# Digital Forensics Focus Area

Barbara Guttman
Forensics@NIST
November 6, 2020

# Digital Forensics – Enormous Scale

- Computer crime is now a big volume crime (even worse now that everyone is online) - both in the number of cases and the impacts of the crimes.
  - Estimate of 6000% increase in spam (claiming to be PPP, WHO)
- Most serious crimes have a nexus to digital forensics: Drug dealing uses phones and drones. Murderers communicate with their victims. Financial fraud uses computers. Child sexual exploitation is recorded and shared online.
  - 2018: Facebook sent 45 million images to LE
- Digital Forensics used to support investigations and prosecutions
  - Used by Forensics Labs, Lawyers, Police

# Digital Forensics Overview

Digital Forensics goal: Provide trustworthy, useful and timely information

Digital Forensics has several problems meeting this goal:

1. Overwhelmed with the volume of material
2. Overwhelmed with the variety of material, the constant change and the technical skills needed to understand the material

Digital Forensics needs:

1. High quality tools and techniques
2. Help with operational quality and efficiency

# Digital Forensic Projects

- National Software Reference Library
- Computer Forensics Tool Testing
- Federated Testing
- Computer Forensics Reference Dataset
- Tool Catalog
- Black Box Study and Digital Forensics Scientific Foundation

# National Software Reference Library (NSRL)

The National Software Reference Library (NSRL):

- Collects software
- Populates a database with software metadata, individual files, file hashes
- Researches software identification
- Publishes data (Reference Data Set & other datasets)

- NSRL used daily by virtually all computer forensics labs
- Included in major computer forensics tools
- Most common uses:  Alert/Ignore

# Computer Forensics Tool Testing (CFTT)

Do Digital Forensics Tool Work?

- What are the tools doing?
- Are they good enough to provide evidence in court?
- Do they have limitations examiners should know about?

CFTT:

- Develops specifications
- Tests tools
- Creates material so others can test locally

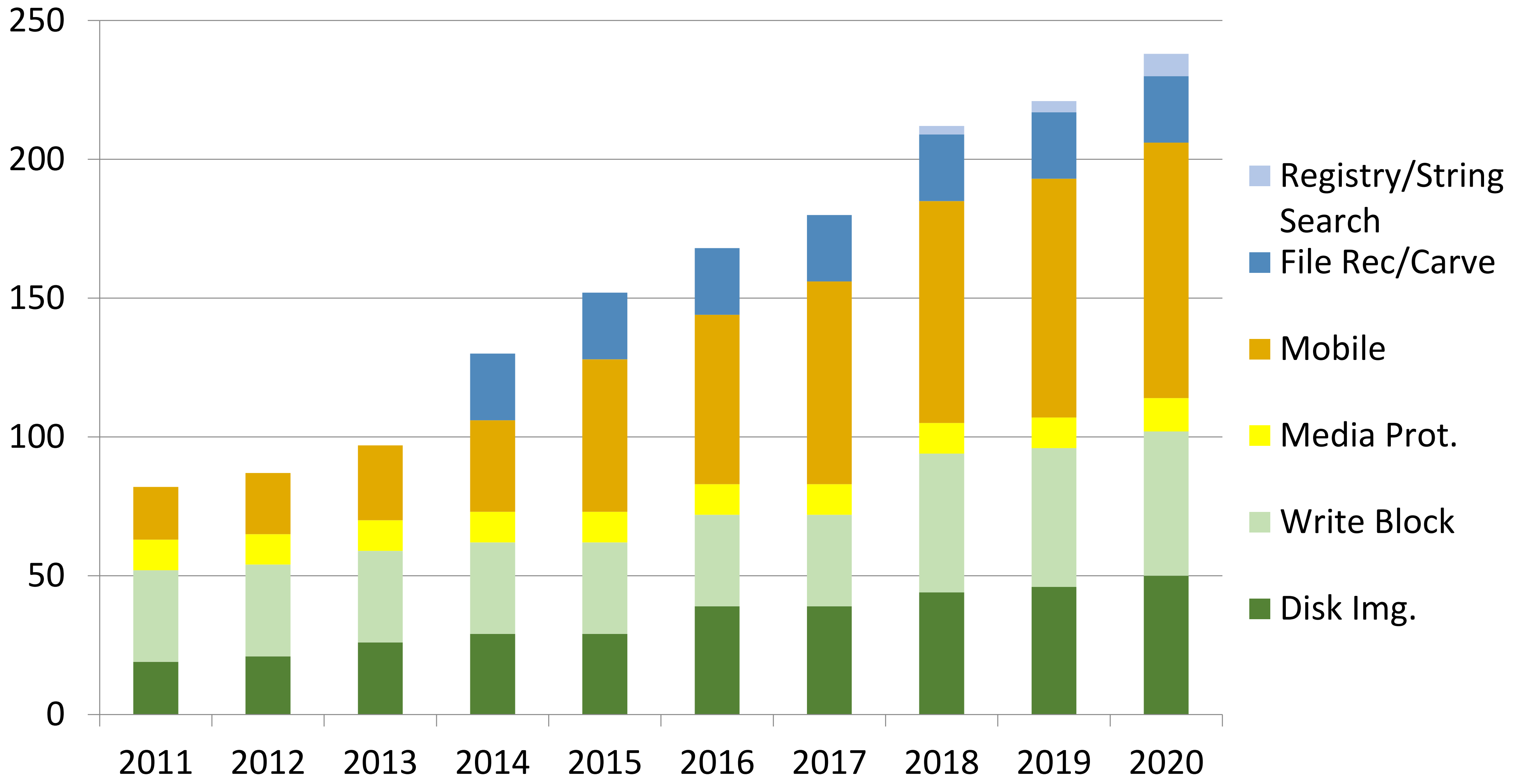
# Benefits of CFTT

- Tool creators make better tools
- Users can make informed choices
- Reduce challenges to admissibility of digital evidence
- Support lab-based validation of tools and accreditation

# CFTT Areas

- Evidence Acquisition and Preservation
  - Disk Imaging
  - Write Blocking
  - Disk Reuse
- Analysis
  - String Searching
  - Deleted File Recovery
  - File Carving – images
  - File Carving – video
  - Windows Registry
- Mobile
  - Logical/Physical Extraction and Analysis
  - JTAAG
  - SQLite Recovery

# Current Status CFTT



# Federated Testing

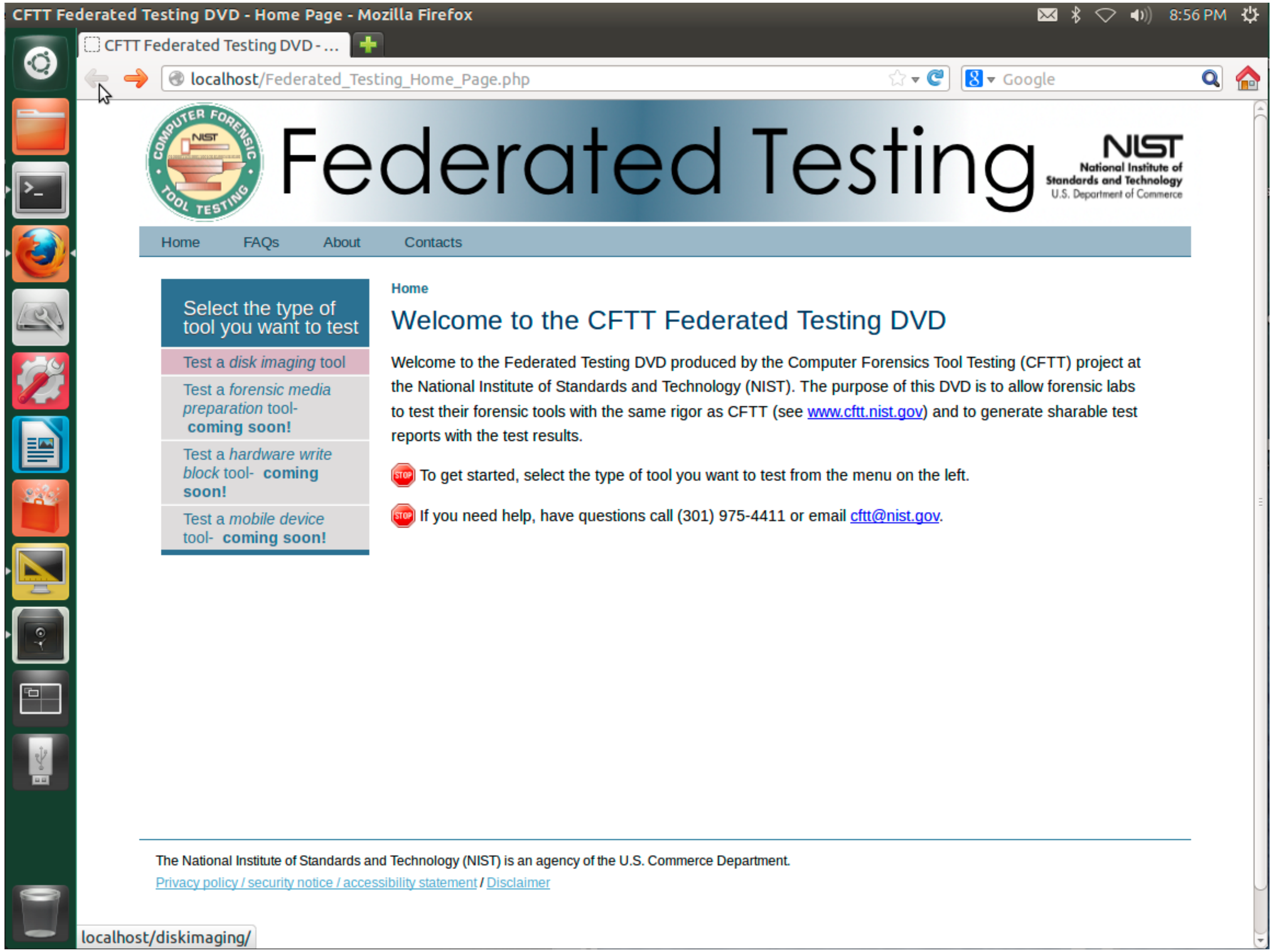
- Tool testing is expensive (time & resources)
  - CFTT only tests 20 or so products per year
  - Testing is a key part of quality management in a forensics lab
- Barriers exist which prevent sharing of test results
  - Labs test differently
  - Quality is unknown
  - Dissimilar report formats

# Federated Testing

- Shared test materials from CFTT
  - Use a common test methodology
  - Use a common test report format
  - Can be shared
- Goals
  - More tools validated
  - Shared test reports = cost savings = faster
  - Allows vendors to improve their tools
  - Helps users to make informed choices
  - Allows labs to mitigate known errors

# Federated Testing

- Modules
  - Mobile Phones
  - Disk Imaging
  - Write Blocking
  - Windows Registry
  - String Searching
- Infrastructure
  - Bootable Environment
  - Next gen environments
    - Interactive website
    - Considering others



CFTT Federated Testing DVD - Home Page - Mozilla Firefox
8:56 PM
CFTT Federated Testing DVD - ...
localhost/Federated_Testing_Home_Page.php
Google
COMPUTER FORENSIC TOOL TESTING
NIST
Federated Testing
NIST National Institute of Standards and Technology U.S. Department of Commerce
Home
FAQs
About
Contacts

Select the type of tool you want to test

- Test a *disk imaging* tool
- Test a *forensic media preparation* tool- **coming soon!**
- Test a *hardware write block* tool- **coming soon!**
- Test a *mobile device* tool- **coming soon!**

Home

## Welcome to the CFTT Federated Testing DVD

Welcome to the Federated Testing DVD produced by the Computer Forensics Tool Testing (CFTT) project at the National Institute of Standards and Technology (NIST). The purpose of this DVD is to allow forensic labs to test their forensic tools with the same rigor as CFTT (see www.cftt.nist.gov) and to generate sharable test reports with the test results.

STOP To get started, select the type of tool you want to test from the menu on the left.

STOP If you need help, have questions call (301) 975-4411 or email cftt@nist.gov.

The National Institute of Standards and Technology (NIST) is an agency of the U.S. Commerce Department.

Privacy policy / security notice / accessibility statement / Disclaimer

localhost/diskimaging/

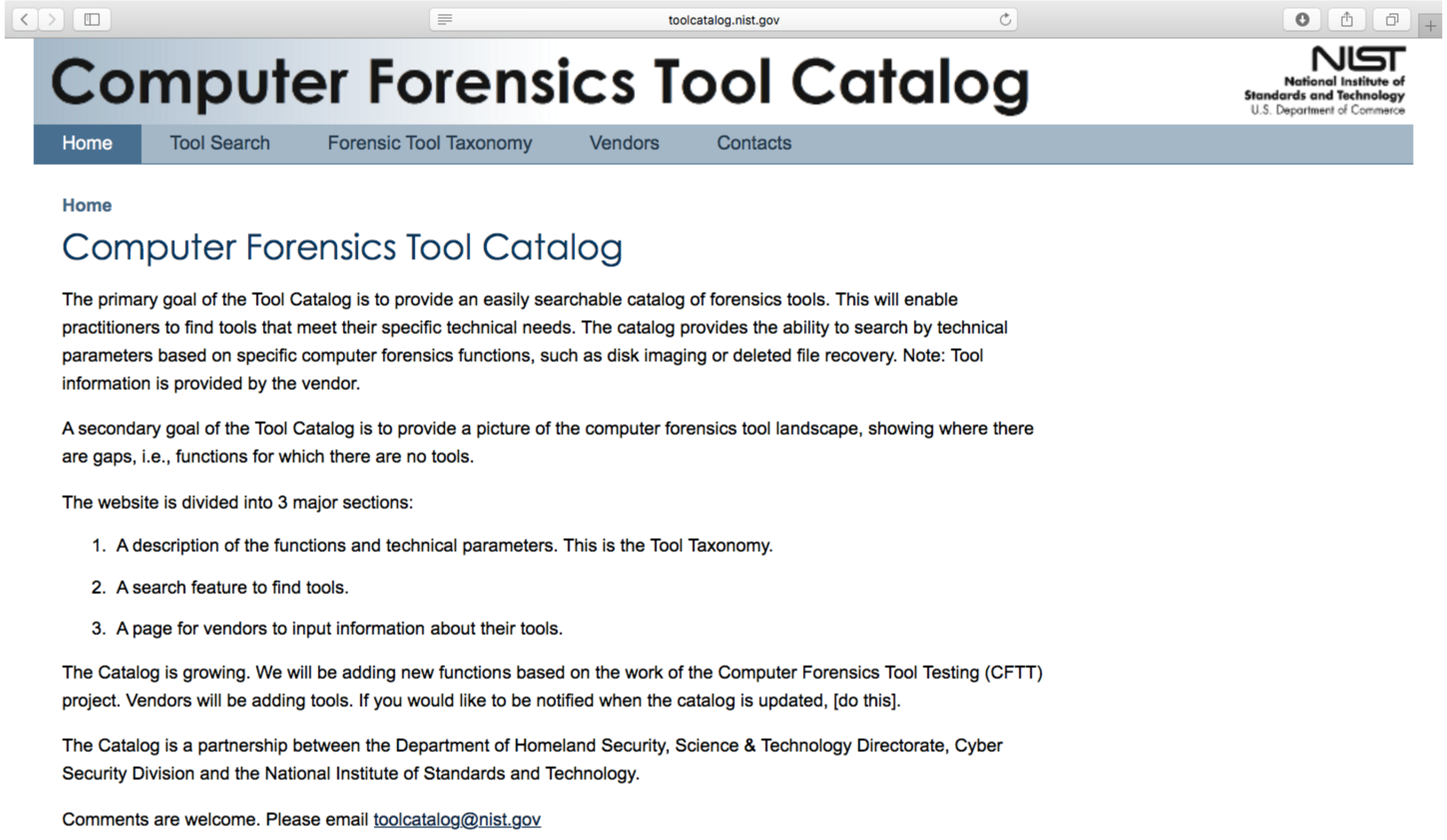
# Computer Forensics Reference Datasets (CFReDS)

- Datasets are difficult to create and useful
    - Testing
    - Training
    - Tool development
- Started a website with the datasets we created for CFTT
- Others contributed
- Community wanted more

- Building a new CFReDS

# Tool Catalog

- Vendor populated
- 37 Functionalities
- 78 Vendors
- 321 tool entries


toolcatalog.nist.gov
Computer Forensics Tool Catalog
NIST
National Institute of Standards and Technology
U.S. Department of Commerce
Home
Tool Search
Forensic Tool Taxonomy
Vendors
Contacts

Home

## Computer Forensics Tool Catalog

The primary goal of the Tool Catalog is to provide an easily searchable catalog of forensics tools. This will enable practitioners to find tools that meet their specific technical needs. The catalog provides the ability to search by technical parameters based on specific computer forensics functions, such as disk imaging or deleted file recovery. Note: Tool information is provided by the vendor.

A secondary goal of the Tool Catalog is to provide a picture of the computer forensics tool landscape, showing where there are gaps, i.e., functions for which there are no tools.

The website is divided into 3 major sections:

1. A description of the functions and technical parameters. This is the Tool Taxonomy.
2. A search feature to find tools.
3. A page for vendors to input information about their tools.

The Catalog is growing. We will be adding new functions based on the work of the Computer Forensics Tool Testing (CFTT) project. Vendors will be adding tools. If you would like to be notified when the catalog is updated, [do this].

The Catalog is a partnership between the Department of Homeland Security, Science & Technology Directorate, Cyber Security Division and the National Institute of Standards and Technology.

Comments are welcome. Please email toolcatalog@nist.gov

# How Do I Make the National Software Reference Library Hashes Fit My Needs?

Douglas White
NIST, Information Technology Laboratory

nsrl@nist.gov
www.nsrl.nist.gov

# Disclaimer

Commercial equipment, instruments, or materials may be identified in this paper to foster understanding.

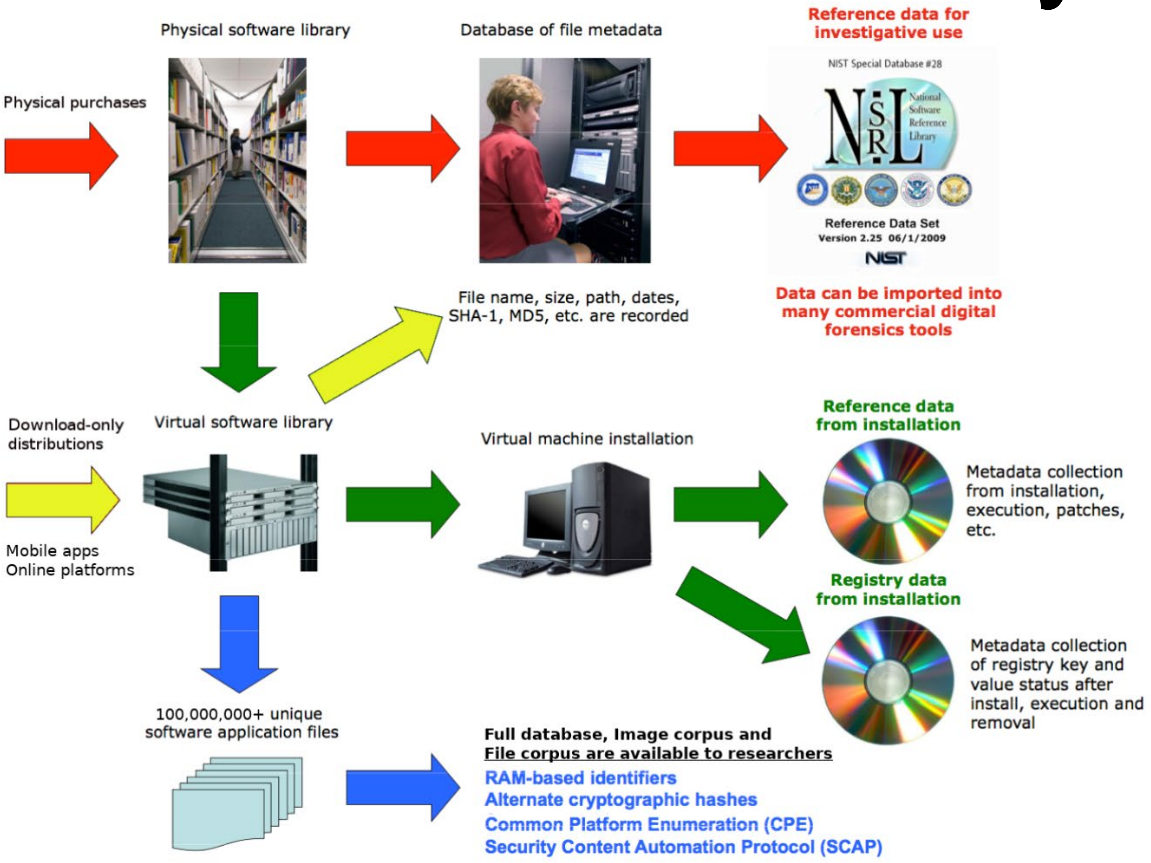
Such identification does not imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the materials or equipment identified are necessarily the best available for the purpose.

