DFRWS

The 2013 Data Sniffing Challenge

Submission deadline: **May 30, 2013**

The overall goal of this challenge is to bring the state of the art into the digital forensic practice by providing an open public venue for a best-of-breed competition. We challenge the competitors to develop the fastest and most accurate data block classifier. The winner will be announced in Aug 2013 at the DFRWS'13 conference in Monterey, CA.

The scoring will be based on the weighted scores of three criteria:

1. *Correctness*, as measured by (IR) precision & recall rates: 55%.
2. *Processing speed*, in terms of throughput & scalability: 30%.
3. *Quality of code* and multi-platform support: 15%.

Rules:

- You may enter individually, or as a team with no restrictions.
- Your tool must have a command line interface and work on at least one of the three main OS platforms—MS Windows, MacOS, Linux—preferably more. It must be implemented in a widely and freely available language platform.
- The tool must have a corresponding library/API such that it could be incorporated as part of other tools.
- Source code must be openly available under a free software license, such as those listed at http://www.gnu.org/licenses/license-list.html. The author(s) retain rights to the source code.
- You may incorporate third party free software, as long as it is compatible with your license and it is included with your submission. However, your submission will be judged on the contribution your own work brings to the challenge.
- Your submission must include clear instructions for building the tool from source code along with all relevant dependencies.
- DFRWS will publish the results of the Challenge, both in detailed and summary form, along with the methodology used and the source of the specific version of each tool.

Technical Requirements:

- Command line invocation:

```
$ <tool_name> <target> <block_size(bytes)> [<concurrency_factor>]
```

- *Tools must work right out of the box*, and will be tested both on actual drive images, as well as sequences of block samples glued together for convenience.

    - Whenever drive images are used, those will be produced by repeated cycles of create/delete file operations; in other words, they will be realistic but of the "difficult" variety. Also, they may lack in certain details, such as filesystem metadata.
    - The target can be of substantial size, e.g. 100GB.
    - The target's file system could be any of FAT, NTFS, or ext3.
- The block sizes we will be testing for are 1024, 4096, 16384, and 65536 bytes.
- The concurrency factor is optional. If your tool does support multi-threading/-processing, it will be tested with up to six values: 1, 4, 8, 16, 24, 48 to evaluate its scalability on commodity hardware.

Output rules:

- The output should consist of one line per block and should identify the offset of the block and the type of data being detected by the tool.
- If multiple types are detected, they should be outputted separated by space. The first type should identify the top level container (e.g., doc, pdf, etc.).
- If your data classifier is able to analyze popular encodings and identify the underlying data, it should first output the type of the data encoding (e.g., base64) and then the type of the underlying data (e.g., jpg), and connect the two by hyphen: e.g., base64-jpg.

Example output (*with comments explaining the meaning*):

```
> data_sniffer target 1024
0 jpg JPEG data
1024 jpg xml            # XML inside a JPEG (presence of JPEG data is implied)
2048 jpg jpg            # JPEG inside another JPEG (thumbnail)
3072 pdf jpg zlib       # JPEG & deflate-compressed data as part of a PDF document
4096 html js            # JavaScript inside html
5120 zlib-xml           # Zlib-compressed xml
6144 pdf base85-jpg     # PDF document with base85-encoded JPEG
7168 null               # Unknown/unable to classify
```

Data types of interest:

The following is a list of the expected output file types and their respective interpretation. A tool's ability to handle additional common data types would be used to help decide a tie or near-tie.

`txt, csv, log`

Text content: plain text, comma-separated values, system log. Note that the csv designation also covers the case where the fields are separated by a different character (<space>, <tab>, "|", etc.).

`html, xml, css`

Web mark-up data: HTML/XML-encoded data; CSS.

`js, json`

JavaScript code, JSON data.

`base64, base85, hex`

Text-encoded binary data: base64/85, hexadecimal encoded data.

`jpg, png, gif, fax, jbig2`

Full-color image data: JPEG, PNG, GIF; bi-tonal images (common in scanned documents): CCITT Fax and JBIG.

`zlib, bzip2`

General-purpose compression: DEFLATE (RFC 1951) and bzip2 (http://bzip.org).

`zip`

Compressed data in zip format.

`pdf`

Portable document format documents.

`msdoc, msxls, msppt`

Microsoft Office 97-2003 compound documents.

