CTFtime!

- Fall break
  - No meeting this Sunday or next Thursday and Sunday

ctf.sigpwny.com

sigpwny{guess_god}



What is Digital Forensics?

- File Forensics
  - Requires understanding of a file format
  - Data can be hidden within the format (steganography)
- Filesystem Forensics
  - We conduct forensics to determine what happened on a system in a holistic manner, usually after some sort of incident
- Memory Forensics
  - Different from traditional filesystem forensics: we are given the contents of RAM in the form of a dump file
  - Given a memory dump, what information can we extract from it?

File Forensics

file

Is a file really what is says it is? Can a file be opened in a different way? Don’t always trust the extension!

`file` 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`)

```
$ 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
```

Usage:

`file <FILE>`

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

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>
```

```
00025a40: d508 de91 1600 65b9 c62a 9b8b ac88 d919  ......e..*......
00025a50: 4f3b 881d 7db3 5f44 5df8 5b50 9dca 468c  O;..}._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%$.o.H..
00025b90: e7e5 aaf9 ff00 63f4 aec6 4f99 3681 9350  ......c...O.6..P
00025ba0: 7952 ff00 cf31 f90a a524 4b8b 3fff d9    yR...1...$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"
```

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

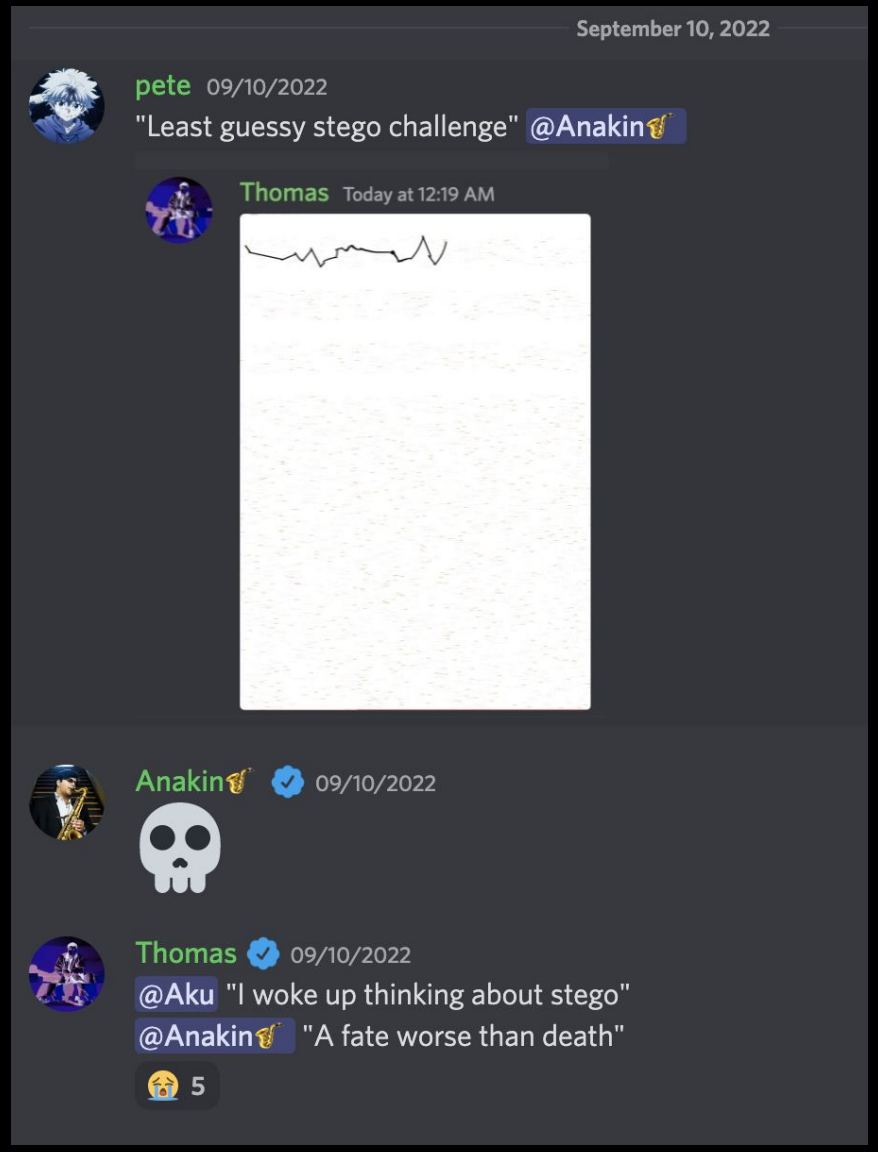
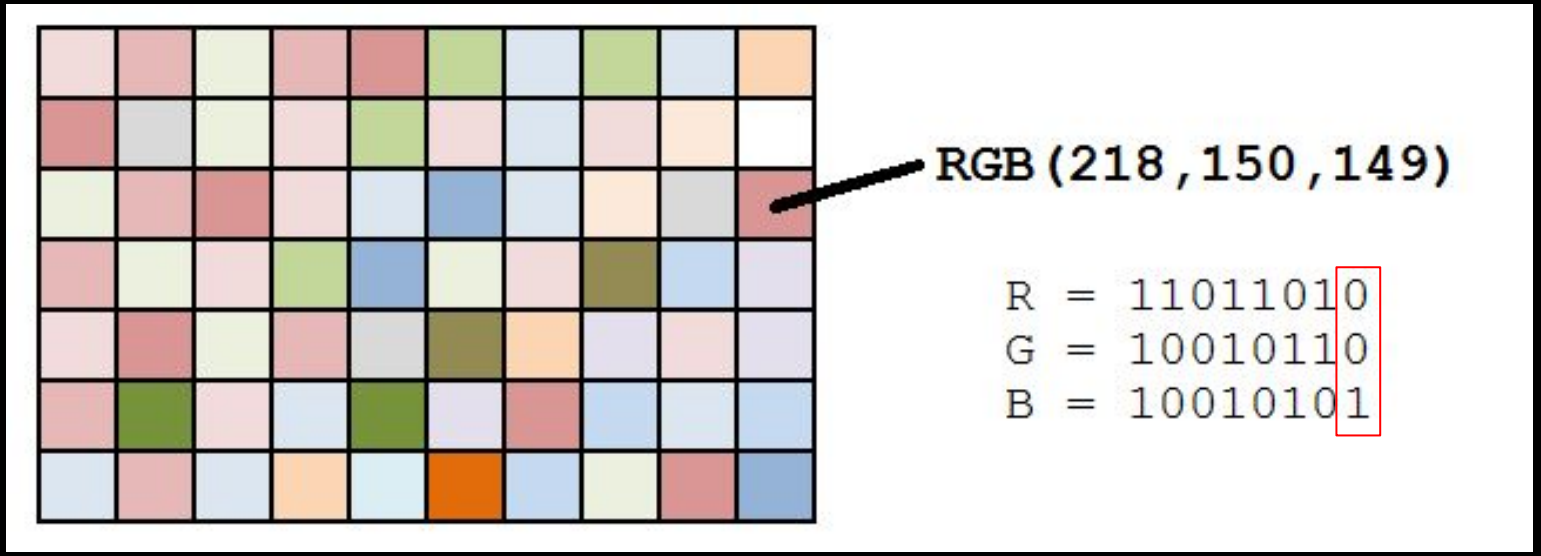


Image Stego - LSB

- 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...

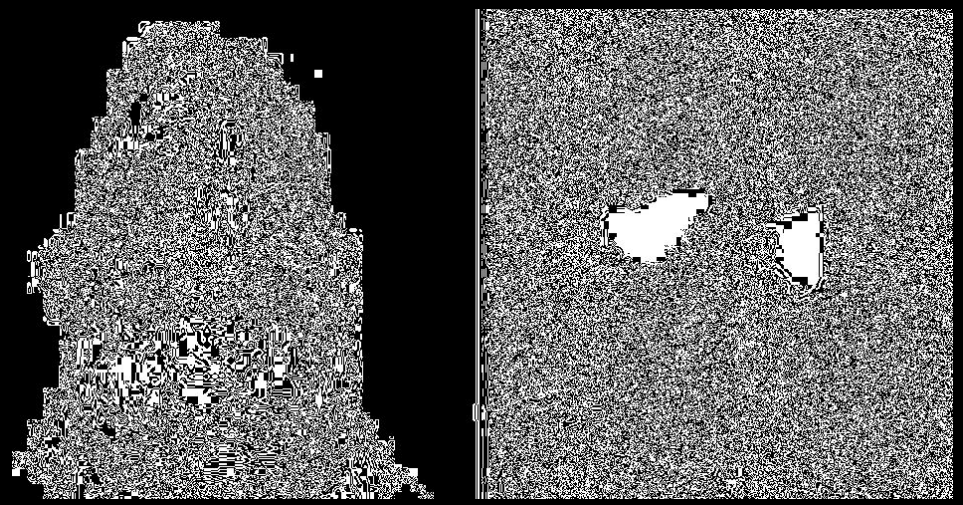
https://towardsdatascience.com/steganography-hiding-an-image-inside-another-77ca66b2acb1

Can you tell the difference?

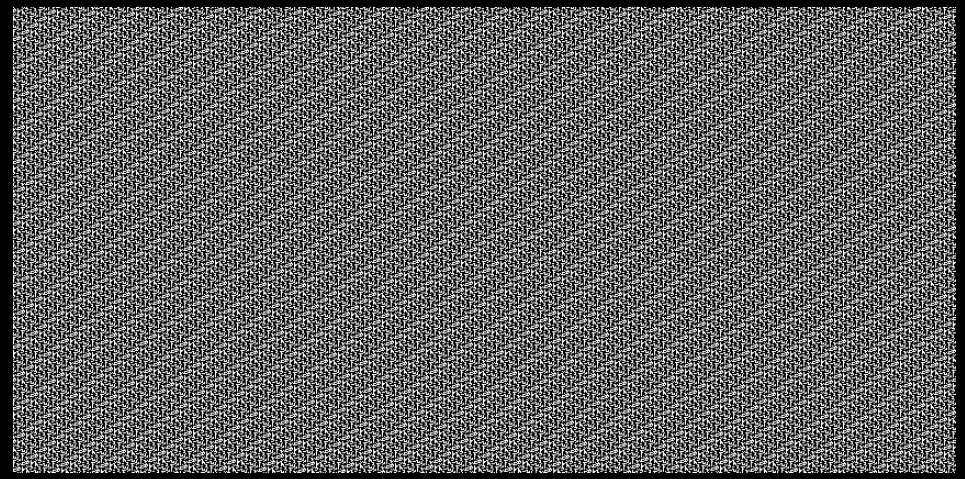
Original

LSB Encoded

Can you tell the difference?



Original (Red, Bit 0)



LSB Encoded (Red, Bit 0)

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

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

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?)
  - 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

Filesystem 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

Windows:

- Event Viewer

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)
  - 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

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
- They can contain sensitive information that was located in memory at the time of crashing
  - Passwords in your password manager
  - Clipboard content at time of crash

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> windows.pslist
```

```
$ python3 vol.py -f MEMORY.DMP windows.pslist
Volatility 3 Framework 2.4.1
Progress:  100.00               PDB scanning finished
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     Disabled88      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
```

Or... just use basic commands

A lot of memory is simply stored as strings!

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

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

Resources

Windows filesystem artifacts:
https://www.sans.org/posters/windows-forensic-analysis/

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

Next Meetings

2022-11-19 - This Saturday

- Sunshine CTF 2022
- Join us virtually to compete in UCF's CTF!

2022-12-01 - Thursday After Break

- PWN II with Kevin
- Learn how to exploit format strings for PWN

2022-12-04 - Sunday After Break

- Graduate talk with Jaron Mink
- Security Unleashed Part 4

SIGPwny