A fragment classification method depending on data type

Ning Zheng
Computer College. Hangzhou Dianzi University
Hangzhou city, Zhejiang province, China
nzheng@hdu.edu.cn

Jinlong Wang
Computer College. Hangzhou Dianzi University
Hangzhou city, Zhejiang province, China
wzhuce_2011@126.com

Ting Wu
Computer College. Hangzhou Dianzi University
Hangzhou city, Zhejiang province, China
peterwuting@hotmail.com

Ming Xu
Computer College. Hangzhou Dianzi University
Hangzhou city, Zhejiang province, China
mxu@hdu.edu.cn

***Abstract*—Data fragment classification is an important problem in many fields, such as intrusion detection, reverse engineering, data recovery, digital forensics and so on. Most of the existing methods try to classify the fragment depending on file type. But the results are poor, because compound file types can contain many other file types, and some file types use the similar data encoding scheme. In this paper, a classification method depending on data type is promoted. In the method the fragment needed to be classified is given a data type instead of file type. First a fragment set including many common data types is created; then the byte frequency distribution and entropy are extracted as features; after that a classifier is built by using those features in training set and SVM algorithm; last the classifier is used to classify the data fragments. The experiment results show that the accuracy of the proposed method is 88.58%, which achieved a 21.2% growth compared with the traditional way.**

Keywords—Fragment Classification; Data Type; File Type; Support Vector Machine

I. INTRODUCTION

Over a decade, many researchers have focused their attention on data or file fragment classification problem. They are trying to find a way can map a sample chunk of data, such as a disk block, a network packet, to a specific of file type. The purpose of this work is also to classify data/file fragments, but we try to solve this problem by using data type rather than file type.

A. Motivation

Most digital forensic investigator and incident response professional can look at a piece of binary data, such as a disk block, a file with unknown type, or a network packet. Identify what kind of data it carries is very helpful in a variety of tasks, such as diagnosing break-ins, making sense of residual data, decoding memory dumps, reverse-engineering malware, data recovery, and so on[1].

This work can be done by manual examination with a hex editor. But the problem is that manual examination can only be operated by professional, and this kind of work is boring and does not scale. So tools that can automated perform the fragment classification work are needed. Once we have a tool that can successfully classify fragments, it can be used to: statistically sample target during triage [2], improve file carving [3, 4], and provide context for other subsequent forensic processing [1].

B. Current state

Most work on data/file fragment classification is to utility a combination of machine learning techniques and statistical analysis. Data fragments, which usually are divided into a "training set" and a "test set", are represented by feature vectors consisting of the histogram of the byte/word values (unigram/bigram counts) and other statistical measurements. A traditional machine learning algorithm is trained by the training set and a classifier is created. Then the test set is used to measure the classifier and the results are reported.

Most of them use file type (PNG, PPT for example) to label the fragment, which ignores two facts: 1) one kind of file type may contain many different data types. For example, PPT document can hold text, PNG image, embedded object and much metadata. 2) Some kinds of file types have similar data type. For example, the PNG and GZ file types both apply the deflate algorithm to compress the data, so the fragments come from these two kinds of files have the same data type.

In this paper, we try to classify the data fragment based on data type instead of file type. The paper is organized as follows. In Section 2, a brief overview of related work about this area is provided. In Section 3, the definitions about file type and data type are given, and the problem of fragment classification with file type is analyzed. The promoted method of classifying fragment using data type is introduced in Section 4. The experiments results and discussion can be found in Section 5. The conclusion and future work can be found in Section 6.

II. RELATED WORK

Veenman [4] extracted the BFD, Shannon entropy and Kolmogorov complexity measures as the features for his classification approach. He used a 450 MB size of corpus, which including 11 different file formats, and used 4096 bytes as fragment size. The classification accuracy for most file types was quite modest: between 18% for zip and 78% for executables. The average classification accuracy is 45%.

Stefan Axelsson [5, 6, and 7] has written a series papers to describe his file fragment classifier, which uses normalized compression distance (NCD) algorithm in conjunction with the k-nearest-neighbors as the classification algorithm. In the paper [6], he used the freely available corpus of forensics research data by Garfinkei and selected 28 file types in the experiments. Using a 512 bytes fragment size, he got accuracy between 32% and 36% when the k ranged from 1 to 10.