`msdocx, msxlsx, mspptx`

Microsoft Office 2007 compound documents.

`mp3, aac`

Audio: MPEG layer III, AAC-encoded audio.

`h264, avi, wmv, flv`

Video encoding & packaging: H.264, AVI, WMV, Flash video.

`fs-fat, fs-ntfs, fs-ext`

Filesystem metadata for FAT, NTFS, ext3.

`encrypted, random, constant, null`

Special cases: encrypted, random, constant data, and unknown data. For the constant designation, at least half the block must be of the same value; constants may be 16 bits.

Testing and Scoring:

- The testing procedure will consist of two parts—controlled tests and real data tests.
    - Controlled tests consist of running the tools against targets wth well-known ground truth. The point is to evaluate the basic suitability of the tool with cleanly separable data encodings that can be reviewed manually.

It is important for the tool's results to be as specific as possible; for example, for a `docx` file, an alternative to the `ms-docx` classification would be `zlib-xml`, however, we will only count `ms-docx` as correct. (There will be a separate test for `zlib` classification.)
To help the developers, we are publishing the test data used in the evaluation of the 2012 submissions—a description of the files and test cases is given in an appendix.

- Real data will consist of large targets filled with a variety of real-world files. It will be inherently much noisier and there will be no room for manual review (due to size). Some automated sanity checks will be applied; the rest will depend primarily on filesystem metadata and file type 'self-identification'.

Notes:

- For primary data encodings, such as text, markup, zlib, etc., the minimum chunk of discovery is 256 bytes. For compound data formats, like zip & pdf, that are no minimums.
- Some of the classification tasks, such as distinguishing encrypted from random, are known to be difficult and may not be solvable in the general case.
- The dash notation indicates the *underlying encoding* of the same piece of data; space indicates *separate, non-overlapping* pieces. In the above example:

  `pdf jpg zlib` : this is part of a PDF doc, it contains (pieces of) a JPG-encoded element (streams in PDF parlance) and (pieces of) zlib-compressed data (probably text).

  `html js` : html and javascript detected.

  `zlib-xml` : this is zlib-compressed XML data (e.g. ms-docx); zlib would also be correct classification, but less specific

  `pdf base85-jpg` : this part of a PDF, it contains JPEG data that is base85-encoded; base85 would be correct but less specific
- Your tool may produce additional information/comment for a sample following a '#' sign. For example:

  ```
  1024 jpg #beginning of file
  ```

  This is strictly optional and will only be taken into account to break ties.

Most test data will be obtained from public Internet sources. We expect that text content will be in English, however, no special filter will be applied. If you wish to obtain test data for development and tool testing, you may consider the data sets at http://digitalcorpora.org and http://www.cfreds.nist.gov, among other publically available.

Submission:

All participants must send an email to `dfrws2013-challenge@dfrws.org` with the subject line *"Solution submission"*. The email should contain official contact information for the participant/team members; it should also indicate to whom a check should be made out, in case the solution is selected for the grand prize.

The actual solution (code and relevant documentation) can be submitted in one of three ways:

- *Email attachment*. If the entire submission can be packed in an archive of less than 5MB, then submission can be sent as an attachment to dfrws2013-`challenge@dfrws.org`.
- *http/ftp download*. The submission email can contain a download link from where the submission can be downloaded as a single file.
- *svn/git checkout*. The submission email should contain appropriate instructions and credentials (if applicable) for organizers to obtain the submission.

Ideally, submissions should be self-contained; however, if bundling of third-party code is not possible (e.g., due to licensing restrictions) appropriate instructions on building the tool should be included.

As stated above, this competition is for open-source tools and, in the interest of open competition, DFRWS may publish the actual submissions along with test results. Beyond that, DFRWS will make no further attempts to distribute the solutions. Although we strongly encourage toolmakers to cover as wide a range of data types as possible, all submissions will be given a fair chance, even if they do not cover all targeted data types.

Prizes:

- **First prize:** DFRWS will provide free conference registration to our 2013 conference for up to two members of the winning team.
- **Grand prize:** DFRWS seeks to award an additional $1,000 cash prize to the winners, if their solution exhibits all the attributes of a field-ready tool with the necessary robustness and performance.
- The decision on prizes will be made by the DFRWS Organizing Committee based on the tool testing results conducted by the challenge team consisting of the following OC members: Vassil Roussev, Wietse Venema, and Eoghan Casey.

