CEUR-WS.org/Vol-2140/paper16.pdf

The Malware Text Collection and Mining Project

Giambattista Amati[*], Simone Angelini[*], Anna Caterina Carli[•], Carlo Majorani[*], and Alessandro Riccardi[*]

[*]Fondazione Ugo Bordoni, Rome, Italy
{gba,sangelini,cmajorani,ariccardi}@fub.it
[•]Istituto Superiore delle Comunicazioni e delle Tecnologie dell'Informazione (ISCOM), Rome, Italy
annacaterina.carli@mise.gov.it

Abstract. We have released a malware collection in TREC style. It contains scripts, html documents and text files extracted from binary files of about 650K malwares. The objective of the project is to index, extract significant features and classify them into malware families. At this aim we will also release a TREC style set of queries for classification tasks. In this abstract, we briefly describe the test collection, the project aims and the problems underlying the use of text mining and information retrieval techniques to malware classification.

1 Introduction

Malware analysis is a growing research area but with still many open problems [1]. For example T signatures for anti-virus toolkits are created manually using some malware-analysis techniques and tools, that can analyze programs either by executing them (*dynamic analysis*), or by inspecting them (*static analysis*). Static analysis can extract information from the binary representation of the program. Data mining techniques for detecting malware were first introduced by [2] on three different static features: Portable Executable (PE), strings and byte sequences. Interpretable text is a high-level specification of malicious behavior, for example: <html><scriptlanguage = 'javascript'> window.open('readme.eml') always occur in worms of type Nimda [3]. Text Mining classification can be useful, and be however prohibitive because of the tokenization process than may either produce a very high dimensionality of features or lose relevant information by the use of a standard text IR tokenization. Nevertheless, Big Data technologies and massive clustering techniques are now possible so that the release of a TREC style collection, that is still missing, will help the IR and the cyber security community to deeply explore at what extent Information Retrieval and Text Mining classification can be effective and useful to malware detection. Our text collection contains about 650K documents with the text extracted from malware and will be extended with a similar size of malware-free collection.

IIR 2018, May 28-30, 2018, Rome, Italy. Copyright held by the author(s).

https://trec.nist.gov/data.html

Collection	Nr Docs	#Tokens	Nr Occurrences	Index Dimensions
MW-TaggedText	655,361	153,587,253	4,222,109,462	21GB

Table 1. Collections that were collected and processed. The VS-TaggedText collection contains the text of subset of the available collection at VirusShare.com and occupies 30GB of malware data.

2 The malware collection

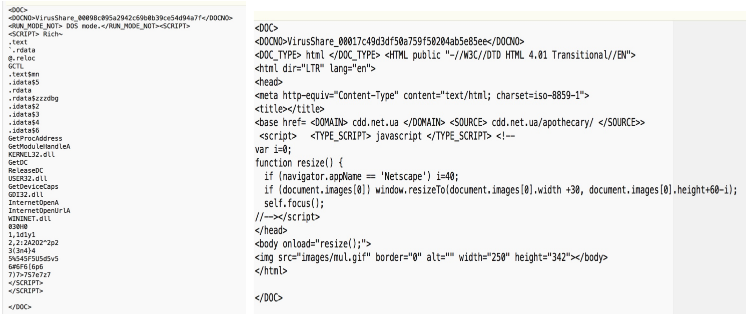
The malware collection was obtained from the VirusShare.com project. VirusShare was born in 2011 with the aim of collecting, indexing and freely sharing malware samples for analysts, researchers and the computer security community. At the moment the site provides about 30 million malwares. We have downloaded a portion containing 655,361 of the most recent malware files (i.e. collected by VirusShare in the last 6 months). Initially the collection was about 286 compressed GB (11 zip archives). We extracted the text part and formed a collection of about 66GB of uncompressed data, or equivalently of about 30GB of compressed data, and obtaining 21 GB of indexes. The text part of the whole collection should therefore contain approximately 14 TeraBytes of compressed data for 9TB of Terrier's indexes.
The malware collection has been subjected to the following operations:

Text extraction The text part was first extracted through the unix script *strings*. From 286GB of compressed data, 30 GB of compressed data were obtained.