# What is the National Software Reference Library?

The NSRL is supported by federal, state, and local law enforcement, and NIST to promote efficient and effective use of computer technology in the investigation of crimes involving computers.

The NSRL collects software from many sources and incorporates file profiles computed from this software into a Reference Data Set (RDS). The RDS is used by law enforcement, government, and industry organizations to review unknown files by matching file profiles in the RDS. This alleviates much of the effort involved in determining which files are important as evidence on computers or file systems that have been seized as part of criminal investigations.

The RDS is a collection of digital signatures (hashes) of known, traceable software applications. There are application hash values in the hash set which may be considered malicious. There are no hash values of illicit data.


NIST Special Database #28
NSRL
National Software Reference Library
Reference Data Set
Version 2.25 06/1/2009
NIST

Physical purchases

Physical software library

Database of file metadata

Reference data for investigative use

File name, size, path, dates, SHA-1, MD5, etc. are recorded

Data can be imported into many commercial digital forensics tools

Download-only distributions

Mobile apps
Online platforms

Virtual software library

Virtual machine installation

Reference data from installation

Metadata collection from installation, execution, patches, etc.

Registry data from installation

Metadata collection of registry key and value status after install, execution and removal

100,000,000+ unique software application files

**Full database, Image corpus and File corpus are available to researchers**

RAM-based identifiers
Alternate cryptographic hashes
Common Platform Enumeration (CPE)
Security Content Automation Protocol (SCAP)

# What Contents are in the NSRL?

The NSRL acquires free software and purchases software through public commercial channels. Some vendors provide and allow NSRL to use unlimited licenses.

The majority of the software in the collection is built for microcomputers that run Windows, Mac OS, or Linux and mobile devices that run iOS or Android.

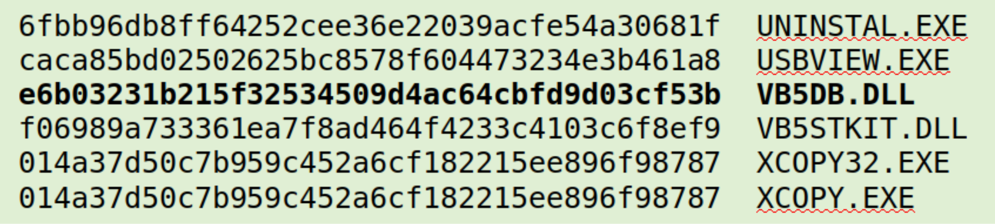
Acquisition is driven by popularity; titles or apps that are most likely to appear during investigations. The steering committee identifies classes of software to be acquired, e.g. keylogging.

Notable items in the collection other than standalone software are mobile phone images, online game platform titles, live system snapshots of updates.

# Basic Hashset Investigation

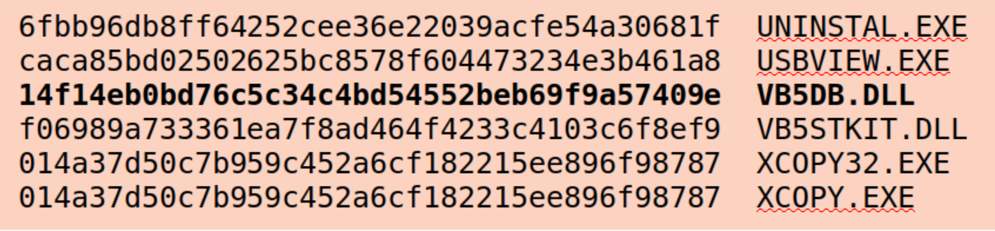
The most common use of the NSRL RDS comes via importing the data into a commercial digital forensics tool. The tool provides a user interface to automate comparison of file signatures (hashes) and filter the files under investigation into sets.

In RDS



```
6fbb96db8ff64252cee36e22039acfe54a30681f  UNINSTAL.EXE
caca85bd02502625bc8578f604473234e3b461a8  USBVIEW.EXE
e6b03231b215f32534509d4ac64cbfd9d03cf53b  VB5DB.DLL
f06989a733361ea7f8ad464f4233c4103c6f8ef9  VB5STKIT.DLL
014a37d50c7b959c452a6cf182215ee896f98787  XCOPY32.EXE
014a37d50c7b959c452a6cf182215ee896f98787  XCOPY.EXE
```

On PC



```
6fbb96db8ff64252cee36e22039acfe54a30681f  UNINSTAL.EXE
caca85bd02502625bc8578f604473234e3b461a8  USBVIEW.EXE
14f14eb0bd76c5c34c4bd54552beb69f9a57409e  VB5DB.DLL
f06989a733361ea7f8ad464f4233c4103c6f8ef9  VB5STKIT.DLL
014a37d50c7b959c452a6cf182215ee896f98787  XCOPY32.EXE
014a37d50c7b959c452a6cf182215ee896f98787  XCOPY.EXE
```

# Available Metadata

Four files are published, which may be imported into a spreadsheet or database application. The highlighted codes form the relations between the files.

```
"ProductCode","ProductName","ProductVersion",
          "OpSystemCode","MfgCode","Language","ApplicationType"
```

```
"MfgCode","MfgName"
```

```
"OpSystemCode","OpSystemName","OpSystemVersion","MfgCode"
```

```
"SHA-1","MD5","CRC32",
          "FileName","FileSize","ProductCode","OpSystemCode","SpecialCode"
```

# Advanced Investigations

- Identify the titles of possible software
- Identify possible operating systems
- Identify the versions of software
- Create a data subset for notable software

# Identify the Titles of Possible Software

A file with a notable name is found; to which titles might it belong?

“**DirtyBombLauncher.exe**” is found, and it has

SHA1 “40D4FA74C353632B92853A164DF05BAC92D7B8B0”

Knowing that, the "ProductCode” can be found – 89715.

Looking up the "ProductCode” shows the product metadata is

89715,"Dirty Bomb","2993087","189","80811","Chinese,English,Russian","Game"

And the “MfgCode” can be used to show the manufacturer

"80811","Splash Damage"

It may be useful to know that files from a multilingual first-person shooter game called “Dirty Bomb” by Splash Damage were found on the computer

# Identify Possible Operating Systems

It may be of interest to know if a computer or device has hosted multiple operating systems.

Once all of the systems' files have been hashed, the OpSystemCode can be found for all files.

The possible operating systems and versions can be listed, based on the OpSystemCode.

If virtual machine disks or mobile device images are open to the forensics tools, the operating systems of those may be determined.

# Identify the Versions of Software

Several vulnerability scanner resource files are found. To which versions of the software do they correspond?

```
"0E0873EA2C068E3F9207BC8EF06987D6C17F28C3","DE7185CD3979B617BCC91FA72504F088","0E1BE534","centos_RHSA-2018-1099.nasl",3428,202144,"362",""

"02A005D08854A6217C128B3A679E3444DF5518F2","17124317BEE392B53455433803BD8D29","BD72BDF9","centos_RHSA-2018-2251.nasl",4753,202145,"362",""

"02A005D08854A6217C128B3A679E3444DF5518F2","17124317BEE392B53455433803BD8D29","BD72BDF9","centos_RHSA-2018-2251.nasl",4753,202146,"362",""
```

# Identify the Versions of Software

The ProductCode values show the files belong to

202144,"Security Center - RPM","5.7.0","336","82000","English","Security"

202145,"Security Center - RPM","5.7.1","336","82000","English","Security"

202146,"Security Center - RPM","5.7.1","936","82000","English","Security"

The MfgCode provides the title of the software.

"82000","Tenable, Inc."

# Create a Data Subset

The product, manufacturer or operating system metadata may be used to build sets of classes of software.

NIST does not determine if a class is notable; investigators may customize a set for their needs.

As an example, a set could be built describing the Kaspersky products which run on Windows 10 Enterprise.

Find the OpSystemCode values:

"872","Windows 10 Enterprise","Windows 10 Ente","5804"

"877","Windows 10 Enterprise x64","Windows 10 Ente","5804"

"878","Windows 10 Enterprise x32","Windows 10 Ente","5804"

# Create a Data Subset

Find the MfgCode values:

"1141","Kaspersky Lab Ltd"

"73190","Kaspersky Lab UK Limited"

"82180","Kaspersky Lab"

Find the products that have both the MfgCode and OpSystemCode values:

182935,"Kaspersky Secure Connection","18.0.0.405","872","1141","English","Security,vpn"

182938,"Kaspersky Free","18.0.0.405","872","1141","English","Security"

182939,"Kaspersky Security Scan","16.0.0.1344","872","1141","English","Security"

183685,"Kaspersky Password Manager","dl. 2017-08-29","872","1141","English","Password Protection"

# Create a Data Subset

Use the ProductCode values to make a smaller file metadata set.

```
182935,"Kaspersky Secure Connection","18.0.0.405","872","1141","English","Security,vpn"

182938,"Kaspersky Free","18.0.0.405","872","1141","English","Security"

182939,"Kaspersky Security Scan","16.0.0.1344","872","1141","English","Security"

183685,"Kaspersky Password Manager","dl. 2017-08-29","872","1141","English","Password Protection"
```

Example results:

```
"3835905A39FE620A5EC24E0DC047DAB8AC019187","CB8C2AA16B277AD0B932D65A311EFCB3","15E1C931",
"crypto_ssl.dll",1643504,182935,"358",""

"08BAAF15CEADFB4011E8E2A6444D7132EA1C066B","DFF9C3A9F2685689436B7FE2D4355C1B","36D81022",
"klsihk64l.dll",262840,182938,"358",""

"357D72A5817BC41F5668ED44C76C40BC3FC08115","85FAC8B134A576DA675BF5670511B7CF","5D007528",
"klsihk64.dll",262840,182938,"358",""

"28689E35649C0C8086413F23E12F92F89E4440EB","3622E4615D654EFE0BA43677A925FB6B","0CACF984",
"SETUP.DLL",6410240,182939,"358",""
```

# How Do I Make the National Software Reference Library Hashes Fit My Needs?

Douglas White
NIST, Information Technology Laboratory

nsrl@nist.gov
www.nsrl.nist.gov

# TESTING DIGITAL FORENSIC STRING SEARCH TOOLS

James R. Lyle

National Institute of Standards and Technology,
100 Bureau Drive Stop 8970,
Gaithersburg, MD 20899-8970

# CFTT

The CFTT project at NIST develops methodologies for testing computer forensic tools. Currently there are CFTT methodologies for testing the following:

- Disk imaging*
- Write blocking*
- Forensic Media Preparation*
- String Searching*
- Deleted File Recovery
- File Carving
- Mobile Devices*

* Starred methods have been incorporated into Federated Testing, a downloadable CD to guide a practitioner through testing tools

# Overview

- Testing String Search Tools
- Describing a test case in Federated Testing
- Examples of test results
  - Easy case
  - Easy case with unexpected interactions
  - Unicode
  - Meta-data
  - Built-in searches
  - Formatted text
- Lessons learned & Observations

# How to do a Test and What to Test?

- Need some test data -- basic idea
  - Put some strings on a hard drive
  - Make an image of the drive
  - Document the location of the strings; define expected results
  - Run the search tool, see if it can find the strings
- What does find a string mean? & What should the tool report?
  - Location of match: file name, byte offset from somewhere
  - Actual string matched – may be searching with some option (e.g., ignore case)
- Some things that might matter for string searching:
  - Tool Settings: match case vs ignore case & word vs substring
  - Data Encoding: ASCII, UTF-8, UTF-16 (BE or LE)
  - What are the special cases? NTFS, meta-data, stemming

# Test Logistics

- For string searching, CFTT provides test images with known content and a list of test cases designed to test specific features.

1. Tester can select relevant test cases from a list of test cases
2. Each case is run by first setting tool options and then searching for a string
3. Record search results
4. Generate a test report.

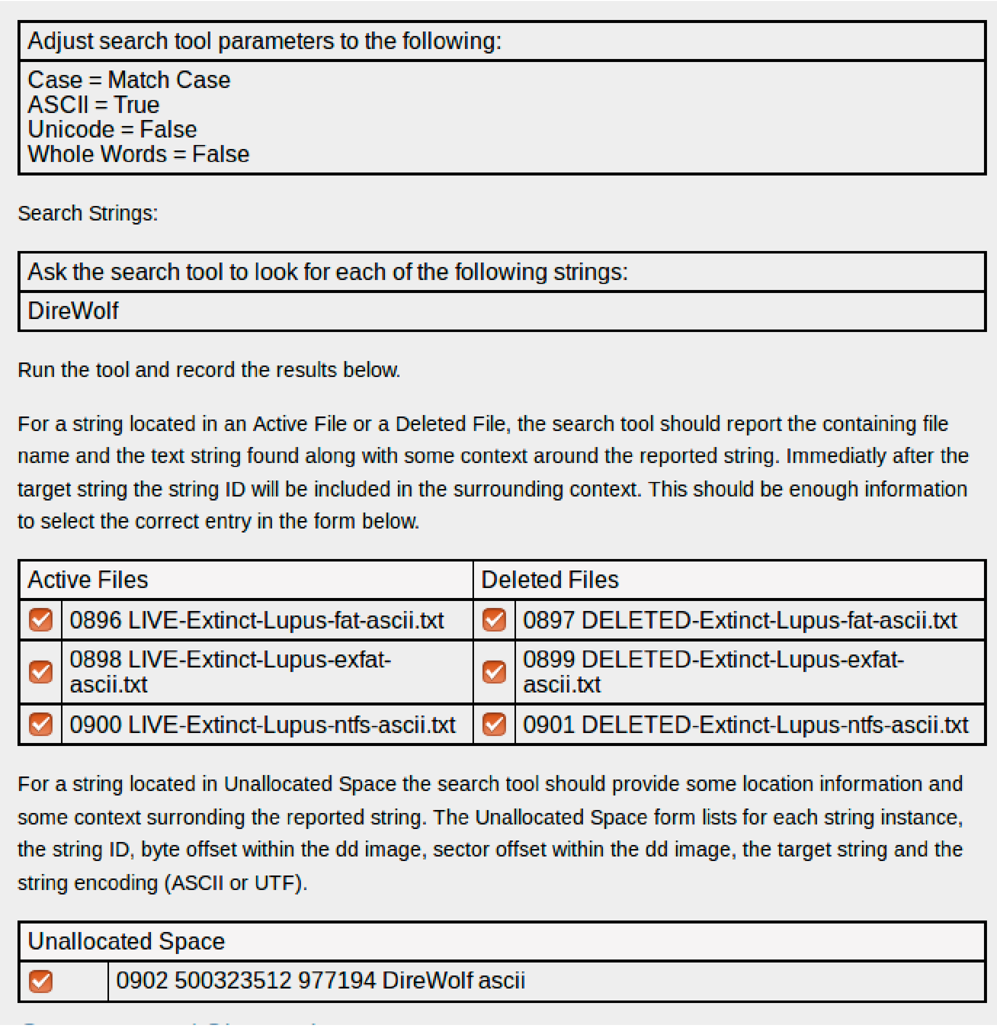
# A basic test case

| Case | Strings | Options | Case Description |
|---|---|---|---|
| FT-SS-01 | DireWolf | Case = Match Case<br>ASCII = True<br>Unicode = False<br>Whole Words = False | Search ASCII |

| ID | Offset | Containing File Name |
|---|---:|---|
| **0897** | 8,197,307 | DELETED-Extinct-Lupus-fat-ascii.txt |
| **0896** | 9,172,152 | LIVE-Extinct-Lupus-fat-ascii.txt |
| **0902** | 500,323,512 | LIVE-Extinct-Lupus-unalloc-ascii.txt |
| **0899** | 1,000,839,354 | DELETED-Extinct-Lupus-exfat-ascii.txt |
| **0898** | 1,001,613,487 | LIVE-Extinct-Lupus-exfat-ascii.txt |
| **0900** | 1,504,877,750 | LIVE-Extinct-Lupus-ntfs-ascii.txt |
| **0901** | 1,666,325,693 | DELETED-Extinct-Lupus-ntfs-ascii.txt |

- Test image has 4 partitions: FAT, Unformatted, ExFAT & NTFS
- Test strings appear multiple (in this case 7) times with something different about each instance
- The search string appears twice in each formatted partition, once in unallocated space
- Each instance of the string has a unique ID, placed just after the string

# Test Case Summary



| Adjust search tool parameters to the following: |
|---|
| Case = Match Case<br>ASCII = True<br>Unicode = False<br>Whole Words = False |

Search Strings:

| Ask the search tool to look for each of the following strings: |
|---|
| DireWolf |

Run the tool and record the results below.