Fitzgerald et al. [8] also used the Garfinkei data corpus to generate file fragments. The file fragments belong to 24 kinds of file types, and are partitioned to a training set and a testing set. He extracted a lot of features from the file fragment, such as 1-gram (unigram), 2-gram (bigram), Shannon entropy, and Hamming weight. They proceeded to train the support vector machine (SVM) on the training set with a linear kernel, and got 47.5% accuracy when they used the trained SVM model to classify the testing set.

Beebe et al. [3] made almost exclusively research on file fragment classification. In their work, they selected 38 file and data types, including several types seldom or not studied in the past. They made a comparative evaluation of various input feature vectors, which include 15 non-n-gram features, unigram and bigram. They adopted SVM algorithm in their experiment and tested different kernel functions and parameters. A 73.4% classification rate was achieved by using the linear function. Besides, they gave a complete confusion matrix and misclassification analyses.

Roussev, V. [1, 9] didn't try to use a machine learning algorithm to build a classifier to class the data fragments. In his paper [1], the authors argued that previous works fail to make the distinction between primitive types and compound data formats, and promote to develop specialized classification method as the only way to produce practical tools. In paper [9], the authors redefined the file fragments classification problem, and planed to build a tree of classifiers – from the most generic to more specific ones. Most importantly, they empirically analyzed the deflate-coded data and write a robust tool to parse deflate-encoded data.

III. FILE TYPE AND DATA TYPE

A. Definition

File type and data type are two different concepts. According to Erbacher and Mulholland's [10] definitions:

"File type": The overall type of file. This is often indicated by the application used to create or access the file.

"Data type": Indicative of the type of file data embedded in a file. For example, while the file type may be a Microsoft Word .doc file, the file may contain text, images, spreadsheets, or tables, etc. Thus a single file type will often incorporate multiple data types.

In [8], the authors introduced primitive type, which is defined as families of homogeneous data with closely related binary structure. Examples of primitive types include: US-ASCII encoded English text, 32-bit x86 protected mode machine code, random number sequences, image bitmaps; essentially any group of similar data, possibly encoded, compressed, or encrypted in the same way. The authors used only 6 primitive types, so it's a coarsely grained category method. For example, they regarded the PNG, JPG, BZIP2 and other file types as one big category in their experiment.

We use Erbacher and Mulholland's file type's definition, and give our own data type's definition. Data type is used to indicative of the type of file data embedded in a file; they are families of homogeneous data with closely related binary structure. Intuitively, data type is one kind of data encoding scheme with special organization form. For example, the deflate data type means the output data's type encoding by deflate compression algorithm.

Logically, the data in a file can be split into two parts – actual data, and structural metadata. The amount of metadata can vary significantly across file formats – text files may have little or none, whereas MS Office documents are, in fact, small file systems. The actual data and metadata usually belong to different data types. In this paper we treat distinctly structured regions as discrete types.

In this paper, file type are written in uppercase letter, and data type written in lower case. For example, "BMP" means the file type and "bmp", which means the data part in BMP files, is a data type.

B. The problem of classification using file type.

Prior work tried to classify a fragment into one kind of file type. They give the fragment a file type label, ignoring the fragment's underlying data type. These methods' results are poor, because they ignored two facts.

1. One file type contains many data types.

In compound file types, one file type can contain many data types. For example, PPT file may contain text, PNG images, JPG images, spreadsheets and so on. So, fragments come from PPT files belong to many different data types. Fitzgerald et al. [8] used SVM algorithm to classify fragments come from 24 file types. In their experiments, the PPT file type's classification accuracy is only 13.6%. The PPT fragments were wrongly classified as PNG, GZ or ZIP file fragments. It's not surprising for PPT files usually contain PNG images or OLE objects which are stored in deflate compressed format.

2. One data type used by many file types.

Many file formats use the same or similar data encoding scheme, so the fragments come from those files are actual have the same data type. This is particularly true for archive and compression formats, as they are rather expensive to code and debug from scratch, and are surrounded by numerous patent

and copyright restrictions [9]. For example, the GZ compressed file type uses the deflate algorithm and zlib storage format, and the PNG image file type and ZIP file type use the deflate algorithm, too. So, the GZ, PNG and ZIP file fragments are confused with each other in Simran et al.'s experiments.

