DFRWS 2013 Forensic Challenge Report

Release 1.0

Vassil Roussev

August 04, 2013
CONTENTS

1 Introduction
   1.1 Submissions .................................................. 1
   1.2 Test Data Sets ............................................... 1

2 2012 Data Set
   2.1 Text ......................................................... 3
   2.2 Markup ...................................................... 4
   2.3 Javascript .................................................. 4
   2.4 Images ....................................................... 4
   2.5 PDF .......................................................... 4
   2.6 Lossless compression ....................................... 5
   2.7 Text-encoded binary data .................................. 5
   2.8 MS Office documents (2007) ............................... 5
   2.9 Audio ......................................................... 5
   2.10 Video ......................................................... 6

3 Test Results: 2012 Data Set
   3.1 Text ......................................................... 7
   3.2 Markup ...................................................... 7
   3.3 Javascript/JSON ............................................. 8
   3.4 Images ....................................................... 8
   3.5 PDF .......................................................... 8
   3.6 Compressed .................................................. 9
   3.7 MS Office (2007) .......................................... 9
   3.8 Base64/85/16-encoded ..................................... 9
   3.9 Audio ......................................................... 10
   3.10 Video ......................................................... 10
   3.11 Timing ...................................................... 10

4 Test Results: 2013 Data Set ........................................ 11

5 Observations ....................................................... 13

6 Teams ............................................................... 15

7 Winner ............................................................. 17
The DFRWS 2013 Forensic Challenge is an effort to improve upon the state of the art in digital forensic techniques in the area of data encoding classification. This year’s challenge is an extension of the 2012 Challenge and is, again, an explicit ‘best-of-breed’ competition of tools.

1.1 Submissions

There were two submissions, referred throughout the report as S1 and S2, which is lower than last year’s six. However, despite the lower number, the quality of this year’s submissions easily exceeds that of 2012. There was an evident effort to produce practical tools that work on multiple platforms.

1.2 Test Data Sets

The evaluation was done with two data sets–the 2012 set, which was published with the challenge, and an additional 2013 set, which consisted of more complex test.

One of the principle problems of prior published work is the fact that researchers did not have a clear appreciation of the complexity of the undelying data being classified. As a result, the evaluation results were difficult to interpret and compare with each other.

The main goal of the controlled experiments is to produce a reference data set with known ground truth. For many complex data encodings (e.g. PDF), this cannot be accomplished by simply downloading a random set of files from the Internet–deeper analysis is needed to get to the ground truth. (It is somewhat ironic that, in the process of establishing the ground truth, the challenge organizers/evaluators had to partially solve the challenge problem themselves.)
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.

### 2.1 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|
   0000154|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/ ...
   ...
2.2 Markup

1. **ml-01**: 1.1M, sample of concatenated 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 concatenated together into a single file; this randomizes the alignment of the beginning/ending of the files.

2. **ml-02**: 1.1M, concatenated XML. These were produced the same way as the prior set, except there was no need to filter out code.

2.3 Javascript

1. **js-01**: 1.2M, concatenated 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, concatenated 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.

2.4 Images

1. **img-01**: 1.3M, concatenated 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, concatenated PNG files; as is. Note: PNG files contained insignificant amounts of embedded metadata.

3. **img-03**: 1.2M, concatenated GIF; as is. Note: GIF files contained insignificant amounts of embedded metadata.

2.5 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 straightforward 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 concatenating 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 corpus):

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

### 2.6 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.

### 2.7 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.

### 2.8 MS Office documents (2007)

1. **msx-01**: 1.2M, 72 concatenated 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 concatenated ms-xlsx documents; all downloaded from US Govt servers.
3. **msx-03**: 1.8M, 8 concatenated ms-pptx documents; all downloaded from US Govt servers.

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

### 2.9 Audio

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

### 2.10 Video

1. **vid-01**: 2.1M, h264-encoded video sample.
2. **vid-02**: 1.2M, wmv-encoded video sample.
The overall goal was to achieve the most specific possible classification. To that end, the tools are first given a generic example of the data encoding (e.g., text), followed by more specific examples (csv,log). In the latter case, we only count the specific results as true positives.

