Hi, I’m with the Computer Forensic Tool Testing Project at the National Institute of Standards & Technology. We develop test methods and test data for testing forensic tools.
Disclaimer

Certain trade names and company products are mentioned in the text or identified. In no case does such identification imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the products are necessarily the best available for the purpose. No financial interest.

Tools and products mentioned (not endorsed): Autopsy, EnCase, FTK & X-Ways, MS Office, Mitsubishi and Subaru

I do not have any financial interest in any of these products. I do not endorse any of the products.
We develop methodologies for testing forensic tools and we apply the methodology to specific tools and the Department of Homeland Security publishes the results. We are also developing Federated Testing to distribute the tool testing effort and sharing of test results. Jenise will be talking about that in
the next session.
If you're going to test something, you need some test data. You also need to consider what attributes of the test data is likely to reveal something about the tool tested.

For string searching you need to know what strings are present and where the string you are looking for is. You also should have an idea of what you expect the tested tool to do.
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.

This is the general process for how we do testing. From a set of test cases select what applies to your situation and run those cases, configure the tool to run the case, Record the test results and when you finish create a test report.
A basic test case

<table>
<thead>
<tr>
<th>Case</th>
<th>Strings</th>
<th>Options</th>
<th>Case Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FT-SS-01</td>
<td>DireWolf</td>
<td>Case = Match Case, ASCII = True, Unicode = False, Whole Words = False</td>
<td>Search ASCII</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>Byte Offset</th>
<th>Containing File</th>
<th>File System</th>
</tr>
</thead>
<tbody>
<tr>
<td>0765</td>
<td>9207995</td>
<td>DELETED-Extinct-Lupus-fat-ascii.txt</td>
<td>fat32</td>
</tr>
<tr>
<td>0764</td>
<td>11006116</td>
<td>LIVE-Extinct-Lupus-fat-ascii.txt</td>
<td>fat32</td>
</tr>
<tr>
<td>0790</td>
<td>504563656</td>
<td>UNALLOCATED SPACE</td>
<td>unalloc</td>
</tr>
<tr>
<td>0767</td>
<td>1007456442</td>
<td>DELETED-Extinct-Lupus-exfat-ascii.txt</td>
<td>exfat</td>
</tr>
<tr>
<td>0766</td>
<td>1008124079</td>
<td>LIVE-Extinct-Lupus-exfat-ascii.txt</td>
<td>exfat</td>
</tr>
<tr>
<td>0788</td>
<td>1514852790</td>
<td>LIVE-Extinct-Lupus-nfs-ascii.txt</td>
<td>nfs</td>
</tr>
<tr>
<td>0789</td>
<td>1677365437</td>
<td>DELETED-Extinct-Lupus-nfs-ascii.txt</td>
<td>nfs</td>
</tr>
</tbody>
</table>

- 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

This is a simple test case. The goal of a set of test cases in a DE lab environment is to try the tool features that are relevant to the DE Lab’s work. Each individual test case will focus on a subset of tool features that are convenient to test together. In this test case, the main question examined is can the tool find an ASCII string. The secondary issues include: Type of file system, string surrounding environment: active or deleted file or unallocated space.
and tool option settings.

Find the ASCII string "DireWolf". The string appears seven times in the test data. The test image has four partitions. Three partitions are formatted and one partition is unformatted (no filesystem). In the formatted partitions, two files are created with the test string in each partition. One file from each partition is then deleted. Another copy of a file with the string is added to the unformatted partition.

Most search tools return context around the string hit and this makes the string ID visible and helps identify the string instance found by the tool.

We also have a test image for Mac and Linux file systems.
Here is what one tool reported for DireWolf. This tool did not find the string in unallocated space. Why?

Maybe I configured the search wrong? Maybe my test data is not what I think it is? Maybe the tool does not search unallocated space? I tried two other tools and got all 7 hits. Need to note this and look for a pattern in other test runs?
This is a glimpse of what you need to run that simple test case and record the test results. Settings for the search tool to run the case and expected results to know what to expect.
Here are the issues we considered for testing. We implemented test cases for these parameters because they might cause a search tool to reveal behavior that a user should be aware of. It turns out that these have been good choices because we often observe an unexpected result that could have been missed if we hadn’t tested these parameters. For example, if we just tried UTF-8 text we wouldn’t know if a tool misses UTF-16, both encodings.
have potential for mishandling by a tool since both are present in almost every file system.