File fragment is just a part of file, which may contain the actual data or metadata. And compound file types may contain other kinds of files. Besides, fragments which come from different kind of file types may belong to one data types. So it's unwise to use file type to classify file or data fragments. Classify the fragments depending on data type is more appropriate and feasible.

IV. PROMOTED METHOD

In this section, we promoted a classification method in which we classify the fragments depending on data types.

Classification is a standard machine learning problem. So we apply support vector machine (SVM) machine learning techniques to the problem of data fragment classification. The method includes five steps:

1. Create a data set including many data types;

2. Divide the data set into two parts, one is training set and another is testing set;

3. Extract features from the fragments so every fragment is represented by a feature vector;

4. Use the training set's feature vectors and SVM algorithm to build a classifier;

5. Use the testing set's data to test the classifier.

The general process is the same with the prior methods, what we different is that we use data type in the experiments.

A. Data types and file types

It needs to analysis the file format to decide which data types it carries. Some file types contain only one kind data type. For example, the XML file type contains xml data type. Some file types may contain many kinds of data types. For example, the PPT file type can contain texts, jpg data type, deflate data type and so on. Some different file types may have the same data type. For example, the PNG file type and GZ file type have the same data type – deflate compressed data type. We try to classify the unknown fragment by data types. So, in the training section, we label the fragments by its data type instead of file type. By this way, the fragments come from the PNG files and GZ files, are labeled as deflate compressed data type. The fragments come from PPT files, may be labeled different data types.

The Fig-1 shows the relationship of 12 file types and 12 data types we use in our experiments. The CSV, XML, RTF, JAVA, BMP, MP3, JPG file types are simple, so each of they contains only one data type. The PNG, ZIP and GZ file format apply the deflate compression algorithm, so they are the deflate data type. The DOC and PPT are compound file types. "wrx" means the text content in word document and "wrc" means the metadata extract from word document's table stream and word document stream. "pdx" means the text content in PPT files and the "pdc" means the metadata extract from PPT files [11]. They both can contain JPG and PNG files, so they contain the jpg and deflate data types. Actually DOC and PPT can contain each other, but this is seldom so we didn't take it into account.

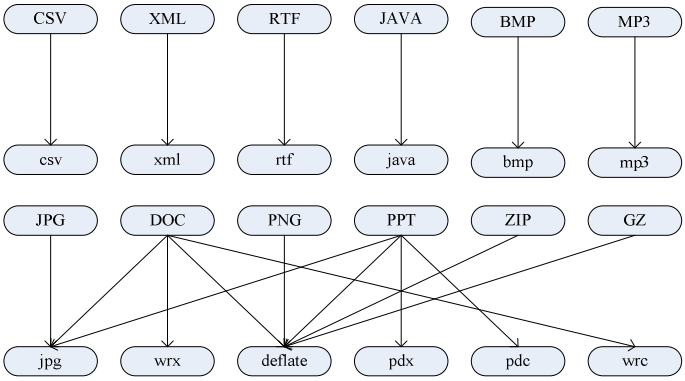

Fig-1 Relationship of 12 file types and 12 data types

B. Data set and feature selection

The data set we used for training and testing is derived from a freely available corpus of forensic data set, which was created by Garfinkel et al. [12]. 12 kinds of file types are selected: CSV, XML, RTF, JAVA, MP3, BMP, PNG, JPEG, DOC, PPT, ZIP and GZ. All files are come from the GovDocs dataset. 12 kinds of data types are derived from those selected file types. The relationship between the file types and data types can be seen in Fig-1. We select 512 bytes as the fragment size, for this is currently the smallest available sector size on magnetic media. When generating the fragments, we omitted the first and last fragments of each file, as the first fragment frequently contains signature, and the last fragment might not be 512 bytes in length. We would have at least 50 files made up of at least 10000 512-byte fragments for the 12 data types.