Based on current research, we used larger block size of 4,096 and 16,384 bytes (as opposed to 512 and 4,096 last year) as these provide a more realistic opportunity for correct classification.

### 3.1 Text

Acceptable classifications:

1. **txt-01:** txt
2. **txt-02/03:** csv
3. **txt-04:** log
4. All other classification are considered incorrect.

All test targets have been chosen such that they provide typical and unambiguous samples, therefore, the expectation is that all blocks are readily classifiable.

<table>
<thead>
<tr>
<th></th>
<th>4k S1</th>
<th>4k S2</th>
<th>16k S1</th>
<th>16k S2</th>
</tr>
</thead>
<tbody>
<tr>
<td>txt-01 (text)</td>
<td>71</td>
<td>100</td>
<td>78</td>
<td>100</td>
</tr>
<tr>
<td>txt-02 (csv)</td>
<td>0</td>
<td>100</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>txt-03 (pipe)</td>
<td>0</td>
<td>100</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>txt-04 (log)</td>
<td>73</td>
<td>100</td>
<td>77</td>
<td>100</td>
</tr>
<tr>
<td><strong>average</strong></td>
<td>36</td>
<td>100</td>
<td>39</td>
<td>100</td>
</tr>
</tbody>
</table>

### 3.2 Markup

Acceptable classifications:

1. **ml-01:** xml, txt *(where applicable)*;
2. **ml-02:** xml

Results: True positive rate (percent)
### 3.3 Javascript/JSON

Acceptable classifications:

- **js-01/02/03**: js, json.

**Results**: *True positive rate* (percent)

<table>
<thead>
<tr>
<th></th>
<th>4k S1</th>
<th>4k S2</th>
<th>16k S1</th>
<th>16k S2</th>
</tr>
</thead>
<tbody>
<tr>
<td>js-01 (lib)</td>
<td>43</td>
<td>98</td>
<td>57</td>
<td>99</td>
</tr>
<tr>
<td>js-02 (minlib)</td>
<td>99</td>
<td>98</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>js-03 (json)</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td><strong>average</strong></td>
<td>81</td>
<td>99</td>
<td>86</td>
<td>100</td>
</tr>
</tbody>
</table>

### 3.4 Images

Acceptable classifications:

1. **img-01**: jpg
2. **img-02**: png, zlib
3. **img-03**: gif

**Results**: *True positive rate* (percent)

<table>
<thead>
<tr>
<th></th>
<th>4k S1</th>
<th>4k S2</th>
<th>16k S1</th>
<th>16k S2</th>
</tr>
</thead>
<tbody>
<tr>
<td>img-01 (jpg)</td>
<td>100</td>
<td>17</td>
<td>0</td>
<td>53</td>
</tr>
<tr>
<td>img-02 (png)</td>
<td>80</td>
<td>99</td>
<td>76</td>
<td>100</td>
</tr>
<tr>
<td>img-03 (gif)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>45</td>
</tr>
<tr>
<td><strong>average</strong></td>
<td>60</td>
<td>39</td>
<td>25</td>
<td>66</td>
</tr>
</tbody>
</table>

### 3.5 PDF

Acceptable classifications:

<table>
<thead>
<tr>
<th>set</th>
<th>GT</th>
<th>acceptable</th>
</tr>
</thead>
<tbody>
<tr>
<td>pdf-01</td>
<td>pdf</td>
<td>pdf, pdf zlib, zlib pdf</td>
</tr>
<tr>
<td></td>
<td>zlib</td>
<td></td>
</tr>
<tr>
<td>pdf-02</td>
<td>pdf</td>
<td>pdf, pdf jpg, jpg pdf</td>
</tr>
<tr>
<td></td>
<td>jpg</td>
<td>jpg</td>
</tr>
<tr>
<td>pdf-03</td>
<td>pdf</td>
<td>pdf, pdf fax, fax pdf</td>
</tr>
<tr>
<td></td>
<td>fax</td>
<td></td>
</tr>
<tr>
<td>pdf-04</td>
<td>pdf</td>
<td>pdf, pdf jbig, jbig pdf</td>
</tr>
<tr>
<td></td>
<td>jbig</td>
<td></td>
</tr>
</tbody>
</table>