This is where CFTT test cases focus, there are plenty of other issues that we don’t investigate.
Let's take a look at UNICODE searching. UNICODE testing can be very complicated; we can’t test everything that could be tested, so we tried to cover some high level features. Each of the strings has a different feature: Kanji, kana, hangul are all different scripts, A Chinese character may have two versions: traditional or simplified, the first character of Tokyo is also traditional Chinese, last character of China is a simplified character, Boy and
Beauty have diacritical marks and Arabic is written Right-to-left.
Here are the results for the UNICODE tests run against tool A. The misses column should contain only zeros. Oh dear, they are not all zeros here! Tool A is missing the UTF-16 copies of the strings. It also sometimes reports a string from a deleted file as belonging to unallocated space.

Let's try another tool.
## Unicode Search Results – Tool B

<table>
<thead>
<tr>
<th>Case</th>
<th>Expected String</th>
<th>Active Files</th>
<th>Delected Files</th>
<th>Unicode Space</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Expected</td>
<td>Misses</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>His</td>
<td>Misses</td>
<td>Expected</td>
</tr>
<tr>
<td>PT-SS-07-</td>
<td></td>
<td>18</td>
<td>0</td>
<td>18</td>
</tr>
<tr>
<td>CJK-ahar</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>18</td>
<td>0</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>18</td>
<td>0</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| All instances of search targets found

OK, no anomalies here.
Let's look at text with embedded formatting. Often text such as HTML or MS Office is stored this way with embedded tags. The list of possibilities is long. Look at crossbow. To match CrossBow a search tool has to ignore the bold and underline tags.
Formated Text Searches – Find nitrolycerin

The string nitrolycerin appears 4 times:

- Text in the FAT Partition (8005) and in unallocated space (8513)
- Formatted text in a docx file: nitrolycerin (8005 in FAT and 9513 in unallocated space.
- This tool found formatted text in FAT, but no tool found string in unallocated space.
- Tried two other tools with slightly different results

This is cool, nitrolycerin is formatted with embedded tags for the bold and underlining and packaged in a DOCX file which is really XML in a ZIP file.

I'm not sure it's fair to expect it to be found in unallocated space, may need to do file carving first before searching. Tool A found the HTML case, but tool B didn’t. Tool C found the formatted string in both DOCX & HTML using indexed search, but
missed both using a live search. I suspect I need to configure the search better.
Just because a tool being tested doesn't do what you wanted, it doesn't mean the tool made an error. It could be a feature instead of a bug or you could have messed up making the test data, messed up running the test or misinterpreted the results.

Knowing about problematic tool behaviors provides an opportunity to mitigate the effects.
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 was copying/pasting from a PDF.
- Arabic + PDF = Weird. The string renders correctly in the search tool, but the byte codes copied are not UNICODE.

Keep in mind that tool testing is a chance for you to learn things. It takes a lot of work to ensure that that a test data set will function as intended. Sometimes during the quality control process unknown unknowns reveal themselves in interesting ways.

I found a mystery one day. I have a program that examines my test image and reports all the locations for each test
string. Sometimes it missed strings that it should have found, so I used a hex editor and tracked down the strings and sure enough they had two bites of corruption. Now the cases where this showed up were all for special test with NTFS filesystems. If you have a really small file it is actually stored in the MFT. The corruption was the fix up byte at the end of the sector.

My other mystery showed up when Arabic searches stopped working, not just for one tool but all the tools. It turns out that the way Arabic was stored in my PDF file was not what I expected, so the copy/paste didn't transfer something that would match in the search. I’m still looking at the issue but copy and paste from PDF may get you something unexpected.
Some Observed Tool Behaviors

- All tools could parse FAT, ExFAT, NTFS, ext4, journaled OSX and case-sensitive OSX partitions.
- 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 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.

These are some of our preliminary observations from informal trials of some widely used forensic tools. More complete formal testing is coming soon.

Most of these behaviors are situations where the search tool might miss a string.

I didn't see any situations where a tool said that something was in the image when it wasn't there.
What Does Software Testing Get 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.

As you have hopefully seen, our string search test cases shows that string search tools work in general and will alert the forensic practitioner to limitations that can be mitigated.

It is challenging to find the right questions. You want each question to bring something unique to the test. You want each question to encourage the tool to do something different. With this test
data we found different behavior based on file system, search method (engine), character encoding, language, active-deleted state, and formatted file type.
The goal of federated testing is to move high quality testing to labs and to produce more test reports for more tools to enable sharing the tool test results. Federated testing makes the NIST test methods available to a wide audience of users so that many organizations can use the same method to test tools and produced test reports in a similar format. By using the same or similar test data it is easy to compare results for testing tools by
different organizations. In this way, labs can help each other too. Jenise will say more about federated testing.
Contact Information

Jim Lyle
jlyle@nist.gov
http://www.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).

Get on the mailing list if you want to know about federated testing.

Questions?

Thank you for your attention.

Bye now.