For a string located in an Active File or a Deleted File, the search tool should report the containing file name and the text string found along with some context around the reported string. Immediatly after the target string the string ID will be included in the surrounding context. This should be enough information to select the correct entry in the form below.

| Active Files | | Deleted Files | |
|---|---|---|---|
| ☑ | 0896 LIVE-Extinct-Lupus-fat-ascii.txt | ☑ | 0897 DELETED-Extinct-Lupus-fat-ascii.txt |
| ☑ | 0898 LIVE-Extinct-Lupus-exfat-ascii.txt | ☑ | 0899 DELETED-Extinct-Lupus-exfat-ascii.txt |
| ☑ | 0900 LIVE-Extinct-Lupus-ntfs-ascii.txt | ☑ | 0901 DELETED-Extinct-Lupus-ntfs-ascii.txt |

For a string located in Unallocated Space the search tool should provide some location information and some context surrounding the reported string. The Unallocated Space form lists for each string instance, the string ID, byte offset within the dd image, sector offset within the dd image, the target string and the string encoding (ASCII or UTF).

| Unallocated Space | |
|---|---|
| ☑ | 0902 500323512 977194 DireWolf ascii |

- Specifies what search options to select
- Specifies what string or pattern to search for
- Presents expected results – after running the search select the checkboxes to record all strings found
- Record false hits and other notable behavior in a comment text box (not shown)

# What We Selected to Test

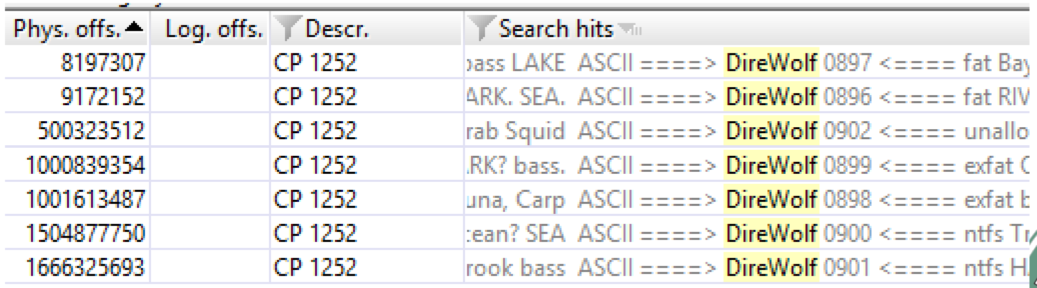
- *Match case* vs *ignore case*
- Match whole words vs substrings
- Search method: indexed vs live vs physical
- File systems: FAT32, ExFAT, NTFS, ext4, OSXJ, OSXC & APFS
- Encoding: ASCII, UTF-8, UTF-16 (BE & LE) with & without byte-order-mark
- Language: CJK, Latin with diacritics, non-Latin, right-to-left
- Live Files vs Deleted Files vs Unallocated Space
- Logical expressions
- Regular expressions
- Special Cases
  - Meta-data
  - Formatted documents (.doc, .docx, .html)
  - Small files in NTFS $MFT
  - Search target spans fragmentation
  - Stemming

# Simple Example

- Try to find a DireWolf (just in case “Winter is Coming”)
- Expected Results: 7 hits

| ID | String | Offset | File Name |
|---|---|---|---|
| 0897 | DireWolf | 8,197,307 | DELETED-Extinct-Lupus-fat-ascii.txt |
| 0896 | DireWolf | 9,172,152 | LIVE-Extinct-Lupus-fat-ascii.txt |
| 0902 | DireWolf | 500,323,512 | LIVE-Extinct-Lupus-unalloc-ascii.txt |
| 0899 | DireWolf | 1,000,839,354 | DELETED-Extinct-Lupus-exfat-ascii.txt |
| 0898 | DireWolf | 1,001,613,487 | LIVE-Extinct-Lupus-exfat-ascii.txt |
| 0900 | DireWolf | 1,504,877,750 | LIVE-Extinct-Lupus-ntfs-ascii.txt |
| 0901 | DireWolf | 1,666,325,693 | DELETED-Extinct-Lupus-ntfs-ascii.txt |

Found 7 hits; this is what I Expected:
(Tool screen shot)



| Phys. offs. | Log. offs. | Descr. | Search hits |
|---|---|---|---|
| 8197307 | | CP 1252 | ass LAKE ASCII ====> DireWolf 0897 <==== fat Bay |
| 9172152 | | CP 1252 | ARK. SEA. ASCII ====> DireWolf 0896 <==== fat RIV |
| 500323512 | | CP 1252 | rab Squid ASCII ====> DireWolf 0902 <==== unallo |
| 1000839354 | | CP 1252 | .RK? bass. ASCII ====> DireWolf 0899 <==== exfat C |
| 1001613487 | | CP 1252 | una, Carp ASCII ====> DireWolf 0898 <==== exfat b |
| 1504877750 | | CP 1252 | ean? SEA ASCII ====> DireWolf 0900 <==== ntfs Tr |
| 1666325693 | | CP 1252 | rook bass ASCII ====> DireWolf 0901 <==== ntfs H |

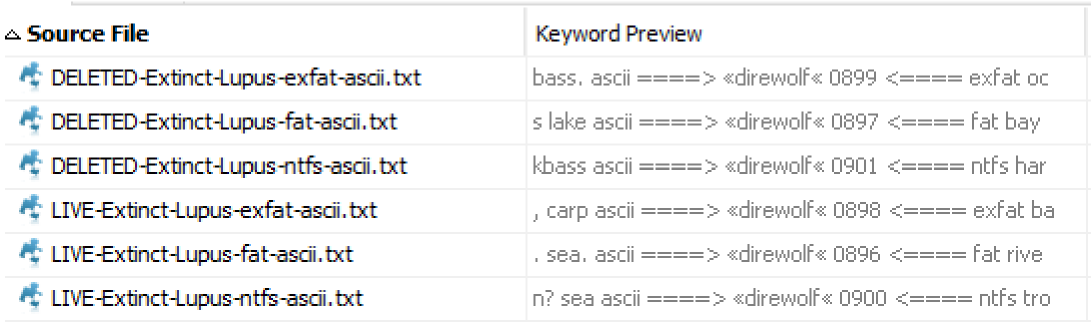
Wow, this is easy & simple. Are We Done?

What else should be tested?

# Tool Settings Matter

Results first try:

Oops, 6 hits, Did we miss one?



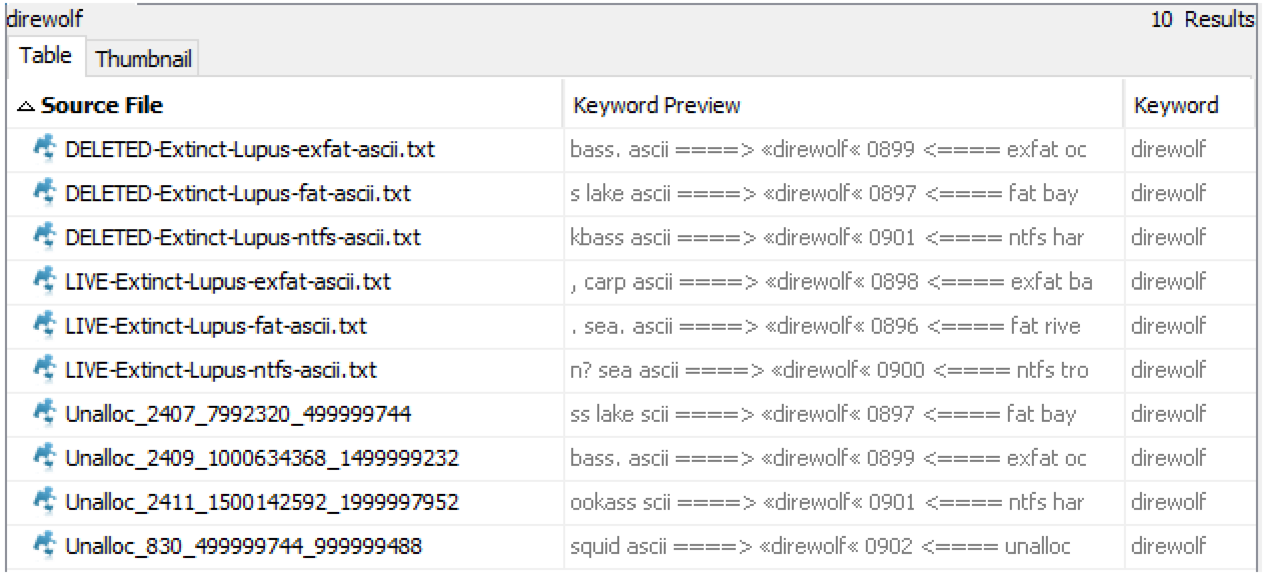
| △ Source File | Keyword Preview |
|---|---|
| DELETED-Extinct-Lupus-exfat-ascii.txt | bass. ascii ====> «direwolf« 0899 <==== exfat oc |
| DELETED-Extinct-Lupus-fat-ascii.txt | s lake ascii ====> «direwolf« 0897 <==== fat bay |
| DELETED-Extinct-Lupus-ntfs-ascii.txt | kbass ascii ====> «direwolf« 0901 <==== ntfs har |
| LIVE-Extinct-Lupus-exfat-ascii.txt | , carp ascii ====> «direwolf« 0898 <==== exfat ba |
| LIVE-Extinct-Lupus-fat-ascii.txt | . sea. ascii ====> «direwolf« 0896 <==== fat rive |
| LIVE-Extinct-Lupus-ntfs-ascii.txt | n? sea ascii ====> «direwolf« 0900 <==== ntfs tro |

Second try:

Now 10 hits, too many?

3 hits are reported twice!

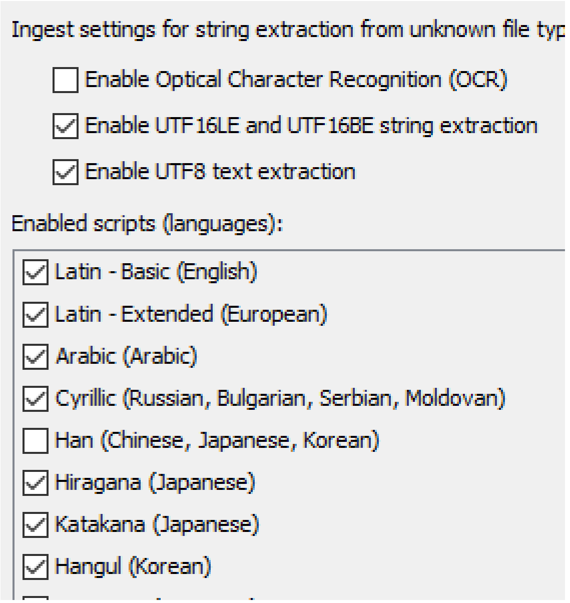
String in deleted file is also reported as unallocated space!!



| △ Source File | Keyword Preview | Keyword |
|---|---|---|
| DELETED-Extinct-Lupus-exfat-ascii.txt | bass. ascii ====> «direwolf« 0899 <==== exfat oc | direwolf |
| DELETED-Extinct-Lupus-fat-ascii.txt | s lake ascii ====> «direwolf« 0897 <==== fat bay | direwolf |
| DELETED-Extinct-Lupus-ntfs-ascii.txt | kbass ascii ====> «direwolf« 0901 <==== ntfs har | direwolf |
| LIVE-Extinct-Lupus-exfat-ascii.txt | , carp ascii ====> «direwolf« 0898 <==== exfat ba | direwolf |
| LIVE-Extinct-Lupus-fat-ascii.txt | . sea. ascii ====> «direwolf« 0896 <==== fat rive | direwolf |
| LIVE-Extinct-Lupus-ntfs-ascii.txt | n? sea ascii ====> «direwolf« 0900 <==== ntfs tro | direwolf |
| Unalloc_2407_7992320_499999744 | ss lake scii ====> «direwolf« 0897 <==== fat bay | direwolf |
| Unalloc_2409_1000634368_1499999232 | bass. ascii ====> «direwolf« 0899 <==== exfat oc | direwolf |
| Unalloc_2411_1500142592_1999997952 | ookass scii ====> «direwolf« 0901 <==== ntfs har | direwolf |
| Unalloc_830_499999744_999999488 | squid ascii ====> «direwolf« 0902 <==== unalloc | direwolf |

# Tool Search Settings

- This tool has selections for searching and indexing **Unallocated Space**
- If we select Han (i.e., Chinese Character 汉子), the ASCII string is not found in unallocated space
- If we unselect UTF-8 & UTF-16, ASCII string not found in unallocated space


Ingest settings for string extraction from unknown file typ

□ Enable Optical Character Recognition (OCR)

☑ Enable UTF16LE and UTF16BE string extraction

☑ Enable UTF8 text extraction

Enabled scripts (languages):

☑ Latin - Basic (English)

☑ Latin - Extended (European)

☑ Arabic (Arabic)

☑ Cyrillic (Russian, Bulgarian, Serbian, Moldovan)

□ Han (Chinese, Japanese, Korean)

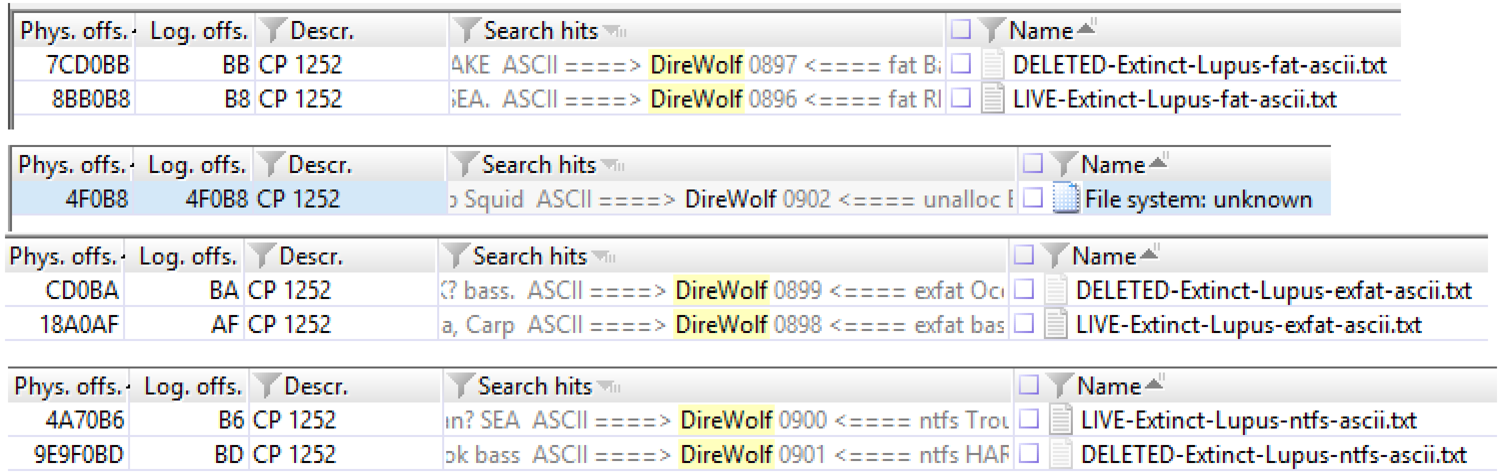
☑ Hiragana (Japanese)

☑ Katakana (Japanese)

☑ Hangul (Korean)

# Search Method Matters

- That First search was a physical search, one sector at a time.
- With this tool you can also search one file at a time: Logical Search



| Phys. offs. | Log. offs. | Descr. | Search hits | □ | Name |
|---|---|---|---|---|---|
| 7CD0BB | BB | CP 1252 | AKE  ASCII ====> DireWolf 0897 <==== fat Bi | □ | DELETED-Extinct-Lupus-fat-ascii.txt |
| 8BB0B8 | B8 | CP 1252 | EA.  ASCII ====> DireWolf 0896 <==== fat RI | □ | LIVE-Extinct-Lupus-fat-ascii.txt |

| Phys. offs. | Log. offs. | Descr. | Search hits | □ | Name |
|---|---|---|---|---|---|
| 4F0B8 | 4F0B8 | CP 1252 | ɔ Squid  ASCII ====> DireWolf 0902 <==== unalloc | □ | File system: unknown |

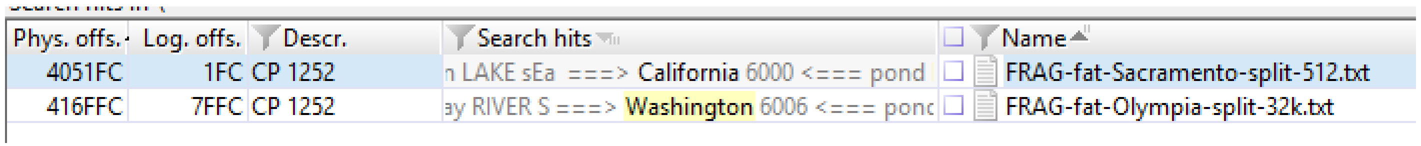
| Phys. offs. | Log. offs. | Descr. | Search hits | □ | Name |
|---|---|---|---|---|---|
| CD0BA | BA | CP 1252 | ? bass.  ASCII ====> DireWolf 0899 <==== exfat Oc | □ | DELETED-Extinct-Lupus-exfat-ascii.txt |
| 18A0AF | AF | CP 1252 | a, Carp  ASCII ====> DireWolf 0898 <==== exfat bas | □ | LIVE-Extinct-Lupus-exfat-ascii.txt |

| Phys. offs. | Log. offs. | Descr. | Search hits | □ | Name |
|---|---|---|---|---|---|
| 4A70B6 | B6 | CP 1252 | ın? SEA  ASCII ====> DireWolf 0900 <==== ntfs Trou | □ | LIVE-Extinct-Lupus-ntfs-ascii.txt |
| 9E9F0BD | BD | CP 1252 | ɔk bass  ASCII ====> DireWolf 0901 <==== ntfs HAR | □ | DELETED-Extinct-Lupus-ntfs-ascii.txt |

- This tool can also do an indexed search

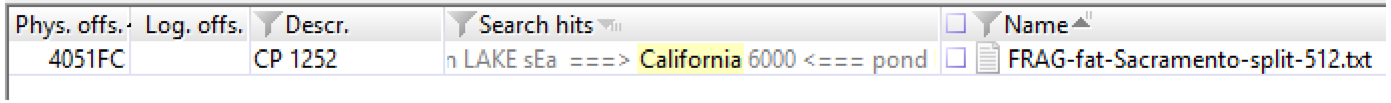
# Physical Search has Limitations

- There is a file, “Olympia” with a string, “Washington”, that crosses a file storage unit (cluster) boundary.
- Logical Search:



| Phys. offs. | Log. offs. | Descr. | Search hits | □ | Name |
|---|---|---|---|---|---|
| 4051FC | 1FC | CP 1252 | n LAKE sEa ===> California 6000 <=== pond | □ | FRAG-fat-Sacramento-split-512.txt |
| 416FFC | 7FFC | CP 1252 | ay RIVER S ===> Washington 6006 <=== ponc | □ | FRAG-fat-Olympia-split-32k.txt |

- Physical Search:



| Phys. offs. | Log. offs. | Descr. | Search hits | □ | Name |
|---|---|---|---|---|---|
| 4051FC | | CP 1252 | n LAKE sEa ===> California 6000 <=== pond | □ | FRAG-fat-Sacramento-split-512.txt |

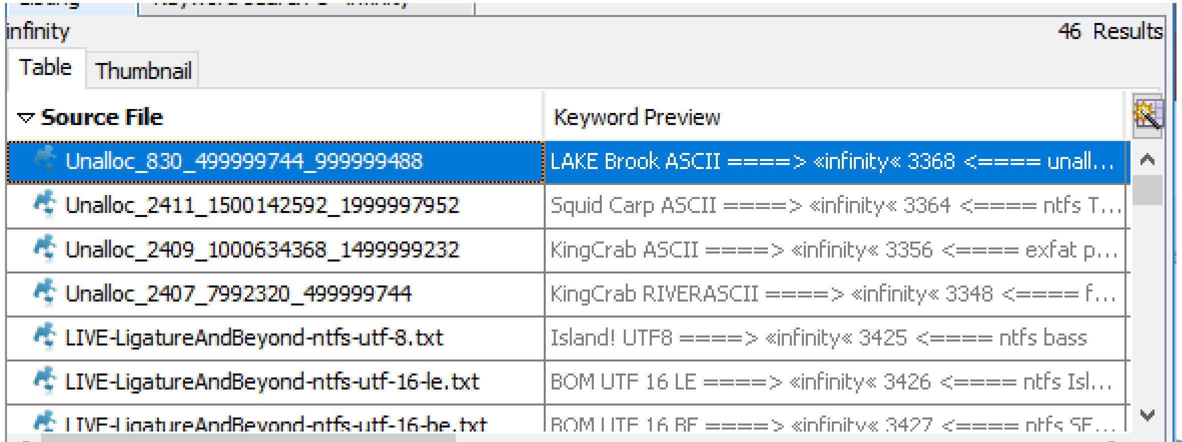
# New Topic: What’s a Ligature?