There are many kinds of features can be extracted from the data fragment, such as byte frequency distribution (BFD, or 1-gram), 2-gram, entropy, mean byte value, and Hamming weight. In this paper, the BFD and entropy are selected, for they are easy to achieve, and they are commonly used by previous workers. The followed experiments results show that these features can give satisfied classification results. While using more features or trying different feature combination may yield better classification results, we didn't try those experiments, for the main purpose of this paper is not to find the most effective feature combination.

C. Support vector machine

A lot of machine learning algorithms can be used to classify fragment type, such as support vector machine (SVM), k-nearest neighbor (k-NN), artificial neural networks (ANN) and so on. In this paper the SVM algorithm is selected. We tried the k-NN and ANN algorithms in our preliminary experiments, and found that it is not as effective as the SVM

algorithm. Besides, the algorithm is used by many pervious researchers, too.

A classification task usually involves separating data into training and testing sets. Each instance in the training set contains one "target value" (the class labels) and several "attributes" (the fragment features). The goal of SVM is to produce a model (based on the training data) which predicts the target values of the test data given only the test data attributes.

Given a training set of instance-label pairs ($\mathbf{x}_i, y_i$), $i = 1,\ldots,l$ where $\mathbf{x}_i \in R^n$ and $\boldsymbol{y} \in \{1, -1\}^l$, the support vector machines require the solution of the following optimization problem:

$$\min \quad \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi_i \tag{1}$$

$$\text{s.t.} \quad y_i (w^T \phi(x_i) + b) \geq 1 - \xi_i \tag{2}$$

$$b, w \text{ unrestricted} \tag{3}$$

$$\xi_i \geq 0 \tag{4}$$

Here training vectors $\mathbf{x}_i$ are mapped into a higher dimensional space by the function ϕ. SVM finds a linear separating hyper plane with the maximal margin in this higher dimensional space. $C > 0$ is the penalty parameter of the error term. Furthermore, $K(x_i, x_j) \equiv \phi(x_i)^T \phi(x_j)$ is called the kernel function. Though new kernels are being proposed by researchers, beginners may find in SVM books the following four basic kernels [13]:

Linear: $$K(x_i, x_j) = x_i^T x_j \tag{5}$$

Polynomial: $$K(x_i, x_j) = (\gamma x_i^T x_j + r)^d, \gamma > 0 \tag{6}$$

Radial basis function (RBF):

$$K(x_i, x_j) = \exp(-\gamma \left\| x_i - x_j \right\|^2), \gamma > 0 \tag{7}$$

Sigmoid: $$K(x_i, x_j) = \tanh(\gamma x_i^T x_j + r) \tag{8}$$

Here, γ r and d are kernel parameters.

It needs to select a kernel function when using the SVM. Usually the linear kernel and radial basis function are used firstly. Fitzgerald et al. [8] and Beebe et al. [3] found that the linear kernel was more effective than the radial basis function (RBF). During some preliminary experiments, we also found that the RBF was not as effective as the linear kernel, which support pervious researchers' findings. So the linear kernel was selected in out experiments.

V. EXPERIMENT RESULT AND DISCUSSION

A. Experiment results

The parameter C of SVM may affect the prediction accuracy. We did some experiments to find the optimal value when use the linear kernel. We tried the C=1, 2, 4 … 512, and the results can be found in Table I and Fig-2. It's clearly a value bigger than 128 is suitable for C. In the later experiments, we used value of 256 for C parameter. The fragment number for every type is 1000 and 10-fold cross-validation is employed.