Contact:

Send all questions to `dfrws2013-challenge@dfrws.org`. (Your email will be used only for this purpose and will be forgotten after DFRWS'13.)

Good luck!

Appendix: Test Data Sets

http://dfrws.org/2013/challenge/dfrws-13-challenge-tests.zip

Most of the data samples were chosen to be in the 1-2 MB range; this was done to enable manual inspection of the source and to manage the long processing of some of the solutions. All data is sourced from the NPS GovDocs corpus, unless otherwise noted, and carries no copyright restrictions.

Text

1. **`txt-01`**: 1.0M, text sample from the beginning of Shakespeare's Complete Works in plain ASCII text with no formatting.

   ASCII text is the simplest encoding and just about any text source would be suitable. We chose the *Shakespeare* sample as it would not contain any technical terms that might appear in other encodings as keywords, and has very few numeric values.

2. **`txt-02`**: 994K, sample of CSV data. The sample provides both textual and numerical values and has a very regular structure:

   ```
   SOLA2,3,56.00000 ,60.43128 ,-151.1286,AKDOT, ,Sterling Hwy @ DOT
   Soldotna MS  - MP 9
   FTSA2,3,411.0000 ,60.64045 ,-149.5007,AKDOT, ,Seward Hwy @ Summit Lake
   Lodge MP 45.8
   RUFA2,3,108.0000 ,60.48723 ,-150.0021,AKDOT, ,Sterling Hwy @ Russian R.
   Ferry MP 54.8
   NRBA2,3,27.00000 ,60.04784 ,-151.6589,AKDOT, ,Sterling Hwy @ Ninilchik
   River Bridge
   ...
   ```

3. **`txt-03`**: 914K, sample of tabular data separated by "|". This appears to be a standardized nomenclature of terms, designed for import into a database:

   ```
   E0000046|A-1|A1|
   E0000050|A.A.M.D.|AAMD|
   E0000081|AIDS related complex|AIDS-related complex|
   E0000154|act|Act|
   E0000155|active vertical corrector|Active Vertical Corrector|
   ...
   ```

4. **`txt-04`**: 975K, sample from an actual *http* log:

   ```
   10.1.1.140 - - [01/Aug/2009:09:36:58 -0700] "GET
   /home/themes/ComBeta/images/ ...
   12.1.1.140 - - [01/Aug/2009:09:36:58 -0700] "GET
   /themes/ComBeta/images/corne ...
   12.1.1.140 - - [01/Aug/2009:09:36:58 -0700] "GET
   /home/themes/ComBeta/images/ ...
   12.1.1.140 - - [01/Aug/2009:09:36:58 -0700] "GET
   /themes/ComBeta/images/bugs/ ...
   ```

. . .

Markup

1. **`ml-01`**: 1.1M, sample of concatinated *HTML* files with **no** embedded javascript. We took a large number of *html* files and filtered out any containing the `script` tag. To provide variety and reduce the possibility of skewing the results, we used small-to-medium files. They were concatinated together into a single file; this randomizes the alignment of the beginning/ending of the files.
2. **`ml-02`**: 1.1M, concatinated *XML*. These were produced the same way as the prior set, except there was no need to filter out code.

Javascript

1. **`js-01`**: 1.2M, concatinated *Javascript* code. Consists of the source of five common libraries (dojo-1.7.3.js, ext-4.1.1.js, jquery-1.7.2.js, mootools-core-1.4.5.js, yui-3.5.1.js) stripped of all comments.
2. **`js-02`**: 606K, concatinated *minimized Javascript* code. Consists of the minimized versions of the same five libraries.
3. **`js-03`**: 1M, a 'pure' *JSON* data sample with very minimalistic content; format is easy to recognize from very small samples. There are no large blobs of text/markup to confuse the classifier.

Image

1. **`img-01`**: 1.3M, concatinated *JPEG* files, with **no metadata**. We used the **`exiftool -all= *.jpg`** command to produce 'pure' versions of the images, presumably with no distractions for the classifiers. The expectation is that classifiers should be able to characterize all the data as JPEG, as there are no "noise" such as *EXIF* and other metadata.
2. **`img-03`**: 1.4M, concatinated *PNG* files; **as is**. Note: PNG files contained insignificant amounts of embedded metadata.
3. **`img-03`**: 1.2M, concatinated *GIF*; **as is**. Note: GIF files contained insignificant amounts of embedded metadata.