Chapter 3. Test Results: 2012 Data Set
Note: GT == ground truth

Results: True positive rate (percent)

<table>
<thead>
<tr>
<th></th>
<th>4k S1</th>
<th>S2</th>
<th>16k S1</th>
<th>S2</th>
</tr>
</thead>
<tbody>
<tr>
<td>pdf-01 (defl)</td>
<td>52</td>
<td>94</td>
<td></td>
<td></td>
</tr>
<tr>
<td>pdf-02 (jpg)</td>
<td>78</td>
<td>23</td>
<td></td>
<td></td>
</tr>
<tr>
<td>pdf-03 (fax)</td>
<td>0</td>
<td>79</td>
<td></td>
<td></td>
</tr>
<tr>
<td>pdf-04 (jbig)</td>
<td>0</td>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>average</td>
<td>32</td>
<td>51</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3.6 Compressed

Acceptable classifications:
1. z-01/02: zip pkzip
2. z-03/04: bzip2

Results: True positive rate (percent)

<table>
<thead>
<tr>
<th></th>
<th>4k S1</th>
<th>S2</th>
<th>16k S1</th>
<th>S2</th>
</tr>
</thead>
<tbody>
<tr>
<td>z-01 (zlib)</td>
<td>61</td>
<td>0</td>
<td>39</td>
<td>0</td>
</tr>
<tr>
<td>z-02 (zlib)</td>
<td>70</td>
<td>94</td>
<td>100</td>
<td>94</td>
</tr>
<tr>
<td>z-03 (bz2)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>z-04 (bz2)</td>
<td>0</td>
<td>100</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>average</td>
<td>33</td>
<td>48</td>
<td>35</td>
<td>50</td>
</tr>
</tbody>
</table>

3.7 MS Office (2007)

The ms-docs/xlsx/ppts documents are fully-formed zip archives; therefore, the purpose of this test is to understand how specific the tools can be in identifying the underlying content.

Acceptable classifications:
1. ms-docx: zip-html zip-xml
2. ms-xlsx: xlsx ms-xlsx ms-xlsx zlib zlib ms-xlsx ms-xlsx zlib-compound
3. *ms-pptx*: pptx ms-pptx zip-pptx ms-pptx zlib pkzip ms-pptx zlib ms-pptx

Results: True positive rate (percent)

<table>
<thead>
<tr>
<th></th>
<th>4k S1</th>
<th>S2</th>
<th>16k S1</th>
<th>S2</th>
</tr>
</thead>
<tbody>
<tr>
<td>msx-01 (docx)</td>
<td>74</td>
<td>100</td>
<td>90</td>
<td>100</td>
</tr>
<tr>
<td>msx-02 (xlsx)</td>
<td>0</td>
<td>97</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>msx-03 (pptx)</td>
<td>0</td>
<td>100</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>average</td>
<td>25</td>
<td>99</td>
<td>30</td>
<td>100</td>
</tr>
</tbody>
</table>

3.8 Base64/85/16-encoded

Acceptable classifications:
1. **b-01/02**: base64 base64-txt  
2. **b-03**: base85 base85-txt  
3. **b-04/05**: hex hex-txt  

Results: *True positive rate (percent)*

<table>
<thead>
<tr>
<th></th>
<th>4k</th>
<th></th>
<th>16k</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>S1</td>
<td>S2</td>
<td>S1</td>
<td>S2</td>
</tr>
<tr>
<td>b-01 (base64)</td>
<td>0</td>
<td>100</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>b-02 (base64)</td>
<td>0</td>
<td>100</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>b-03 (base85)</td>
<td>0</td>
<td>100</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>b-04 (hex)</td>
<td>0</td>
<td>92</td>
<td>0</td>
<td>91</td>
</tr>
<tr>
<td>b-05 (hex)</td>
<td>0</td>
<td>100</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td><strong>average</strong></td>
<td>0</td>
<td>98</td>
<td>0</td>
<td>98</td>
</tr>
</tbody>
</table>