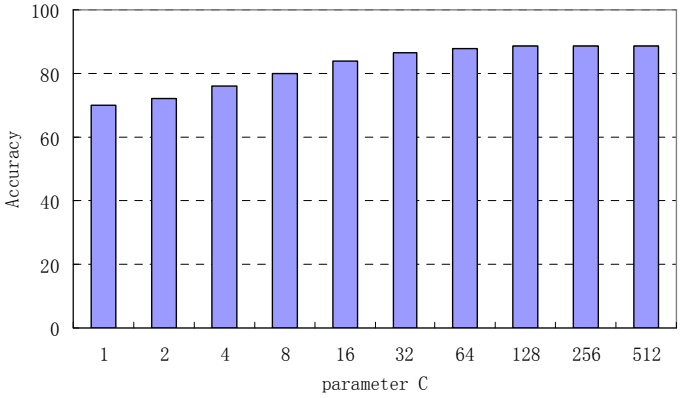

Fig-2 Accuracy by using different C

After running the experiments, we found that our approach produce very good results compared to the file type method. The table II shows the confusion matrix by using data type and table III shows the confusion matrix by using file type ("def" means deflate). For the experiment with file type, the average classification accuracy is 67.38%. The PNG, ZIP and GZ file type's results are poor, for these file types use the same data encoding scheme. The PPT file type's result is very low – only 29.4%. Many PPT fragments are classified as PNG, JPG, GZ or ZIP file types, because PPT document contains many PNG or JPG files, or other OLE objects (using deflate algorithm). The promoted method's average prediction accuracy is 88.58%. The mp3, jpg and deflate data types' accuracy is below the average, for they are high entropy data types, which are a little difficult to classify. The wrc, wrx, pdc and pdx's accuracies are promising; the average of their accuracy is 91.6%. More importantly, they are particularly data types for DOC and PPT file types, so we can infer the fragments' file type when we get their data types.

The promoted method got a 21.2% increase (88.58% VS 67.38%). Further more, the promoted method provides a way to classify compound file type's fragments. It indicates that using data type is more suitable and appropriate than file type to classify file fragments.

B. Discussion

1. Jpg data type's identification

In our experiments, the jpg data type's accuracy is 69.2%, and it is confused with deflate and mp3 data type. Actually we can get a higher accuracy if we use the special feature of jpg format image. In the jpg data, jpg encoders stuff a 0x00 byte after every 0xFF. Besides, there are a few more legal markers that may appear – mostly in the 0xD0 to 0xDB range [9]. So we may distinct jpg fragments and other fragment if we apply those rules.

TABLE I. ACCURACY BY USING DIFFERENT *C*

C	1	2	4	8	16	32	64	128	256	512
accuracy	69.88	72.34	75.95	80.02	83.98	86.69	87.69	88.48	88.58	88.54

TABLE II. CONFUSION MATRIX OF RESULTS BY USING DATA TYPE

	csv	xml	rtf	java	bmp	mp3	jpg	wrx	wrc	def	pdx	pdc
csv	998	0	0	1	0	0	0	1	0	0	0	0
xml	2	937	0	10	0	0	0	51	0	0	0	0
rtf	1	0	968	1	0	0	0	30	0	0	0	0
java	4	21	5	966	0	0	0	4	0	0	0	0
bmp	0	0	0	1	894	3	39	0	31	22	0	10
mp3	0	0	0	0	1	787	78	0	22	112	0	0
jpg	0	5	0	1	22	81	692	0	23	169	3	4
wrx	4	2	3	1	0	0	0	946	13	0	29	2
wrc	0	0	0	0	8	12	8	0	891	2	3	76
def	0	0	0	0	5	109	162	0	0	723	0	1
pdx	0	0	0	0	0	0	0	14	3	0	974	9
pdc	0	0	0	0	15	20	4	4	51	23	29	854

TABLE III. CONFUSION MATRIX OF RESULTS BY USING FILE TYPE

	CSV	XML	RTF	JAVA	BMP	MP3	JPG	DOC	PNG	PPT	ZIP	GZ
CSV	998	0	0	1	0	0	0	1	0	0	0	0
XML	3	948	2	11	0	0	0	36	0	0	0	0
RTF	0	5	978	3	0	0	1	13	0	0	0	0
JAVA	3	17	5	969	0	0	0	6	0	0	0	0
BMP	0	0	0	0	867	6	22	35	7	42	10	11
MP3	0	0	0	0	1	731	61	28	67	21	51	40
JPG	1	6	0	0	15	72	537	21	86	118	100	44
DOC	1	3	0	1	43	42	18	642	95	20	97	38
PNG	0	1	0	0	6	72	59	0	357	77	259	169
PPT	0	0	0	0	32	64	135	32	177	294	150	116
ZIP	0	0	0	0	1	48	80	0	225	51	387	208
GZ	0	0	0	0	3	49	63	0	199	56	252	378

2. A part of a compound file?

Sometimes after classifying the data type of a fragment, we need to decide whether it belongs to a compound file or not. It needs the metadata to identify whether the fragment belong to a compound file or not. The PPT file can contain PNG, JPG pictures and other embedded objects. PPT file is composed of records. The PNG picture is stored in an OfficeArtBStoreContainerFileBlock record and the record type value is 0xF01E [10]. So, if a PNG file head is found and the corresponding record type value (0xF01E) is found before it, too. We can conclude that this PNG file is a part of the PPT compound file. This means the metadata is critical for determine the compound file type.