PDF

PDF files consist of a sequence of objects (of various encodings) glued together with some PDF-specific markup. Thus, the classification problem is two-fold: when the sample contains the inside of an object the classifier must identify the object's encoding; otherwise, it must identify the markup as PDF-specific.

There is an even trickier problem--what constitutes a typical PDF file, and by what criteria is that to be established?

To establish the ground truth without solving the complete classification problem, and to produce meaningful comparisons, we used two techniques: a) we carved out individual streams from the PDF files; and b) we (quantitavely) extracted a list of PDF markup keywords.

Carving out the streams is a straighforward process and can be accomplished with any configurable carver by using the `obj`/`endobj` keywords and filtering out all files that do not contain `stream` and `endstream`. The benefit of this carving process is that we can easily identify the encoding of each stream (using the markup in the beginning of the object).

For the actual test sets, we create synthetic PDF targets by concatinating a set of the same type (e.g., `FlateDecode`) and searching each of its blocks for PDF markup keyword. If a block contains a keyword, then it is recognizable as PDF. In any case, we know the encoding for the remainder of the block, so for each block the classification should be either `<offset> pdf <encoding>`, or `<offset> <encoding>`.

By creating synthetic targets, we get around the problem of what constitutes a typical target--if a classifier can handle successfully each of the different object encodings, and recognizes the PDF markup, then we can conclude that it can classify PDF files. If not, we can precisely pinpoint its weaknesses.

The following are the data sets used in the evaluation process (all derived from a 121M sample from the NPS GovDocs corpuso):

1. **`pdf-01`**: 1.9M, concatinated *FlateDecode* data streams between 512 and 50000 bytes in size.
2. **`pdf-02`**: 2.0M, concatinated *DCTDecode* data streams, 512--50000 bytes.
3. **`pdf-03`**: 2.0M, concatinated *CCITTFaxDecode* data streams, 512--50000 bytes.
4. **`pdf-04`**: 1.5M, concatinated *JBIG2Decode* data streams (1K-85K); includes all available data from the set.

One notable omission are `LZWDecode` streams. We did not test these as these were not in the original challenge specification (as they should have).

Lossless compression

1. **`z-01`**: 1.1M, sample of *zlib*-compressed data, using the *gzip* tool; header info is removed.
2. **`z-02`**: 1.2M sample of *zlib*-compressed data, using *zip* tool. Target consists of a (relatively) large number of small (text/markup) files; nothing is removed.
3. **`z-03`**: 951K, sample of *bzip*-compressed data, using the *bzip2* tool; all header info is removed.
4. **`z-04`**: 951K, *bzip*-compressed data, using the *bzip2* tool. Same data as **`z-03`** but nothing is removed.

Text-encoded binary data

1. **`b-01`**: 1M, base64-encoded plain text.
2. **`b-02`**: 1M, base64-encoded *zlib*-compressed data (compressed data is text).
3. **`b-03`**: 1M, base85-encoded *zlib*-compressed data (from PDF streams).
4. **`b-04`**: 1M, hexadecimal (base16)-encoded plain text.
5. **`b-05`**: 1M, hexadecimal (base16)-encoded compressed data.

MS Office documents (07)

1. **`msx-01`**: 1.2M, 72 concatinated `ms-docx` documents. The documents were generated by converting *html* documents from the *NPS GovDocs* corpus to `docx`, producing files between 11K and 48K. All documents were opened in MS Word to verify correctness.
2. **`msx-02`**: 1.6M, 13 concatinated `ms-xlsx` documents; all downloaded from US Govt servers.
3. **`msx-03`**: 1.8M, 8 concatinated `ms-pptx` documents; all downloaded from US Govt servers.

Note:

None of the documents contain images; these were screened both manually and by carving.

Audio

1. **`au-01`**: 1M, mp3-encoded audio sample; no additional metadata.
2. **`au-02`**: 1M, aac-encoded audio sample; no additional metadata.

Video

1. **`vid-01`**: 2.1M, h264-encoded video sample.
2. **`vid-02`**: 1.2M, wmv-encoded video sample.