### 3.9 Audio

Acceptable classifications:
1. **au-01**: mp3  
2. **au-02**: mp4 aac  

Results: *True positive rate (percent)*

<table>
<thead>
<tr>
<th></th>
<th>4k</th>
<th></th>
<th>16k</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>S1</td>
<td>S2</td>
<td>S1</td>
<td>S2</td>
</tr>
<tr>
<td>au-01 (mp3)</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>au-02 (aac)</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td><strong>average</strong></td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>4</td>
</tr>
</tbody>
</table>

### 3.10 Video

Acceptable classifications:
1. **vid-01**: h.264  
2. **vid-02**: wmv  

Results: *True positive rate (percent)*

<table>
<thead>
<tr>
<th></th>
<th>4k</th>
<th></th>
<th>16k</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>S1</td>
<td>S2</td>
<td>S1</td>
<td>S2</td>
</tr>
<tr>
<td>vid-01 (h.264)</td>
<td>0</td>
<td>47</td>
<td>0</td>
<td>66</td>
</tr>
<tr>
<td>vid-02 (wmv)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td><strong>average</strong></td>
<td>0</td>
<td>24</td>
<td>0</td>
<td>34</td>
</tr>
</tbody>
</table>

### 3.11 Timing

The single-core execution time for **S1** (for the entire test suite) was 42 seconds; for **S2**–188 seconds.
TEST RESULTS: 2013 DATA SET

To give a more thorough evaluation of the tools, we created a new data set that focuses on classifying compressed data—arguably the most difficult classification task.

Based on the results from the 2012 test, we picked several categories on which at least one of the tools seems to be doing quite well. We skipped multimedia types as these do not seem adequately covered (even for cases with known solutions, such as mp3).

Specifically, we created the following source data sets, 1MiB each:

1. zlib-txt: a set of small (<25KB) plain text files were individually compressed with gzip; gzip header is removed and the files are concatenated.
2. zlib-html: a set of small html files were individually compressed with gzip; gzip header is removed and the files are concatenated.
3. bzip2-txt: same data and process as 1. but using the bzip2 compressor.
4. bzip2-html: same data and process as 1. but using the bzip2 compressor.
5. png: concatenation of small png files.
6. ms-docx: a concatenation of small docx files with no embedded images.
7. ms-xlsx: a concatenation of small xlsx files with no embedded images.

Note: The use of small files is to ensure that there are multiple compressed blocks in the data as gzip tends to produce a single compressed block.

From the above sets, we created mixed targets by extracting exactly one block of 4/16KiB from each source in a round-robin fashion and gluing them together. The end result is shown below:

Results: True positive rate (percent) for 2013 data set

<table>
<thead>
<tr>
<th></th>
<th>4k</th>
<th>16k</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>S2</td>
<td>S1</td>
</tr>
<tr>
<td>01 zlib-txt</td>
<td>34</td>
<td>1</td>
</tr>
<tr>
<td>02 zlib-html</td>
<td>22</td>
<td>0</td>
</tr>
<tr>
<td>03 bz2-txt</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>04 bz2-html</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>05 png/zlib</td>
<td>82</td>
<td>64</td>
</tr>
<tr>
<td>06 ms-docx</td>
<td>8</td>
<td>59</td>
</tr>
<tr>
<td>07 ms-xlsx</td>
<td>0</td>
<td>13</td>
</tr>
</tbody>
</table>
Overall, the submitted tool show a higher level of maturity than last year’s submissions and are approaching a point where they could be fruitfully deployed in the field.

Some of the high result we observed on the 2012 test for S2 (e.g. bzip2) were not confirmed in more rigorous tests but, it appears, that real progress is being made with respect to deflate/zlib-coded data.
CHAPTER SIX

TEAMS

S1: Naval Postgraduate School:
Simson Garfinkel (team leader)
Bruce Allen
Mike Shick
Joel Young

S2: Digital Forensic Research Center, Korea University:
Jungheum Park, junghmi@korea.ac.kr
Jewan Bang, jwbang@korea.ac.kr
Yunho Lee, dfrc21@korea.ac.kr
Jonghyun Choi, antares@korea.ac.kr

Assisted by Prof. Sangjin Lee sangjin@korea.ac.kr
The first prize in the 2013 DFRWS Forensic Challenge is awarded to the team from DFRC, Korea University. Congratulations.