- Compare:
  - Infinity
  - Inﬁnity
- English has several ligatures: ff, fl, ffi, ffl, Æ, æ, Œ, . . ., etc
- A single Unicode byte code may represent more than one letter

- This happens in German, French, Spanish or Japanese . . . .
- Umlaut, accents, tilde . . .

# Where is Buzz Lightyear?

- Buzz is trying to get to “infinity” (and maybe beyond). . .
- Expected Results: 49 unique strings.
- Let’s try a search tool . . .
- Tool reports 46 strings, but . . .
- 4 of the hits are in  unalloc space
- 3 of these hits are duplicates of hits in deleted files (46 – 3 => 43)
- The other unallocated hits should have 7 hits (43 + 7 – 1 => 49)

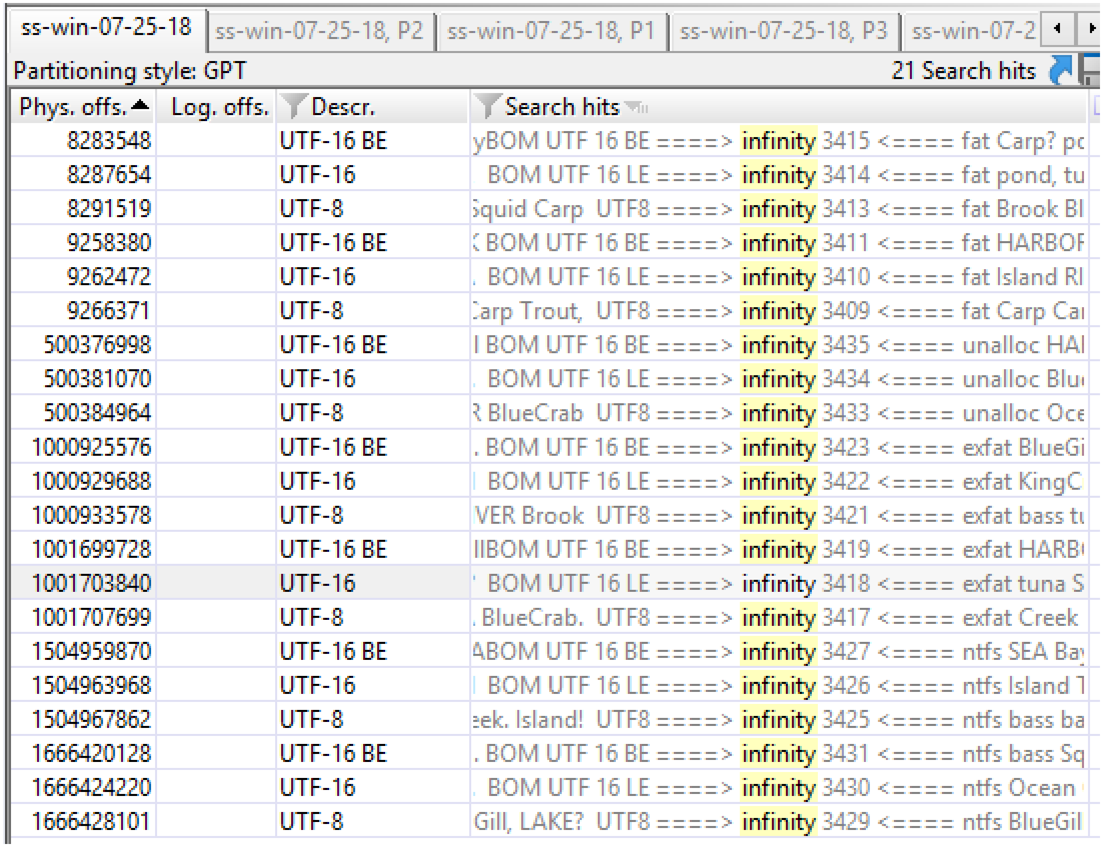


infinity 46 Results

| Source File | Keyword Preview |
|---|---|
| Unalloc_830_499999744_999999488 | LAKE Brook ASCII ====> «infinity« 3368 <==== unall… |
| Unalloc_2411_1500142592_1999997952 | Squid Carp ASCII ====> «infinity« 3364 <==== ntfs T… |
| Unalloc_2409_1000634368_1499999232 | KingCrab ASCII ====> «infinity« 3356 <==== exfat p… |
| Unalloc_2407_7992320_499999744 | KingCrab RIVERASCII ====> «infinity« 3348 <==== f… |
| LIVE-LigatureAndBeyond-ntfs-utf-8.txt | Island! UTF8 ====> «infinity« 3425 <==== ntfs bass |
| LIVE-LigatureAndBeyond-ntfs-utf-16-le.txt | BOM UTF 16 LE ====> «infinity« 3426 <==== ntfs Isl… |
| LIVE-LigatureAndBeyond-ntfs-utf-16-be.txt | BOM UTF 16 BE ====> «infinity« 3427 <==== ntfs SF… |

# More Buzz

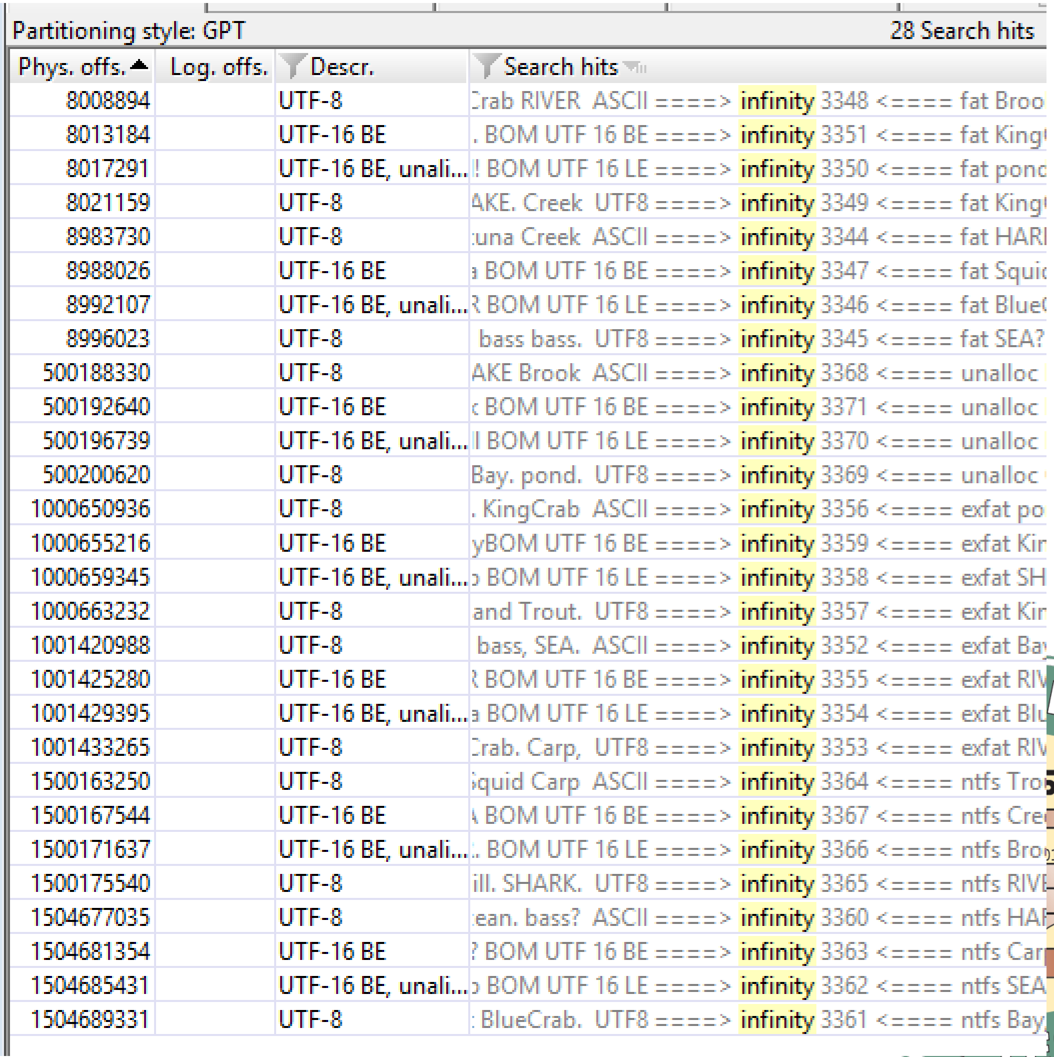
Tool results:
Of the 49 expected hits, 21 hits are with ligature and 28 Hits are without a ligature.



ss-win-07-25-18 | ss-win-07-25-18, P2 | ss-win-07-25-18, P1 | ss-win-07-25-18, P3 | ss-win-07-2

Partitioning style: GPT 21 Search hits

| Phys. offs. ▲ | Log. offs. | Descr. | Search hits |
|---|---|---|---|
| 8283548 | | UTF-16 BE | yBOM UTF 16 BE ====> infinity 3415 <==== fat Carp? pc |
| 8287654 | | UTF-16 | BOM UTF 16 LE ====> infinity 3414 <==== fat pond, tu |
| 8291519 | | UTF-8 | Squid Carp UTF8 ====> infinity 3413 <==== fat Brook Bl |
| 9258380 | | UTF-16 BE | ‹ BOM UTF 16 BE ====> infinity 3411 <==== fat HARBOF |
| 9262472 | | UTF-16 | . BOM UTF 16 LE ====> infinity 3410 <==== fat Island RI |
| 9266371 | | UTF-8 | Carp Trout, UTF8 ====> infinity 3409 <==== fat Carp Ca |
| 500376998 | | UTF-16 BE | I BOM UTF 16 BE ====> infinity 3435 <==== unalloc HAI |
| 500381070 | | UTF-16 | . BOM UTF 16 LE ====> infinity 3434 <==== unalloc Blu |
| 500384964 | | UTF-8 | R BlueCrab UTF8 ====> infinity 3433 <==== unalloc Oce |
| 1000925576 | | UTF-16 BE | . BOM UTF 16 BE ====> infinity 3423 <==== exfat BlueGi |
| 1000929688 | | UTF-16 | I BOM UTF 16 LE ====> infinity 3422 <==== exfat KingC |
| 1000933578 | | UTF-8 | VER Brook UTF8 ====> infinity 3421 <==== exfat bass tu |
| 1001699728 | | UTF-16 BE | IIBOM UTF 16 BE ====> infinity 3419 <==== exfat HARB |
| 1001703840 | | UTF-16 | ' BOM UTF 16 LE ====> infinity 3418 <==== exfat tuna S |
| 1001707699 | | UTF-8 | . BlueCrab. UTF8 ====> infinity 3417 <==== exfat Creek |
| 1504959870 | | UTF-16 BE | ABOM UTF 16 BE ====> infinity 3427 <==== ntfs SEA Bay |
| 1504963968 | | UTF-16 | I BOM UTF 16 LE ====> infinity 3426 <==== ntfs Island T |
| 1504967862 | | UTF-8 | eek. Island! UTF8 ====> infinity 3425 <==== ntfs bass ba |
| 1666420128 | | UTF-16 BE | . BOM UTF 16 BE ====> infinity 3431 <==== ntfs bass Sq |
| 1666424220 | | UTF-16 | . BOM UTF 16 LE ====> infinity 3430 <==== ntfs Ocean |
| 1666428101 | | UTF-8 | Gill, LAKE? UTF8 ====> infinity 3429 <==== ntfs BlueGil |



Partitioning style: GPT 28 Search hits

| Phys. offs. ▲ | Log. offs. | Descr. | Search hits |
|---|---|---|---|
| 8008894 | | UTF-8 | Crab RIVER ASCII ====> infinity 3348 <==== fat Broo |
| 8013184 | | UTF-16 BE | . BOM UTF 16 BE ====> infinity 3351 <==== fat King |
| 8017291 | | UTF-16 BE, unali... | ! BOM UTF 16 LE ====> infinity 3350 <==== fat ponc |
| 8021159 | | UTF-8 | AKE. Creek UTF8 ====> infinity 3349 <==== fat King |
| 8983730 | | UTF-8 | tuna Creek ASCII ====> infinity 3344 <==== fat HARI |
| 8988026 | | UTF-16 BE | a BOM UTF 16 BE ====> infinity 3347 <==== fat Squic |
| 8992107 | | UTF-16 BE, unali... | R BOM UTF 16 LE ====> infinity 3346 <==== fat Blue |
| 8996023 | | UTF-8 | bass bass. UTF8 ====> infinity 3345 <==== fat SEA? |
| 500188330 | | UTF-8 | AKE Brook ASCII ====> infinity 3368 <==== unalloc |
| 500192640 | | UTF-16 BE | ‹ BOM UTF 16 BE ====> infinity 3371 <==== unalloc |
| 500196739 | | UTF-16 BE, unali... | I BOM UTF 16 LE ====> infinity 3370 <==== unalloc |
| 500200620 | | UTF-8 | Bay. pond. UTF8 ====> infinity 3369 <==== unalloc |
| 1000650936 | | UTF-8 | . KingCrab ASCII ====> infinity 3356 <==== exfat po |
| 1000655216 | | UTF-16 BE | yBOM UTF 16 BE ====> infinity 3359 <==== exfat Kir |
| 1000659345 | | UTF-16 BE, unali... | › BOM UTF 16 LE ====> infinity 3358 <==== exfat SH |
| 1000663232 | | UTF-8 | and Trout. UTF8 ====> infinity 3357 <==== exfat Kir |
| 1001420988 | | UTF-8 | bass, SEA. ASCII ====> infinity 3352 <==== exfat Bay |
| 1001425280 | | UTF-16 BE | R BOM UTF 16 BE ====> infinity 3355 <==== exfat RIV |
| 1001429395 | | UTF-16 BE, unali... | a BOM UTF 16 LE ====> infinity 3354 <==== exfat Blu |
| 1001433265 | | UTF-8 | Crab. Carp, UTF8 ====> infinity 3353 <==== exfat RIV |
| 1500163250 | | UTF-8 | Squid Carp ASCII ====> infinity 3364 <==== ntfs Tro |
| 1500167544 | | UTF-16 BE | A BOM UTF 16 BE ====> infinity 3367 <==== ntfs Cre |
| 1500171637 | | UTF-16 BE, unali... | . BOM UTF 16 LE ====> infinity 3366 <==== ntfs Bro |
| 1500175540 | | UTF-8 | ill. SHARK. UTF8 ====> infinity 3365 <==== ntfs RIVE |
| 1504677035 | | UTF-8 | ean. bass? ASCII ====> infinity 3360 <==== ntfs HAF |
| 1504681354 | | UTF-16 BE | ? BOM UTF 16 BE ====> infinity 3363 <==== ntfs Car |
| 1504685431 | | UTF-16 BE, unali... | › BOM UTF 16 LE ====> infinity 3362 <==== ntfs SEA |
| 1504689331 | | UTF-8 | : BlueCrab. UTF8 ====> infinity 3361 <==== ntfs Bay |

# New Topic: Unicode Test Strings

- Each string appears multiple (21) times.
- Each string appears in an active file and a deleted file.
- Each string appears in 3 formatted partitions: FAT, ExFAT, NTFS
- Each string appears in 3 UNICODE encodings: UTF-8, 16BE, 16LE
- Each encoding appears once in unallocated space. Total: 2x3x3+3

| String Class | Strings |
|---|---|
| Kanji: Japanese & Chinese | 東□ Tokyo (Japanese)<br>□ □ China (Simplified Chinese) |
| Hangul: Korean | 서울 Seoul (Korean) |
| Kana: Hiragana & Katakana | □ □ □ Su ba ru (Katakana)<br>□ □ □ □ Mi tsu bi shi (Hiragana) |
| Cyrillic: Russian | Сибирь Siberia (Russian) |
| Latin: French & German | Garçon Boy (French)<br>Schönheit Beauty (German) |
| RTL: Arabic | الكسكس The Couscous (Arabic) |

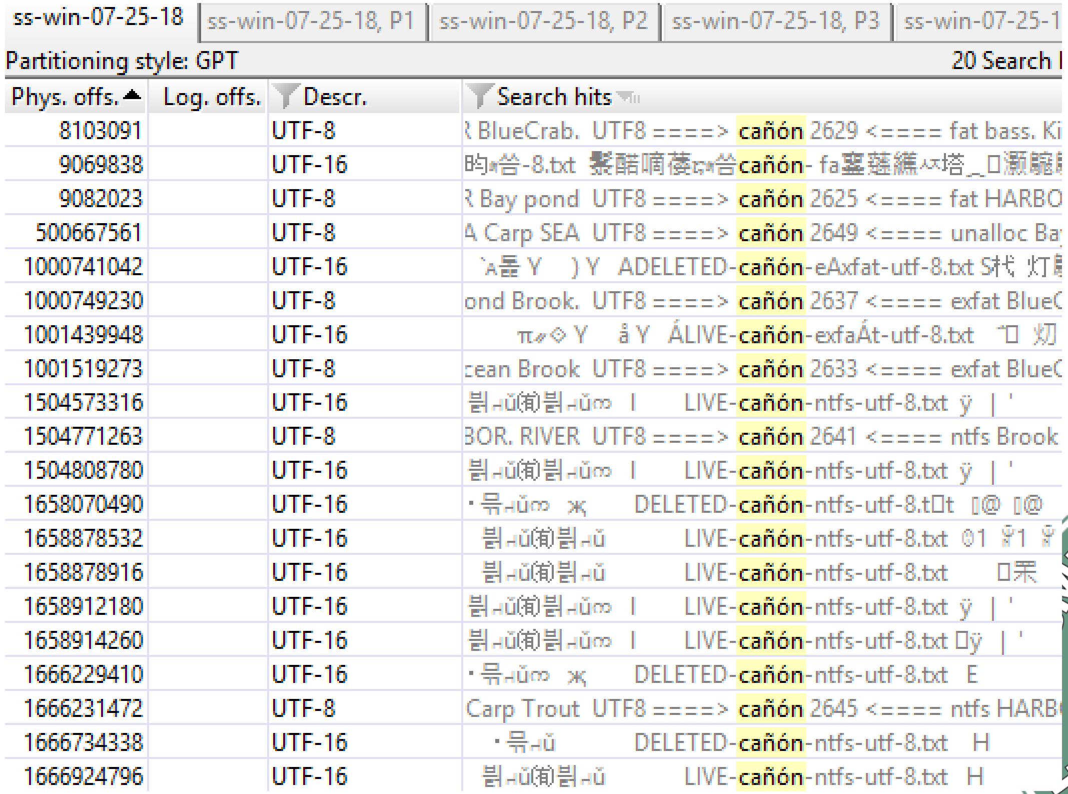
# New Topic: Meta-Data on Windows (FAT, ExFAT & NTFS)

- A target string might be a substring of a file name. What happens then?
- Let's try "cañón" (Expect 7 hits

  + some meta-data hits)

- We got the 7 and then some meta-data

| File System | Meta Data Count |
|---|---|
| FAT | 1 |
| ExFAT | 2 |
| NTFS | 10 |