Tagging The collection was then labeled by introducing the following new *tags*: DOCNO, DOC_TYPE, SCRIPT, TYPE_SCRIPT, CDATA, DOMAIN, SOURCE, RUN_MODE, RUN_MODE_NOT. The labeling module was obtained through a set of syntactic rules of the regex type. We get all domains and URLs, to index them separately, trasforming strings such as http://xxx.yy.201.53/guodanpi/dhnchia.exe into:
< DOMAIN > xxx.yy.201.53</DOMAIN > and
<SOURCE> xxx.yy.201.53/guodanpi/dhnchia.exe</SOURCE>
The new tags contain the following information:

- DOC: Initial malware tag, and DOCNO, the malware file identifier that contains the MD5 hash value of the file; DOC_TYPE, a tag for a html document or not.
- SCRIPT, the tag that encloses a script, and CDATA, the tag that contains data in a document of markup type.
- SOURCE: a complete URL address, and DOMAIN: an internet domain

https://VirusShare.com
These are Terrier indexes, with both inverted and direct files http://terrier.org.



```
<DOC>
<DOCNO>VirusShare_00098c095a2942c69b0b39ce54d94a7f</DOCNO>
<RUN_MODE_NOT> DOS mode.</RUN_MODE_NOT><SCRIPT>
<SCRIPT> Rich~
.text
`.rdata
@.reloc
GCTL
.text$mn
.idata$5
.rdata
.rdata$zzzdbg
.idata$2
.idata$3
.idata$4
.idata$6
GetProcAddress
GetModuleHandleA
KERNEL32.dll
GetDC
ReleaseDC
USER32.dll
GetDeviceCaps
GDI32.dll
InternetOpenA
InternetOpenUrlA
WININET.dll
030H0
1,1d1y1
2,2:2A2O2^2p2
3(3n4}4
5%545F5U5d5v5
6#6F6[6p6
7)7>7S7e7z7
</SCRIPT>
</SCRIPT>

</DOC>
```

```
<DOC>
<DOCNO>VirusShare_00017c49d3df50a759f50204ab5e85ee</DOCNO>
<DOC_TYPE> html </DOC_TYPE> <HTML public "-//W3C//DTD HTML 4.01 Transitional//EN">
<html dir="LTR" lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title></title>
<base href= <DOMAIN> cdd.net.ua </DOMAIN> <SOURCE> cdd.net.ua/apothecary/ </SOURCE>>
 <script>   <TYPE_SCRIPT> javascript </TYPE_SCRIPT> <!--
var i=0;
function resize() {
  if (navigator.appName == 'Netscape') i=40;
  if (document.images[0]) window.resizeTo(document.images[0].width +30, document.images[0].height+60-i);
  self.focus();
//--></script>
</head>
<body onload="resize();">
<img src="images/mul.gif" border="0" alt="" width="250" height="342"></body>
</html>

</DOC>
```

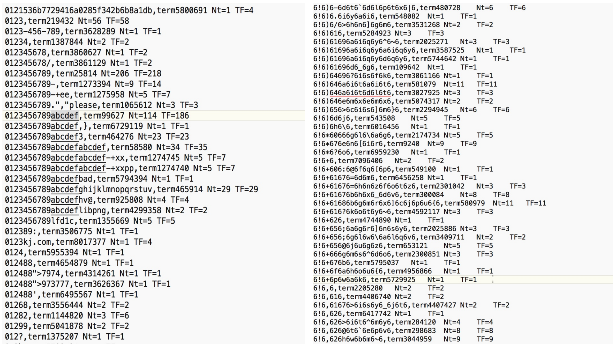
Fig. 1. Examples of a tagged malware script and of a labeled html document.