VI. CONCLUSIONS AND FUTURE WORK

Data fragment classification is an import problem in digital forensics and computer security. Previous workers try to classify a fragment depending on file types, which had two problems: 1) compound file fragment's prediction accuracy is poor for compound file can contain other file types; 2) some different file types' fragments are confused with each other since they use the same data encoding scheme. In this paper, we promoted a method to classify file fragments by using data type instead of file type. We extracted BFD and entropy as the fragment features, and used SVM algorithm to build classifier. The experiments results show that we got 21.2% promotion by using 12 data types and 12 file types.

The results we got are promising, and some directions for future work on this field are apparent. For one, what kind of data can be treated as a data type, which needs more discussion and analysis. Another, some (non-primitive, deflate, for example) data types need more deeply analysis and investigation to infer their related file types. Besides, the analysis of compound documents can help us to infer a fragment belong a compound file or not.

ACKNOWLEDGEMENTS

This work is supported by the Natural Science Foundation Natural Science Foundation of China under Grant No.61070212 and 61003195, the Zhejiang Province Natural Science Foundation Natural Science Foundation of China under Grant No. LY12F02006.

REFERENCES

[1] V. Roussev and C. Quates, File fragment encoding classification—An empirical approach. Digital Investigation, 2013; 10, S69-S77.

[2] S. Garfinke, A. Nelson, D. White, and V. Roussev. Using purpose-built functions and block hashes to enable small block to enable small block and sub-file forensics. Proceedings of the tenth annual DFRWS conference. 2010; 55-70

[3] N. Beebe, L. Maddox, L. Liu, and M. Sun, Sceadan: Using Concatenated N-Gram Vectors for Improved File and Data Type Classification. Information Forensics and Security, IEEE Transactions on, 2013; Vol. 8, 1519-1530.

[4] C.J. Veenman, Statistical disk cluster classification for file carving. Information Assurance and Security. Third International Symposium on. 2007 IEEE; 393-398

[5] S. Axelsson, Using normalized compression distance for classifying file fragments. ARES'10 International Conference on Availability, Reliability, and Security, 2010 IEEE; 641-646.

[6] S. Axelsson, The Normalised Compression Distance as a file fragment classifier. digital investigation, 2010, 7, S24-S31.

[7] S. Axelsson, K.A. Bajwa, and M.V. Srikanth, File Fragment Analysis Using Normalized Compression Distance. Advances in Digital Forensics IX. Springer Berlin Heidelberg, 2011; 171-182.

[8] S. Fitzgerald, G. Mathews, C. Morris, and O. Zhulyn, Using NLP techniques for file fragment classification. Digital Investigation, 2012, 9, S44-S49.

[9] V. Roussev and S.L. Garfinkel, File fragment classification-the case for specialized approaches. Systematic Approaches to Digital Forensic Engineering, 2009. SADFE'09. Fourth International IEEE Workshop on. IEEE. 2009; 3-14

[10] R.F. Erbacher and J. Mulholland, Identification and localization of data types within large-scale file systems. Systematic Approaches to Digital Forensic Engineering, 2007. SADFE 2007. Second International Workshop on. 2007 IEEE. 55-70.

[11] Microsoft Corporation. PowerPoint (.ppt) Binary File Format. http://download.microsoft.com/download/2/4/8/24862317-78F0-4C4B-B355-C7B2C1D997DB/[MS-PPT].pdf

[12] S. Garfinkel, P. Farrell, and V. Roussev, Dinolt, G. Bringing science to digital forensics with standardized forensic corpora. digital investigation, 2009; 6, S2-S11.

[13] C.W. Hsu, C.C. Chang, and C.J. Lin. A practical guide to support vector classification[J]. 2003.