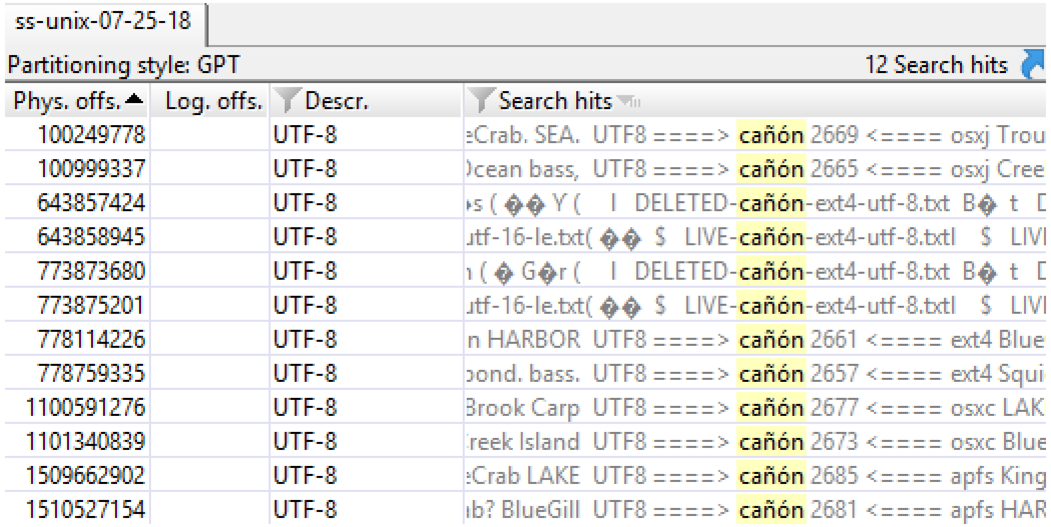
ss-win-07-25-18 | ss-win-07-25-18, P1 | ss-win-07-25-18, P2 | ss-win-07-25-18, P3 | ss-win-07-25-1

Partitioning style: GPT 20 Search I

| Phys. offs. | Log. offs. | Descr. | Search hits |
|---|---|---|---|
| 8103091 | | UTF-8 | ₹ BlueCrab. UTF8 ====> cañón 2629 <==== fat bass. Ki |
| 9069838 | | UTF-16 | 昀ﾛ삽-8.txt 褩醋嘀蘐ﾛ삽cañón- fa塞莚縑ㅅ塔_口灝廳 |
| 9082023 | | UTF-8 | ₹ Bay pond UTF8 ====> cañón 2625 <==== fat HARBO |
| 500667561 | | UTF-8 | A Carp SEA UTF8 ====> cañón 2649 <==== unalloc Ba |
| 1000741042 | | UTF-16 | `ᴀ틆 Y ) Y ADELETED-cañón-eAxfat-utf-8.txt S杙 灯 |
| 1000749230 | | UTF-8 | ond Brook. UTF8 ====> cañón 2637 <==== exfat BlueC |
| 1001439948 | | UTF-16 | π⫽◇ Y å Y ÁLIVE-cañón-exfaÁt-utf-8.txt ⁻口 灱 |
| 1001519273 | | UTF-8 | cean Brook UTF8 ====> cañón 2633 <==== exfat BlueC |
| 1504573316 | | UTF-16 | 븸ﾠů㈲븸ﾠů∞ I LIVE-cañón-ntfs-utf-8.txt ÿ \| ' |
| 1504771263 | | UTF-8 | BOR. RIVER UTF8 ====> cañón 2641 <==== ntfs Brook |
| 1504808780 | | UTF-16 | 븸ﾠů㈲븸ﾠů∞ I LIVE-cañón-ntfs-utf-8.txt ÿ \| ' |
| 1658070490 | | UTF-16 | ·뭉ﾠů∞ ж DELETED-cañón-ntfs-utf-8.t口t ▯@ ▯@ |
| 1658878532 | | UTF-16 | 븸ﾠů㈲븸ﾠů LIVE-cañón-ntfs-utf-8.txt ⓪1 ÿ1 ÿ |
| 1658878916 | | UTF-16 | 븸ﾠů㈲븸ﾠů LIVE-cañón-ntfs-utf-8.txt 口罘 |
| 1658912180 | | UTF-16 | 븸ﾠů㈲븸ﾠů∞ I LIVE-cañón-ntfs-utf-8.txt ÿ \| ' |
| 1658914260 | | UTF-16 | 븸ﾠů㈲븸ﾠů∞ I LIVE-cañón-ntfs-utf-8.txt 口ÿ \| ' |
| 1666229410 | | UTF-16 | ·뭉ﾠů∞ ж DELETED-cañón-ntfs-utf-8.txt E |
| 1666231472 | | UTF-8 | Carp Trout UTF8 ====> cañón 2645 <==== ntfs HARB |
| 1666734338 | | UTF-16 | ·뭉ﾠů DELETED-cañón-ntfs-utf-8.txt H |
| 1666924796 | | UTF-16 | 븸ﾠů㈲븸ﾠů LIVE-cañón-ntfs-utf-8.txt H |

# Meta-Data on Unix – ext4, OSXJ, OSXC & APFS

- Let’s see what we get on Unix-like file systems . . .



ss-unix-07-25-18

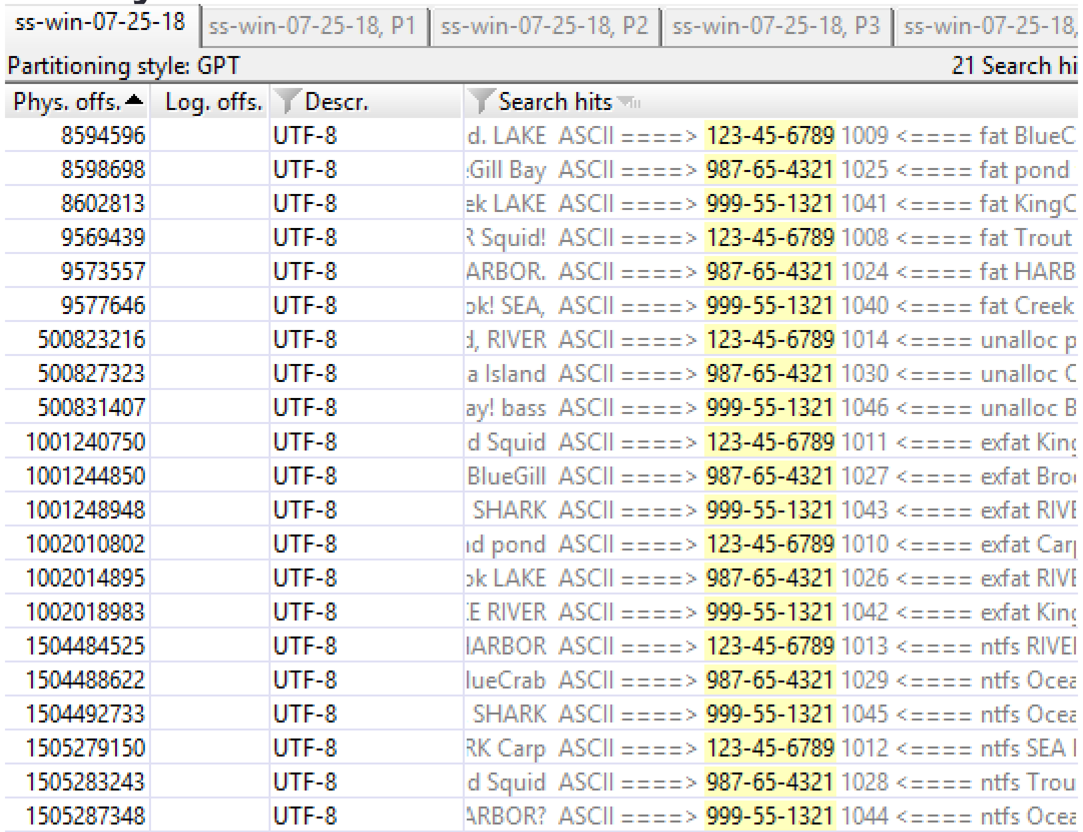
Partitioning style: GPT    12 Search hits

| Phys. offs. | Log. offs. | Descr. | Search hits |
|---|---|---|---|
| 100249778 | | UTF-8 | :Crab. SEA. UTF8 ====> cañón 2669 <==== osxj Trou |
| 100999337 | | UTF-8 | )cean bass, UTF8 ====> cañón 2665 <==== osxj Cree |
| 643857424 | | UTF-8 | ›s ( �� Y ( I DELETED-cañón-ext4-utf-8.txt B� t [ |
| 643858945 | | UTF-8 | ıtf-16-le.txt( �� $ LIVE-cañón-ext4-utf-8.txtl $ LIVI |
| 773873680 | | UTF-8 | ı ( � G�r ( I DELETED-cañón-ext4-utf-8.txt B� t [ |
| 773875201 | | UTF-8 | ıtf-16-le.txt( �� $ LIVE-cañón-ext4-utf-8.txtl $ LIVI |
| 778114226 | | UTF-8 | n HARBOR UTF8 ====> cañón 2661 <==== ext4 Blue |
| 778759335 | | UTF-8 | ɔond. bass. UTF8 ====> cañón 2657 <==== ext4 Squi |
| 1100591276 | | UTF-8 | 3rook Carp UTF8 ====> cañón 2677 <==== osxc LAK |
| 1101340839 | | UTF-8 | :reek Island UTF8 ====> cañón 2673 <==== osxc Blue |
| 1509662902 | | UTF-8 | :Crab LAKE UTF8 ====> cañón 2685 <==== apfs King |
| 1510527154 | | UTF-8 | ıb? BlueGill UTF8 ====> cañón 2681 <==== apfs HAR |

- Nothing found for Mac file systems, but 4 hits on Linux ext4.

# New Topic: Built-in Searches

- Tools often have built-in searches for interesting items like social security numbers, phone numbers, credit cards & IP addresses
- For example, Social Security search returns:



ss-win-07-25-18 | ss-win-07-25-18, P1 | ss-win-07-25-18, P2 | ss-win-07-25-18, P3 | ss-win-07-25-18,

Partitioning style: GPT

21 Search hi

| Phys. offs. | Log. offs. | Descr. | Search hits |
|---|---|---|---|
| 8594596 | | UTF-8 | d. LAKE ASCII ====> 123-45-6789 1009 <==== fat BlueC |
| 8598698 | | UTF-8 | :Gill Bay ASCII ====> 987-65-4321 1025 <==== fat pond |
| 8602813 | | UTF-8 | ek LAKE ASCII ====> 999-55-1321 1041 <==== fat KingC |
| 9569439 | | UTF-8 | R Squid! ASCII ====> 123-45-6789 1008 <==== fat Trout |
| 9573557 | | UTF-8 | ARBOR. ASCII ====> 987-65-4321 1024 <==== fat HARB |
| 9577646 | | UTF-8 | ok! SEA, ASCII ====> 999-55-1321 1040 <==== fat Creek |
| 500823216 | | UTF-8 | d, RIVER ASCII ====> 123-45-6789 1014 <==== unalloc p |
| 500827323 | | UTF-8 | a Island ASCII ====> 987-65-4321 1030 <==== unalloc C |
| 500831407 | | UTF-8 | ay! bass ASCII ====> 999-55-1321 1046 <==== unalloc B |
| 1001240750 | | UTF-8 | d Squid ASCII ====> 123-45-6789 1011 <==== exfat King |
| 1001244850 | | UTF-8 | BlueGill ASCII ====> 987-65-4321 1027 <==== exfat Bro |
| 1001248948 | | UTF-8 | SHARK ASCII ====> 999-55-1321 1043 <==== exfat RIVE |
| 1002010802 | | UTF-8 | ıd pond ASCII ====> 123-45-6789 1010 <==== exfat Carp |
| 1002014895 | | UTF-8 | ok LAKE ASCII ====> 987-65-4321 1026 <==== exfat RIVE |
| 1002018983 | | UTF-8 | E RIVER ASCII ====> 999-55-1321 1042 <==== exfat King |
| 1504484525 | | UTF-8 | IARBOR ASCII ====> 123-45-6789 1013 <==== ntfs RIVE |
| 1504488622 | | UTF-8 | lueCrab ASCII ====> 987-65-4321 1029 <==== ntfs Ocea |
| 1504492733 | | UTF-8 | SHARK ASCII ====> 999-55-1321 1045 <==== ntfs Ocea |
| 1505279150 | | UTF-8 | RK Carp ASCII ====> 123-45-6789 1012 <==== ntfs SEA I |
| 1505283243 | | UTF-8 | d Squid ASCII ====> 987-65-4321 1028 <==== ntfs Trou |
| 1505287348 | | UTF-8 | ARBOR? ASCII ====> 999-55-1321 1044 <==== ntfs Ocea |

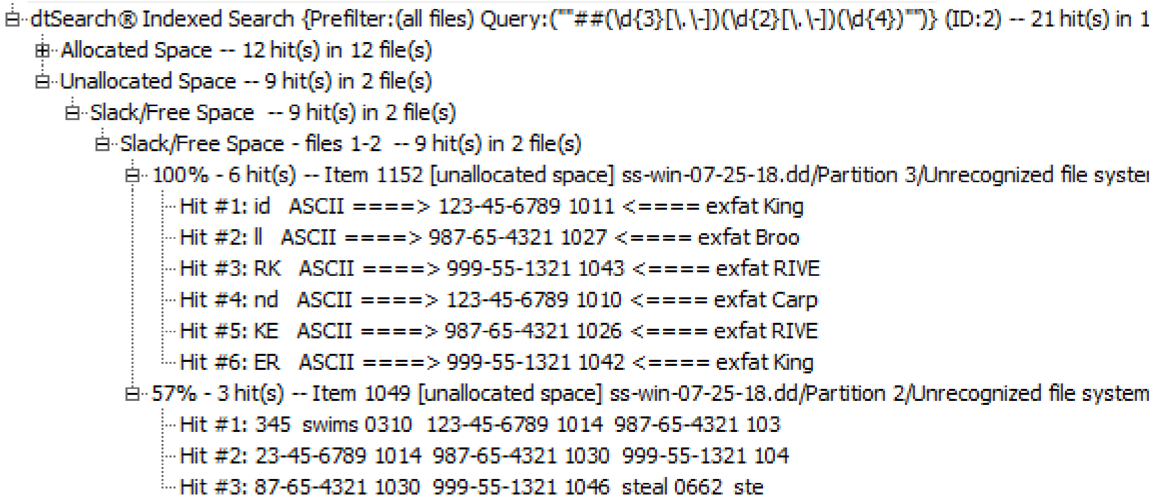
- 3 partitions x 3 strings 2 times per partition + 3 in unallocated = expect 21 hits

# Let’s try “Find SS#” button in Another Tool

Try indexed search

Actual results:

- 12 hits in allocated space +
- 9 hits in unallocated space =
- Total of 21 hits



```
⊟ dtSearch® Indexed Search {Prefilter:(all files) Query:(""##(\d{3}[\.\-])(\d{2}[\.\-])(\d{4})"")} (ID:2) -- 21 hit(s) in 1
  ⊞ Allocated Space -- 12 hit(s) in 12 file(s)
  ⊟ Unallocated Space -- 9 hit(s) in 2 file(s)
     ⊟ Slack/Free Space  -- 9 hit(s) in 2 file(s)
        ⊟ Slack/Free Space - files 1-2  -- 9 hit(s) in 2 file(s)
           ⊟ 100% - 6 hit(s) -- Item 1152 [unallocated space] ss-win-07-25-18.dd/Partition 3/Unrecognized file syste
              Hit #1: id   ASCII ====> 123-45-6789 1011 <==== exfat King
              Hit #2: ll   ASCII ====> 987-65-4321 1027 <==== exfat Broo
              Hit #3: RK   ASCII ====> 999-55-1321 1043 <==== exfat RIVE
              Hit #4: nd   ASCII ====> 123-45-6789 1010 <==== exfat Carp
              Hit #5: KE   ASCII ====> 987-65-4321 1026 <==== exfat RIVE
              Hit #6: ER   ASCII ====> 999-55-1321 1042 <==== exfat King
           ⊟ 57% - 3 hit(s) -- Item 1049 [unallocated space] ss-win-07-25-18.dd/Partition 2/Unrecognized file system
              Hit #1: 345  swims 0310  123-45-6789 1014  987-65-4321 103
              Hit #2: 23-45-6789 1014  987-65-4321 1030  999-55-1321 104
              Hit #3: 87-65-4321 1030  999-55-1321 1046  steal 0662  ste
```

- Total is correct, but Wait, wait. Shouldn’t it be : 18 allocated + 3 unallocated?

- This tool does not support ExFAT (or APFS)
- Also the presentation of the hits from partition 2 is a little unclear

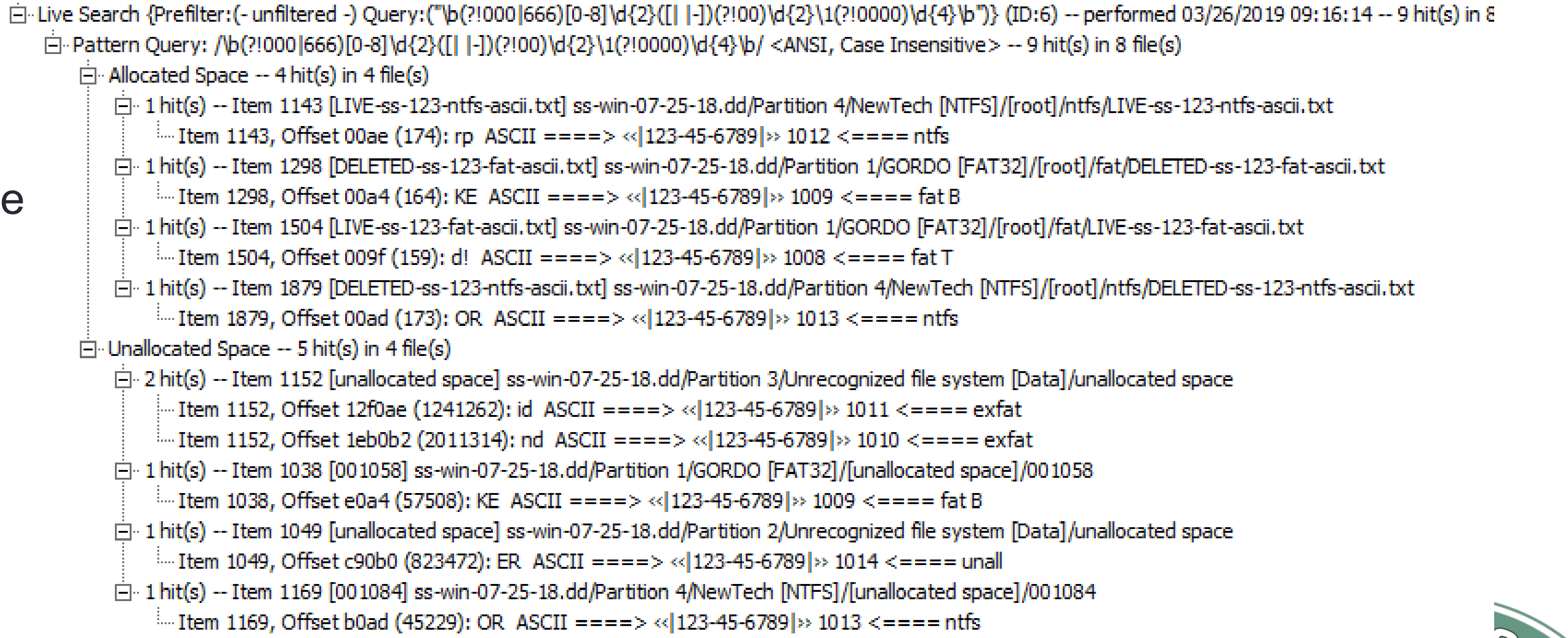
# More Social Security

Search results for the tool doing a LIVE search:

- 4 hits in allocated space
- 5 hits in unallocated space
- 9 hits total, 2 instances reported twice.

- Where did the other two target strings go?
- 987-65-4321 & 999-55-1321

- Not valid SS#, so not reported, however . . .
- They could be valid IRS taxpayer ID numbers issued by IRS to people without SS#s

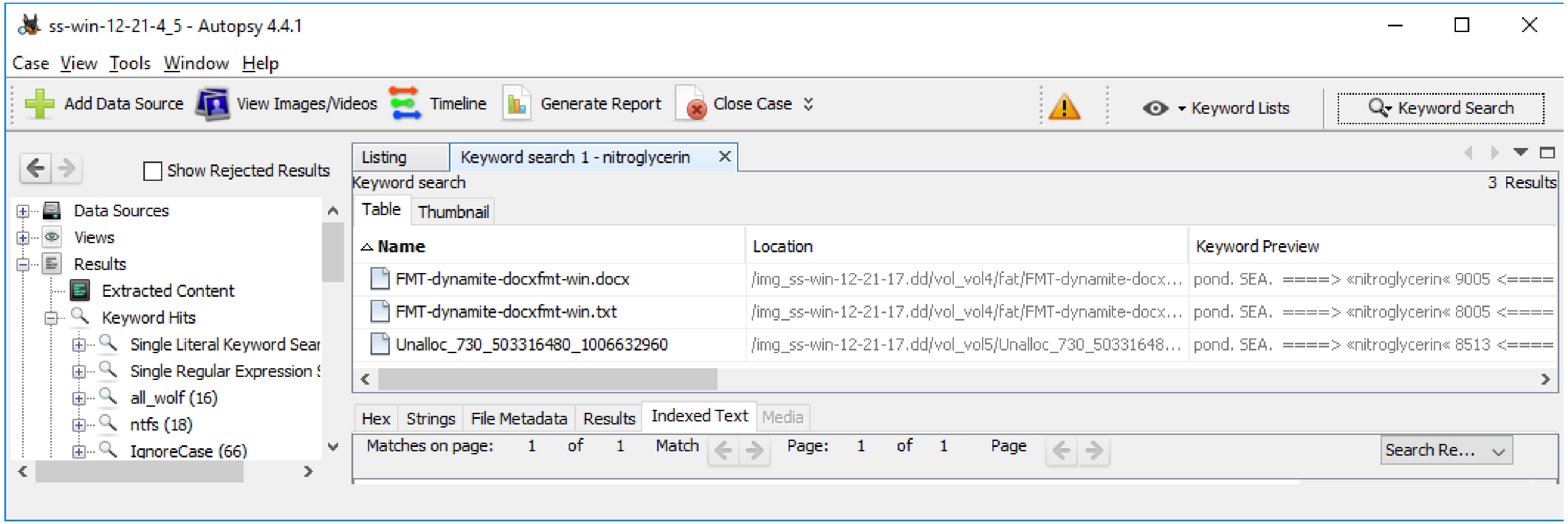


```
⊟ Live Search {Prefilter:(- unfiltered -) Query:("\b(?!000|666)[0-8]\d{2}([| |-])(?!00)\d{2}\1(?!0000)\d{4}\b")} (ID:6) -- performed 03/26/2019 09:16:14 -- 9 hit(s) in 8
  ⊟ Pattern Query: /\b(?!000|666)[0-8]\d{2}([| |-])(?!00)\d{2}\1(?!0000)\d{4}\b/ <ANSI, Case Insensitive> -- 9 hit(s) in 8 file(s)
    ⊟ Allocated Space -- 4 hit(s) in 4 file(s)
      ⊟ 1 hit(s) -- Item 1143 [LIVE-ss-123-ntfs-ascii.txt] ss-win-07-25-18.dd/Partition 4/NewTech [NTFS]/[root]/ntfs/LIVE-ss-123-ntfs-ascii.txt
          Item 1143, Offset 00ae (174): rp  ASCII ====> «|123-45-6789|» 1012 <==== ntfs
      ⊟ 1 hit(s) -- Item 1298 [DELETED-ss-123-fat-ascii.txt] ss-win-07-25-18.dd/Partition 1/GORDO [FAT32]/[root]/fat/DELETED-ss-123-fat-ascii.txt
          Item 1298, Offset 00a4 (164): KE  ASCII ====> «|123-45-6789|» 1009 <==== fat B
      ⊟ 1 hit(s) -- Item 1504 [LIVE-ss-123-fat-ascii.txt] ss-win-07-25-18.dd/Partition 1/GORDO [FAT32]/[root]/fat/LIVE-ss-123-fat-ascii.txt
          Item 1504, Offset 009f (159): d!  ASCII ====> «|123-45-6789|» 1008 <==== fat T
      ⊟ 1 hit(s) -- Item 1879 [DELETED-ss-123-ntfs-ascii.txt] ss-win-07-25-18.dd/Partition 4/NewTech [NTFS]/[root]/ntfs/DELETED-ss-123-ntfs-ascii.txt
          Item 1879, Offset 00ad (173): OR  ASCII ====> «|123-45-6789|» 1013 <==== ntfs
    ⊟ Unallocated Space -- 5 hit(s) in 4 file(s)
      ⊟ 2 hit(s) -- Item 1152 [unallocated space] ss-win-07-25-18.dd/Partition 3/Unrecognized file system [Data]/unallocated space
          Item 1152, Offset 12f0ae (1241262): id  ASCII ====> «|123-45-6789|» 1011 <==== exfat
          Item 1152, Offset 1eb0b2 (2011314): nd  ASCII ====> «|123-45-6789|» 1010 <==== exfat
      ⊟ 1 hit(s) -- Item 1038 [001058] ss-win-07-25-18.dd/Partition 1/GORDO [FAT32]/[unallocated space]/001058
          Item 1038, Offset e0a4 (57508): KE  ASCII ====> «|123-45-6789|» 1009 <==== fat B
      ⊟ 1 hit(s) -- Item 1049 [unallocated space] ss-win-07-25-18.dd/Partition 2/Unrecognized file system [Data]/unallocated space
          Item 1049, Offset c90b0 (823472): ER  ASCII ====> «|123-45-6789|» 1014 <==== unall
      ⊟ 1 hit(s) -- Item 1169 [001084] ss-win-07-25-18.dd/Partition 4/NewTech [NTFS]/[unallocated space]/001084
          Item 1169, Offset b0ad (45229): OR  ASCII ====> «|123-45-6789|» 1013 <==== ntfs
```

# New Topic: Searching Formatted Text – MS Word, HTML

- Each string appears four times
  - Plain Text in FAT partition
  - Formatted Text in FAT partition
  - Plain Text in unallocated space
  - Formatted Text in unallocated space
- Formatting schemes used
  - MS Word .doc & .docx
  - HTML
- Part of the string is formatted bold and underlined
  - **Cross**Bow HTML <u><b>Cross</b></u>Bow
  - **Nitro**glycerin DOCX
  - **Shot**gun DOC

# Formatted Text Searches – Find **nitro**glycerin



Keyword search 3 Results

| Name | Location | Keyword Preview |
|---|---|---|
| FMT-dynamite-docxfmt-win.docx | /img_ss-win-12-21-17.dd/vol_vol4/fat/FMT-dynamite-docx... | pond. SEA. ====> «nitroglycerin« 9005 <==== |
| FMT-dynamite-docxfmt-win.txt | /img_ss-win-12-21-17.dd/vol_vol4/fat/FMT-dynamite-docx... | pond. SEA. ====> «nitroglycerin« 8005 <==== |
| Unalloc_730_503316480_1006632960 | /img_ss-win-12-21-17.dd/vol_vol5/Unalloc_730_50331648... | pond. SEA. ====> «nitroglycerin« 8513 <==== |

The string nitroglycerin appears 4 times:

- Text in the FAT Partition (8005) and in unallocated space (8513)
- Formatted text in a docx file: **nitro**glycerin (9005 in FAT and 9513 in unallocated space.
- This tool found formatted text in FAT, but only some tools found string in unallocated space.
- Tried other tools with slightly different results

# Unexpected Results

If a tool returns an unexpected result for a test case . . .

- Tool is not designed to do what the user expects (it’s a feature)
- Tool is not implemented to correctly do what the designer intended (It’s a bug)
- Tool is not configured to do the exact task the user wants (User error, read the documentation again)

# Two Things Learned Making Test Data

MFT: *fixups* and the *Update Sequence Array.*

- I noticed my string documentation program sometimes missed strings that I knew were in the test image, but forensic string search tools could find the strings that my program missed.

Copy/Paste from PDF may not do what you expect.

- One day I noticed that none of the tools found Arabic text anymore. I had been copying strings from a text document.
- I changed the document to PDF and was copying/pasting strings from the PDF version.
- Arabic + PDF = Unexpected. The string renders correctly in the search tool, but the byte codes copied are not Unicode.

# Some Observed Tool Behaviors

- Most tools could parse FAT, ExFAT, NTFS, ext4, journaled OSX and case-sensitive OSX partitions. Sometimes ExFAT or APFS not supported
- Usually found ASCII, UTF-8 & UTF-16, but sometimes failed to find UTF-16 strings
- Sometimes indexed search and live search have differences.
- Sometimes UTF-16BE reported as UTF-16LE and vice versa
- Usually 1-1 reporting of each hit to location, but sometimes reported as multiple hits
- One older tool version reported a corrupted name for some ExFAT files containing a hit
- One tool (old version) fails to render Korean UNICODE string correctly
- Some tools fail to ignore embedded HTML tags
- Most tools failed to recognize and decode docx file in unallocated space

# Software Testing Gets You . . .

- Tool testing catches specific errors thus **increasing your confidence in the tool**
- Testing NEVER can PROVE a program is always correct.
- Software Testing is asking questions to see how the tested tool reacts to various inputs
- If software gives an unexpected result it usually is triggered by a specific condition
- Better understanding comes from trying more conditions . . .
  - More diversity of questions
  - More detailed questions
- Testing documents tool behaviors that you need to be aware of

# Getting Federated Testing with String Search https://cftt.nist.gov/federated-testing.html

Sharing CFTT Test Methods, Tools & Forensic Lab Test Reports

- Helps a forensic lab test tools easily and with high quality
- For string searching CFTT provides test images with known content and a list of test cases designed to test specific features.
  1. Tester can select relevant test cases from a list of test cases
  2. Each case is run by first setting tool options and then searching for a string
  3. Federated testing tool records search results
  4. Tool to generate a skeleton test report that can then be finished in the style favored by the laboratory.
- The test reports can be shared with other labs

# Contact Information

Jim Lyle
jlyle@nist.gov
cftt.nist.gov

E-Mail federatedtesting-request@nist.gov with the word “subscribe” (without quotes) in the subject line to subscribe to the federatedtesting@nist.gov mailing list. Federatedtesting@nist.gov is a low volume mailing list for distributing updates on the Federated Testing project and the Federated Testing Forensic Tool Testing Environment (e.g., new releases/versions and capabilities).

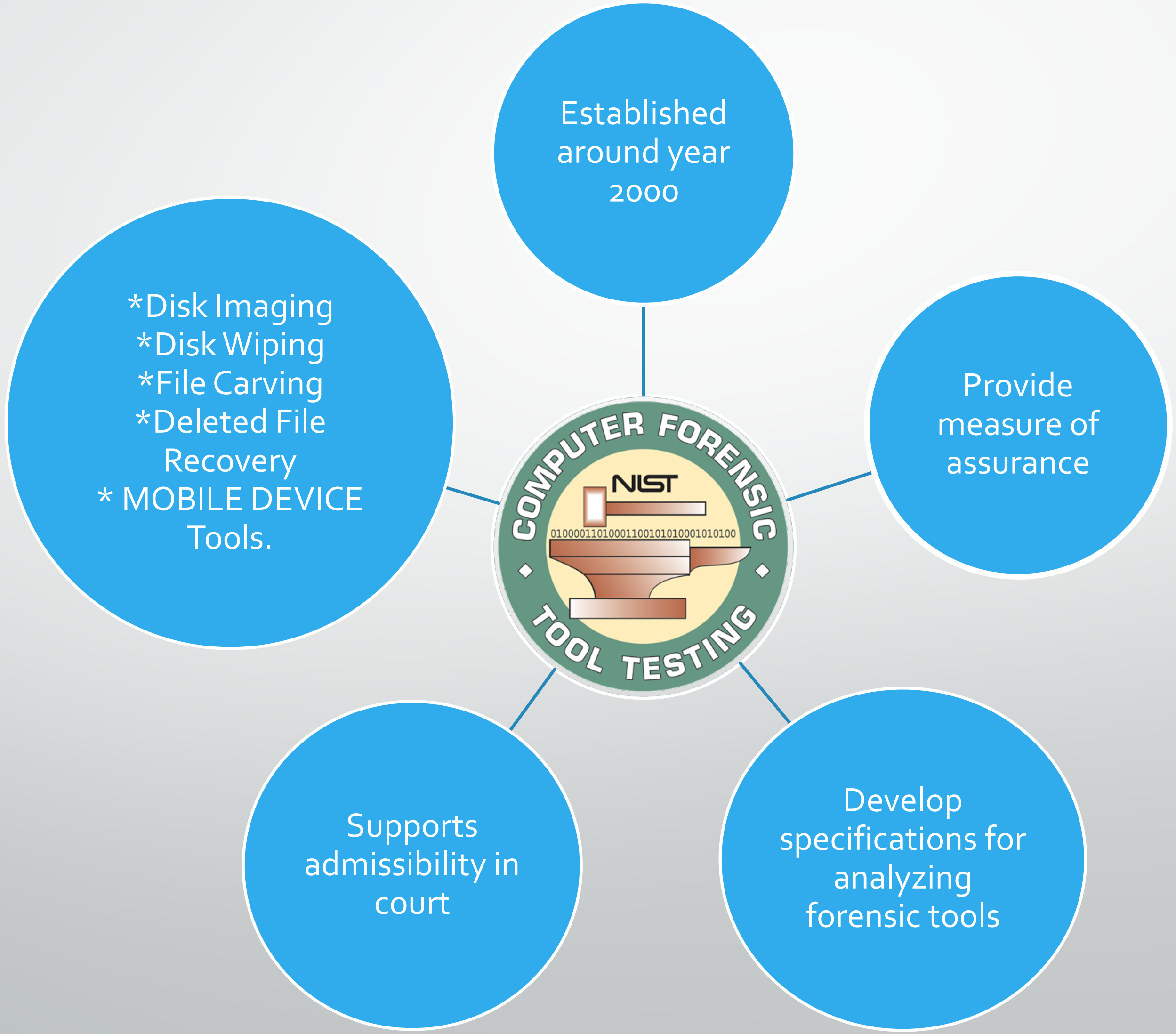
# What's New About Mobile? SQLite, SQLite Recovery and a new Federated Testing Tool

Jenise Reyes-Rodriguez
Software and Systems Division, ITL

Forensics@NIST – November 6, 2020

NIST
National Institute of
Standards and Technology
U.S. Department of Commerce

# Computer Forensic Tool Testing



Established around year 2000

Provide measure of assurance

Develop specifications for analyzing forensic tools

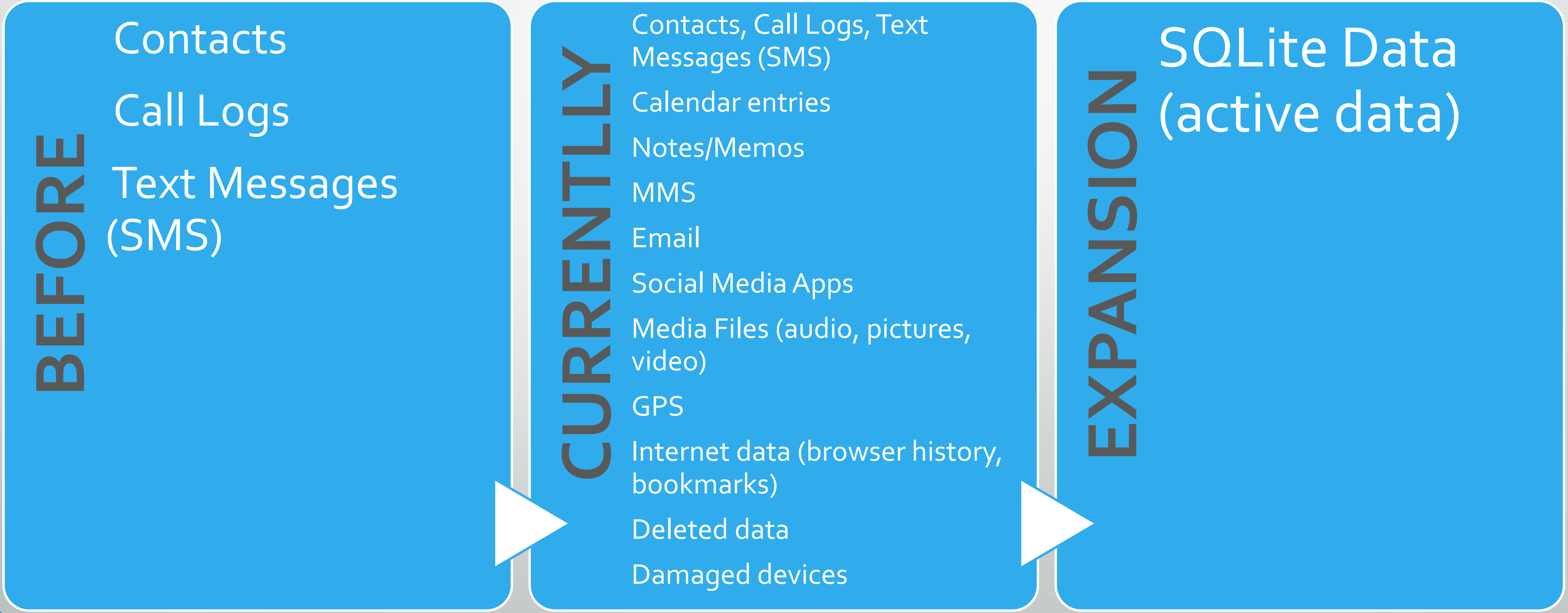
Supports admissibility in court

*Disk Imaging
*Disk Wiping
*File Carving
*Deleted File Recovery
* MOBILE DEVICE Tools.

# New Developments within CFTT

- ❑ Expansion of our mobile forensic tool testing and specification
  - To better address SQLite databases (active data)
- ❑ New specification and testing program for SQLite deleted/modified data recovery
- ❑ Third is a new version of Federated Testing for mobile
  - Can be run on Windows platforms

# Mobile Forensics Tool Testing



BEFORE

- Contacts
- Call Logs
- Text Messages (SMS)

CURRENTLLY

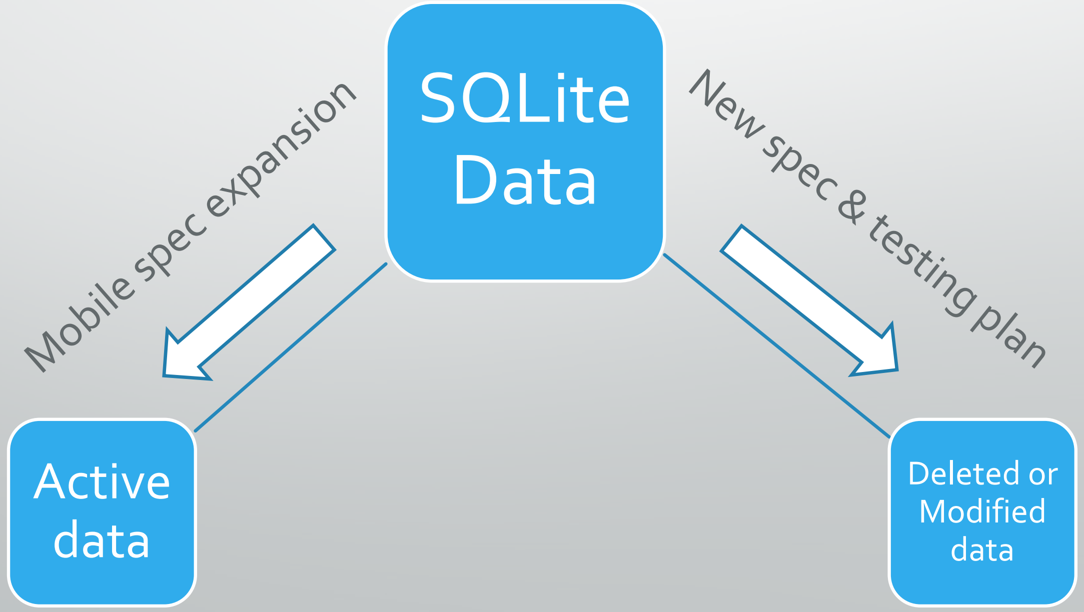
- Contacts, Call Logs, Text Messages (SMS)
- Calendar entries
- Notes/Memos
- MMS
- Email
- Social Media Apps
- Media Files (audio, pictures, video)
- GPS
- Internet data (browser history, bookmarks)
- Deleted data
- Damaged devices

EXPANSION

- SQLite Data (active data)

# SQLite Data

- Why expanding the mobile specification to include SQLite data?
  - SQLite - data format used to store data on devices
  - Tools have incorporated SQLite data viewers into their tools
  - Differences on how tools are reporting SQLite data



# SQLite & SQLite Deleted Data Recovery

- Why separate active and deleted/modified SQLite data?
  - Complexity
  - Incorporating what fit with the specification we already have
- Status: incorporated into the basic mobile forensics tool testing specification and test plan

# New Specification & Testing for SQLite Data Recovery

- ❑ Can you find data that doesn’t appear in the logical view/SQLite viewer?
- ❑ There are tools tailored to SQLite Data recovery
- ❑ Status: currently working on final draft of the Specification and Test plan. Then, create data sets and perform testing.

# You can perform testing too!

- ❑ NIST shares testing methodology through Federated Testing
- ❑ Users can test their tools in their labs

# New version of Federated Testing

- What is Federated Testing?
  - Expansion of the Computer Forensics Tool Testing (CFTT) Program
  - Provides digital forensics investigators and labs with test suites for tool testing and to support shared test reports.
  - Goal: help digital forensics investigators to test the tools that they use in their labs and to enable sharing of tool test results within the digital forensics community.
- Current version
  - Current distribution v5.0
  - Linux based
  - Distribution method ISO image that requires VM, create a virtual drive or 2 machines (1 to run ISO/CD and 2nd to perform the tests)
  - The ISO includes all modules available, not just for mobile

# New version of Federated Testing v5.1

- New version v5.1
  - Add new feature
    - Users will be able to store log files to either a USB device or desktop location of their choice
- Status:  Release expected by 1st /2nd quarter of 2021

# New version of Federated Testing for mobile

- Looking at possible new approach
  - Create a self-contained application that runs on Windows 10
  - User won't have to install any software to their machines – just click and run tests
  - NO need for a second machine
  - The Mobile module will be first to be converted into an app, but this will expand to all other modules
  - User will be able to pick a single app of interest
  - Currently have a prototype of how it would be
  - Will be a lighter version

# Contact Information

# *** www.cftt.nist.gov ***

Jenise Reyes-Rodriguez: jenise.reyes@nist.gov

Richard Ayers : richard.ayers@nist.gov

Jim Lyle: james.lyle@nist.gov

# Computer Forensic Reference Data Sets (CFReDS) Project

RICK AYERS

- CFReDS v2.5 – provides users with a centralized portal of datasets produced by various contributors providing the forensic community with the ability to quickly find datasets of interests.
  - Repository for documented sets of simulated digital evidence or datasets
  - Documented datasets ensure reliability and aid in Testing Digital Forensic tools
  - Provides the forensic community to aid in:
    - Training
      - Investigative and tool-driven
  - Tool/process evaluation
      - Simulate running  of investigative procedures, process or methodology to demonstrate compliance to a standard
  - Data exploration and reverse engineering (R&D)
      - Understanding software/application behavior

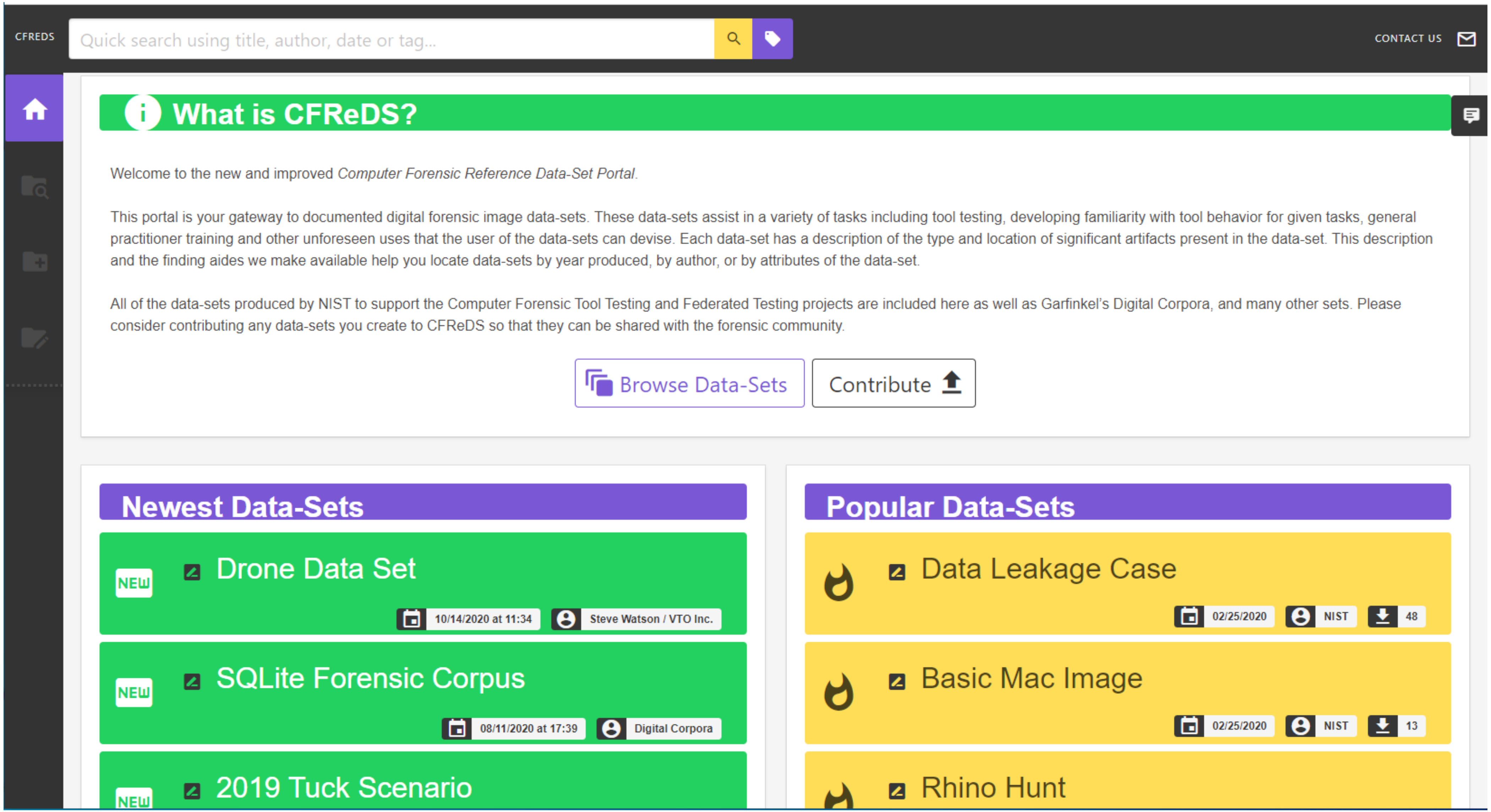
- Why are datasets hard?
  - Accurate Construction – determining the purpose
  - Time consuming
  - Sufficient Documentation
  - Realistic versus testing important attributes
  - Development for a variety of tool functionalities
  - Few standards and best practices for dataset development in digital forensics

- Dataset Categories
  - Technology
    - Camera Datasets
    - Drones
  - Functionality
    - Mobile, String Searching, File Carving, etc.
  - Scenario-based Datasets
    - Russian Tea Room
    - Data Leakage Case

- CFReDS v2.5
  - Supports large amounts of data
  - Taxonomy driven – quickly find specific datasets of Interest
  - Search bar
    - Quick Search using author, title, date or tag
  - Beta development at: https://cfreds.mehdishadid.com

- Dataset entries
  - Currently around 160 entries
    - https://www.cftt.nist.gov
    - https://digitalcorpora.org
    - https://datasets.fbreitinger.de/datasets

# CFReDS v2.5 Homepage



## What is CFReDS?

Welcome to the new and improved *Computer Forensic Reference Data-Set Portal*.

This portal is your gateway to documented digital forensic image data-sets. These data-sets assist in a variety of tasks including tool testing, developing familiarity with tool behavior for given tasks, general practitioner training and other unforeseen uses that the user of the data-sets can devise. Each data-set has a description of the type and location of significant artifacts present in the data-set. This description and the finding aides we make available help you locate data-sets by year produced, by author, or by attributes of the data-set.

All of the data-sets produced by NIST to support the Computer Forensic Tool Testing and Federated Testing projects are included here as well as Garfinkel's Digital Corpora, and many other sets. Please consider contributing any data-sets you create to CFReDS so that they can be shared with the forensic community.

## Newest Data-Sets

NEW Drone Data Set
10/14/2020 at 11:34 · Steve Watson / VTO Inc.

NEW SQLite Forensic Corpus
08/11/2020 at 17:39 · Digital Corpora

NEW 2019 Tuck Scenario

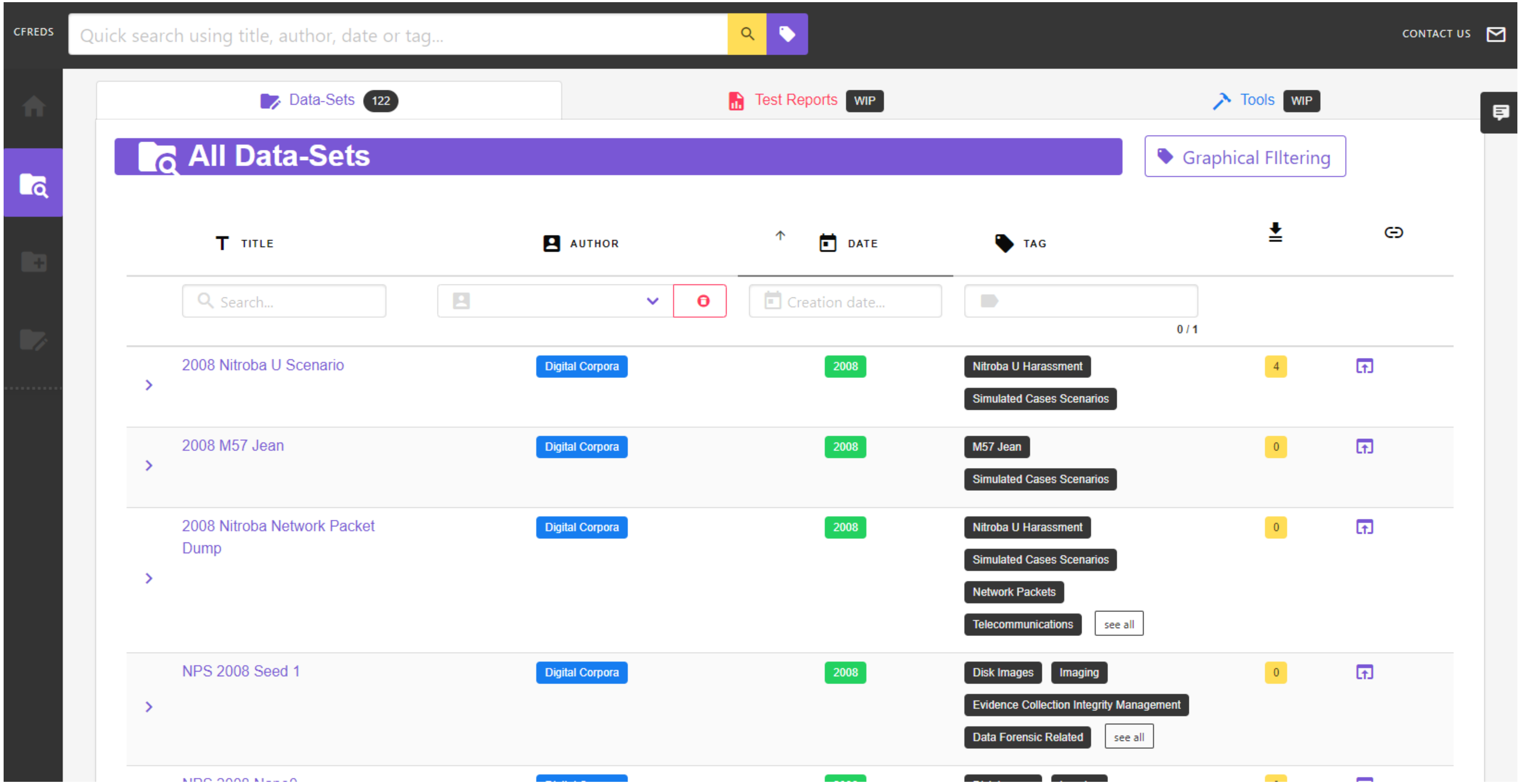
## Popular Data-Sets

Data Leakage Case
02/25/2020 · NIST · 48

Basic Mac Image
02/25/2020 · NIST · 13

Rhino Hunt

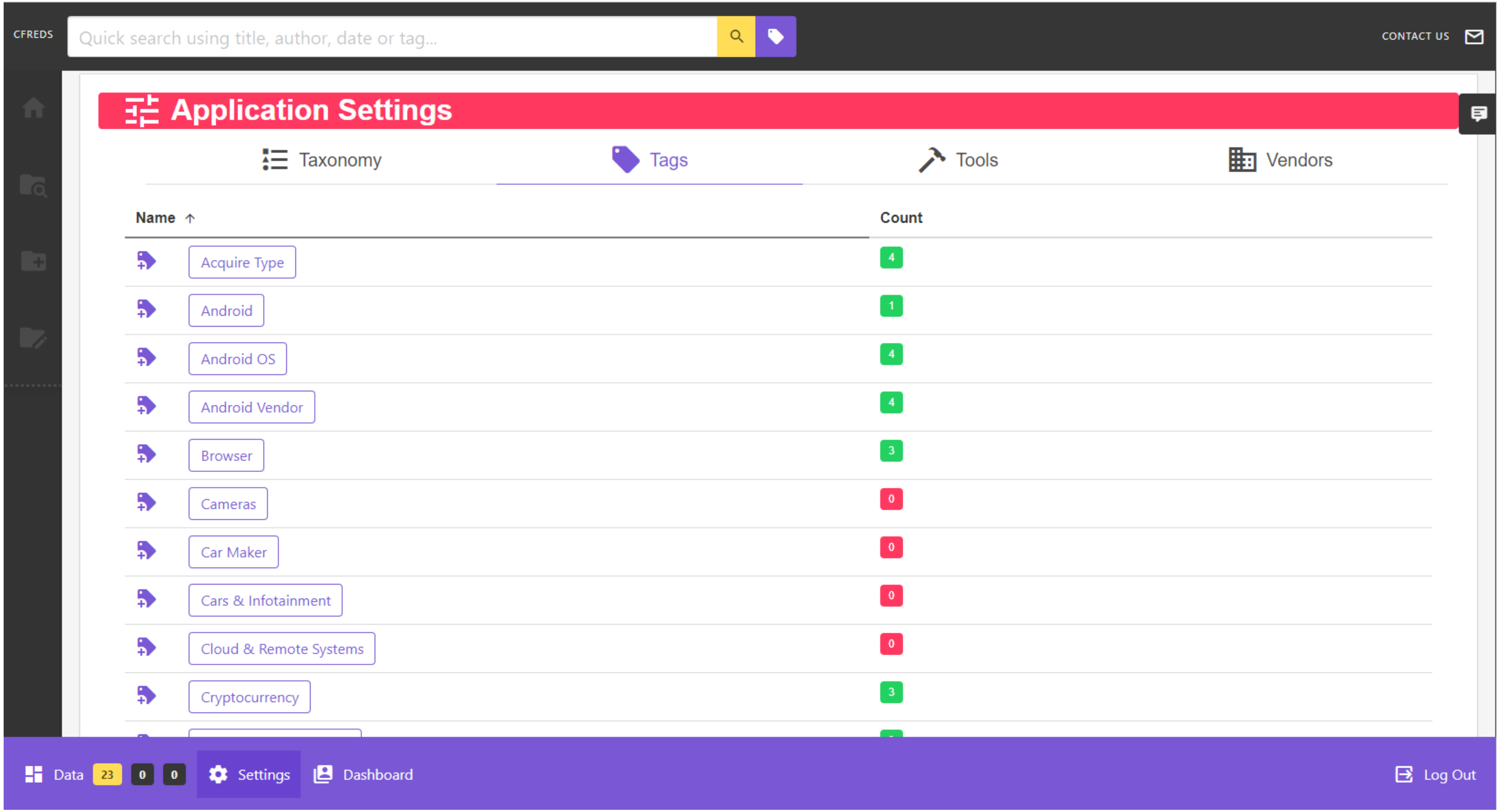
# CFReDS v2.5 Browse



## All Data-Sets

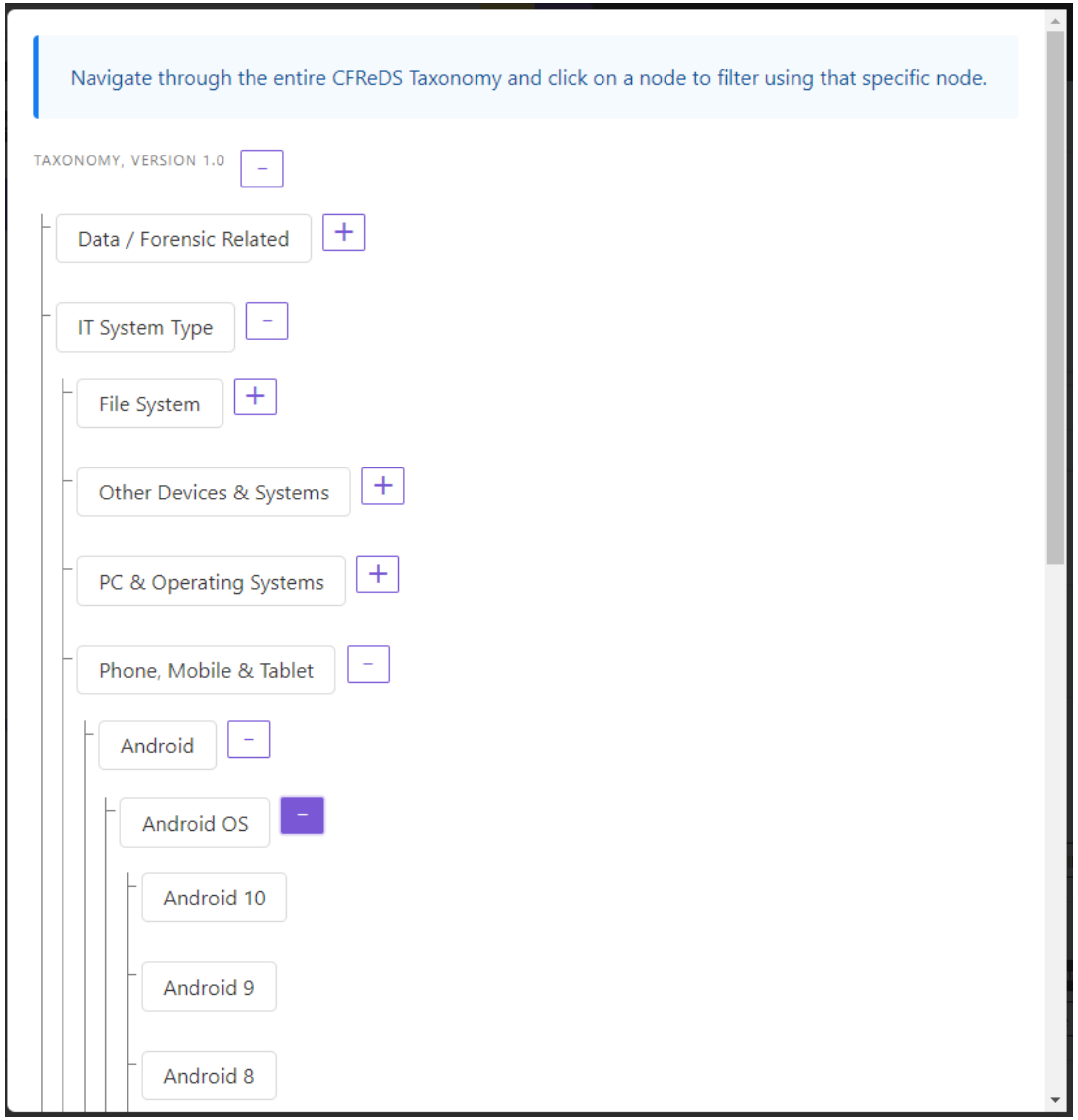
| TITLE | AUTHOR | DATE | TAG | |
|---|---|---|---|---|
| 2008 Nitroba U Scenario | Digital Corpora | 2008 | Nitroba U Harassment<br>Simulated Cases Scenarios | 4 |
| 2008 M57 Jean | Digital Corpora | 2008 | M57 Jean<br>Simulated Cases Scenarios | 0 |
| 2008 Nitroba Network Packet Dump | Digital Corpora | 2008 | Nitroba U Harassment<br>Simulated Cases Scenarios<br>Network Packets<br>Telecommunications | 0 |
| NPS 2008 Seed 1 | Digital Corpora | 2008 | Disk Images<br>Imaging<br>Evidence Collection Integrity Management<br>Data Forensic Related | 0 |

# CFReDS v2.5 Tags



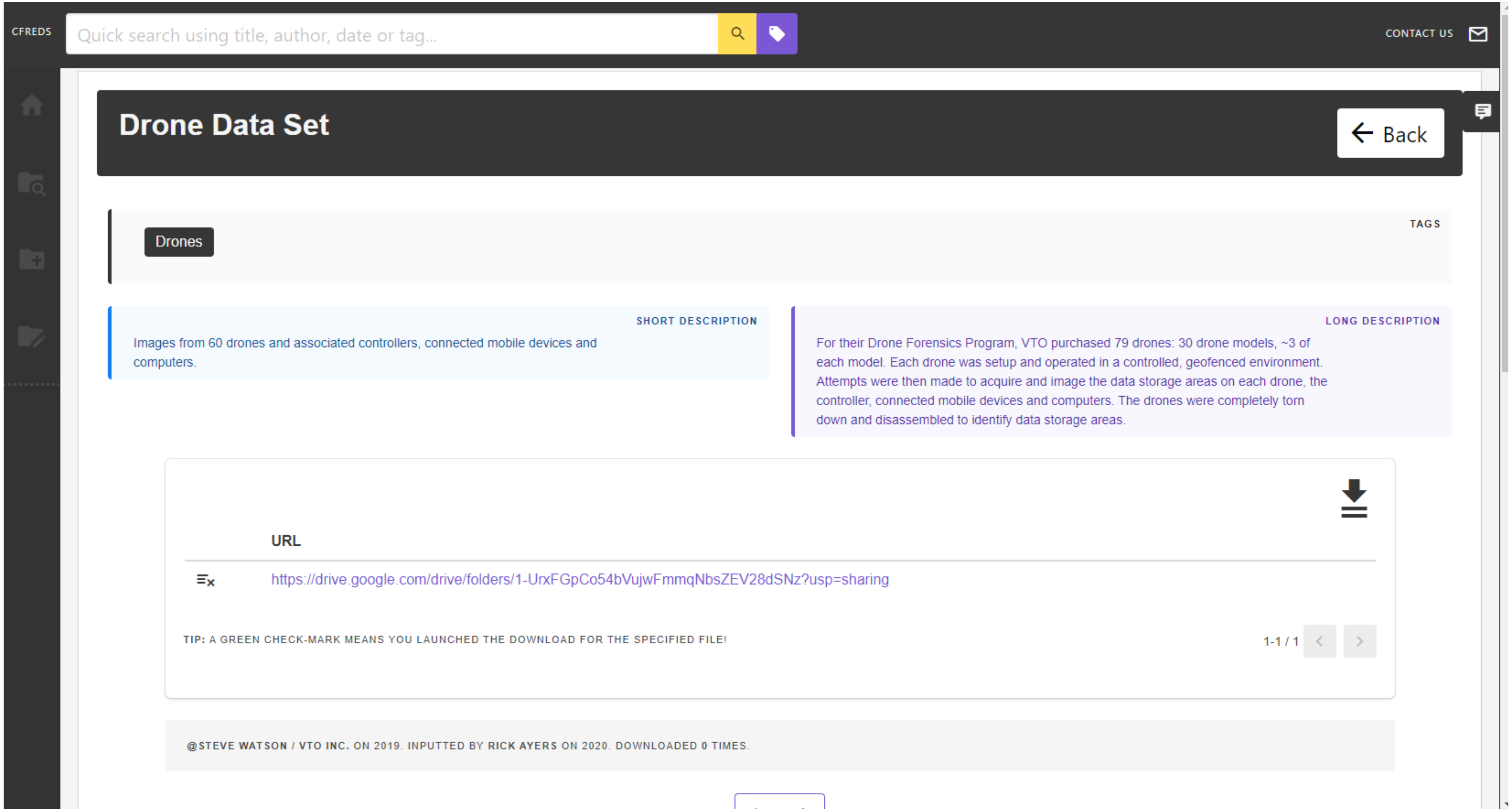
| Name ↑ | Count |
|---|---|
| Acquire Type | 4 |
| Android | 1 |
| Android OS | 4 |
| Android Vendor | 4 |
| Browser | 3 |
| Cameras | 0 |
| Car Maker | 0 |
| Cars & Infotainment | 0 |
| Cloud & Remote Systems | 0 |
| Cryptocurrency | 3 |

# CFReDS v2.5 Taxonomy



Navigate through the entire CFReDS Taxonomy and click on a node to filter using that specific node.

# CFReDS v2.5 Dataset



## Drone Data Set

TAGS

Drones

SHORT DESCRIPTION

Images from 60 drones and associated controllers, connected mobile devices and computers.

LONG DESCRIPTION

For their Drone Forensics Program, VTO purchased 79 drones: 30 drone models, ~3 of each model. Each drone was setup and operated in a controlled, geofenced environment. Attempts were then made to acquire and image the data storage areas on each drone, the controller, connected mobile devices and computers. The drones were completely torn down and disassembled to identify data storage areas.

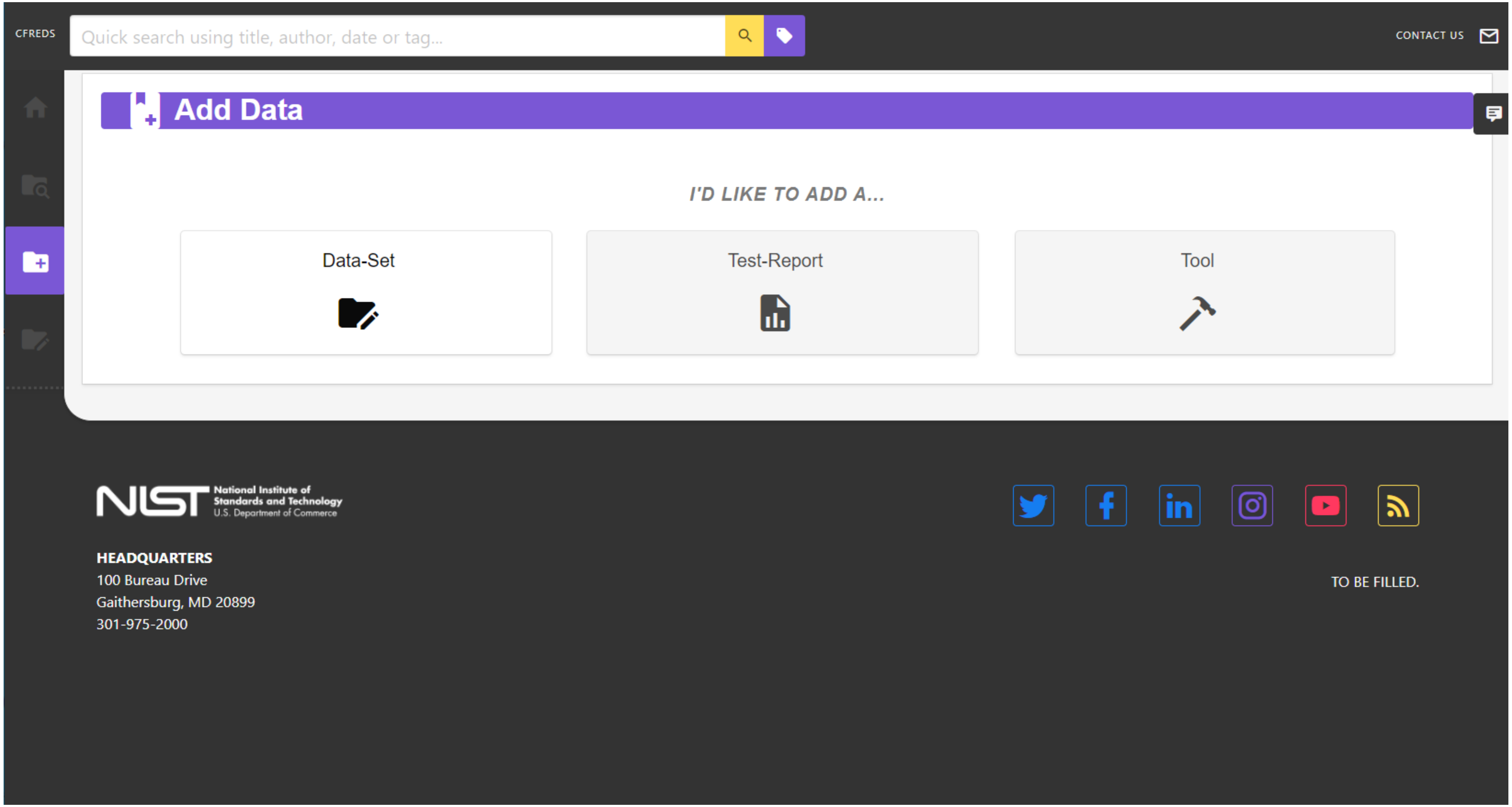
| | URL |
|---|---|
| | https://drive.google.com/drive/folders/1-UrxFGpCo54bVujwFmmqNbsZEV28dSNz?usp=sharing |

TIP: A GREEN CHECK-MARK MEANS YOU LAUNCHED THE DOWNLOAD FOR THE SPECIFIED FILE!

1-1 / 1

@STEVE WATSON / VTO INC. ON 2019. INPUTTED BY RICK AYERS ON 2020. DOWNLOADED 0 TIMES.

# CFReDS v2.5 Dataset



CFREDS

Quick search using title, author, date or tag...

CONTACT US

## Add Data

I'D LIKE TO ADD A...

Data-Set

Test-Report

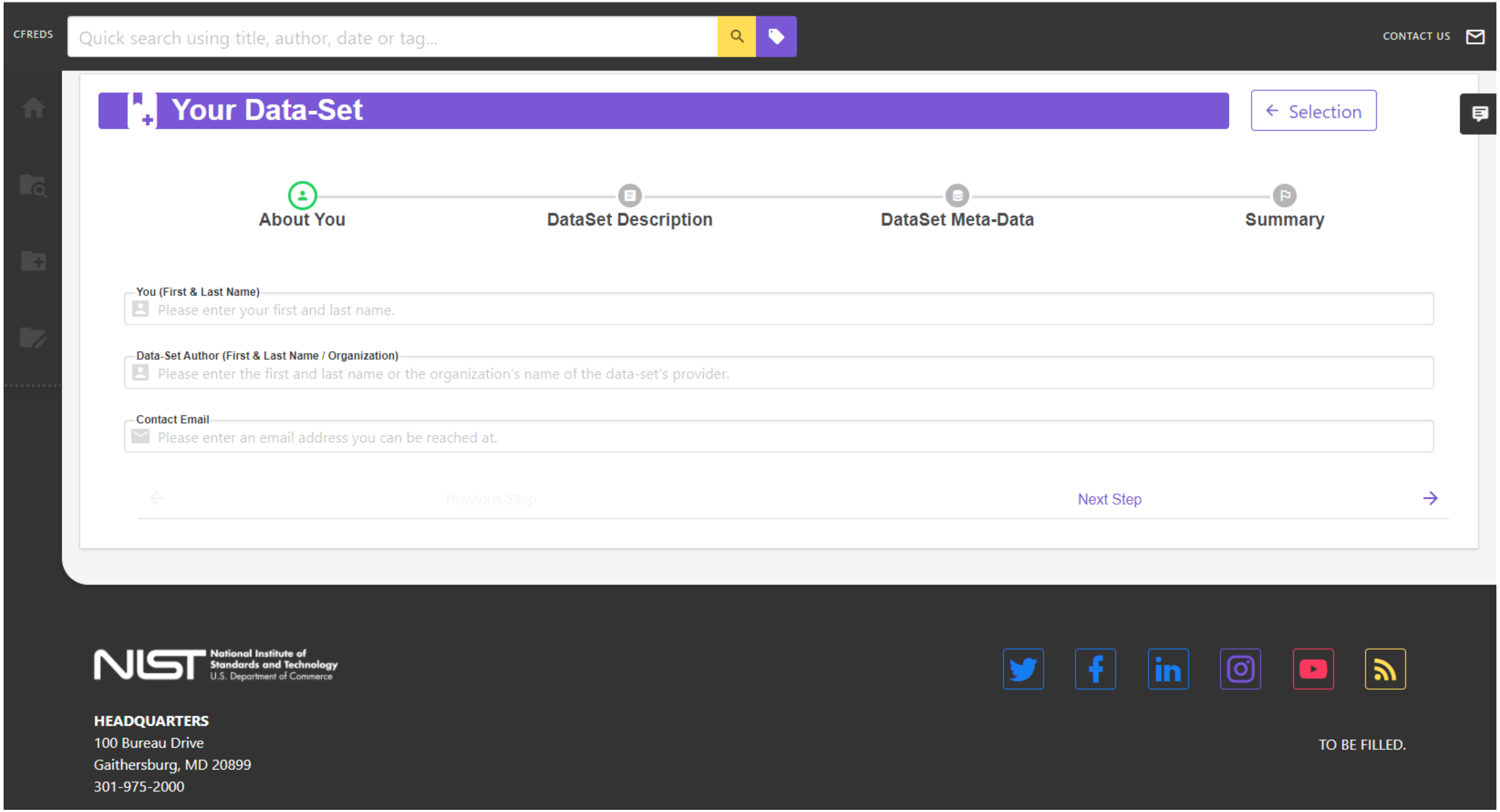
Tool

NIST National Institute of Standards and Technology U.S. Department of Commerce

**HEADQUARTERS**
100 Bureau Drive
Gaithersburg, MD 20899
301-975-2000

TO BE FILLED.

# CFReDS v2.5 Dataset



You (First & Last Name)
Please enter your first and last name.

Data-Set Author (First & Last Name / Organization)
Please enter the first and last name or the organization's name of the data-set's provider.

Contact Email
Please enter an email address you can be reached at.

NIST National Institute of Standards and Technology U.S. Department of Commerce

**HEADQUARTERS**
100 Bureau Drive
Gaithersburg, MD 20899
301-975-2000

TO BE FILLED.

- Upcoming Plans
  - Test Report Integration – will include metadata about test reports
  - Tool Catalogue Integration
  - Improved search facilities based on numerous factors:
    - Vendor software applications, Type of Report, Hardware, Applications, etc.

# DEMO

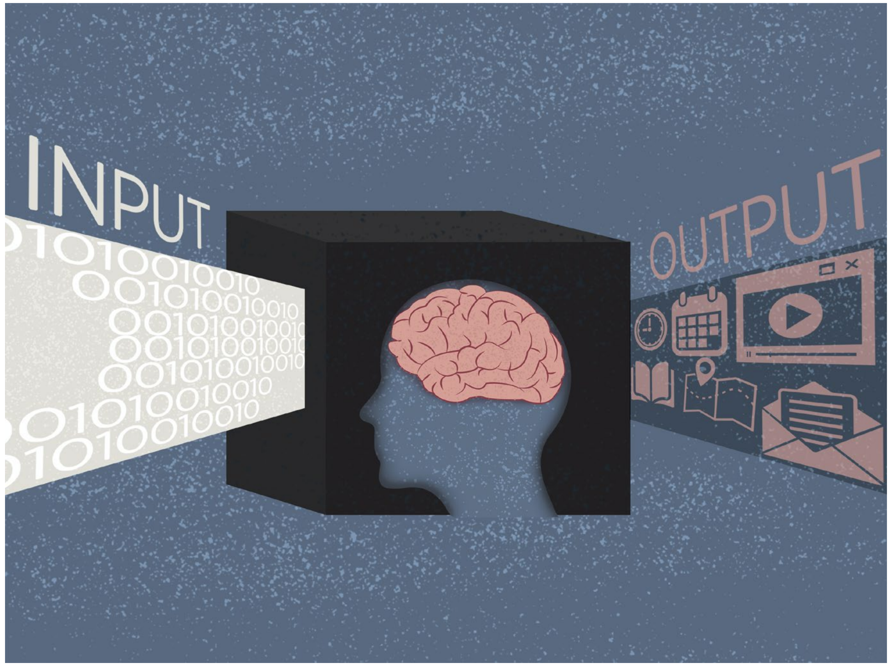
# NIST Black Box Study

Barbara Guttman

November 6, 2020

# Black Box Study

- DE - Scientific Foundation Study
- Two parts:
  - Black box study (interlab)
  - Big picture study



**NIST to Digital Forensics Experts: Show Us What You Got**

**First large-scale “black box” study will test the accuracy of computer and mobile phone forensics.**

**June 2, 2020.**

# Black Box Study

- Opened June 2
- Closed registration October 31
- Results are due November 30

- Study has two tests
  - PC
  - Mobile phone

# Questions

- nsrl.nist.gov
- cftt.nist.gov (for CFTT and Federated Testing)
- cfreds.nist.gov
- toolcatalog.nist.gov

bguttman@nist.gov