- TYPE_SCRIPT, a tag with the type of script VBscript, javascript etc., and RUN_MODE, RUN_MODE_NOT, tag for Win32 or DOS etc.

Pre-classification Documents are pre-classified according to their type and script (javascript for example). In Figure 1 there are two examples of processed malware.

Tokenization Finally, the collection was indexed by considering the text within tags, e.g. Html, script etc., and *any sequence of characters* as meaningful tokens. The separator characters between indexed tokens were any blank type character according to the UTF8 encoding. Therefore, the typical punctuation characters (comma, points, etc.) were not considered separators. This extremely loose and permissive indexing makes possible the simultaneous presence in the dictionary as indexed terms (that is, in the lexicon) both words belonging to the natural language of text documents (such as those with the DOC_TYPE tag equal to html) and commands or tokens belonging to scripts (within the SCRIPT tags). 4.2 billion tokens were obtained with 154 million unique terms across the whole collection. The average frequency of terms is 3.58 occurrences per malware, much higher than the average word frequency in natural language texts.

Indexing Thanks to the tagging operations, one can activate or disable any possible tag during the indexing to obtain either dedicated indexes to only scripts (SCRIPT), to only textual parts, to only URL addresses (SOURCE), to only the domain (DOMAIN) or to a combination of these tags. Thanks to the DOC_TYPE and TYPE_SCRIPT fields you can also obtain statistics on the distribution of malware in the different types of documents.



```
0121536b7729416a0285f342b6b8a1db,term5800691 Nt=1 TF=4
0123,term219432 Nt=56 TF=58
0123-456-789,term3628289 Nt=1 TF=1
01234,term1387844 Nt=2 TF=2
012345678,term3860627 Nt=1 TF=2
012345678/,term3861129 Nt=1 TF=2
0123456789,term25814 Nt=206 TF=218
0123456789-,term1273394 Nt=9 TF=14
0123456789-+ee,term1275958 Nt=5 TF=7
0123456789.",\"please,term1065612 Nt=3 TF=3
0123456789abcdef,term99627 Nt=114 TF=186
0123456789abcdef,},term6729119 Nt=1 TF=1
0123456789abcdef3,term464276 Nt=23 TF=23
0123456789abcdefabcdef,term58580 Nt=34 TF=35
0123456789abcdefabcdef-+xx,term1274745 Nt=5 TF=7
0123456789abcdefabcdef-+xxpp,term1274740 Nt=5 TF=7
0123456789abcdefbad,term5794394 Nt=1 TF=1
0123456789abcdefghijklmnopqrstuv,term465914 Nt=29 TF=29
0123456789abcdefhv@,term925808 Nt=4 TF=4
0123456789abcdeflibpng,term4299358 Nt=2 TF=2
0123456789lfd1c,term1355669 Nt=5 TF=5
012389:,term3506775 Nt=1 TF=1
0123kj.com,term8017377 Nt=1 TF=4
0124,term5955394 Nt=1 TF=1
012488,term4654879 Nt=1 TF=1
012488">7974,term4314261 Nt=1 TF=1
012488">973777,term3626367 Nt=1 TF=1
012488',term6495567 Nt=1 TF=1
01268,term3556444 Nt=2 TF=2
01282,term1144820 Nt=3 TF=6
01299,term5041878 Nt=2 TF=2
012?,term1375207 Nt=1 TF=1
```

```
6!6)6-6d6t6`6d6l6p6t6x6|6,term480728    Nt=6    TF=6
6!6)6.6i6y6a6i6,term548082  Nt=1    TF=1
6!6)6/6>6h6n6]6g6m6,term3531268 Nt=2    TF=2
6!6)616,term5284923 Nt=3    TF=3
6!6)61696a6i6q6y6^6~6,term2025271   Nt=3    TF=3
6!6)61696a6i6q6y6a6i6q6y6,term3587525   Nt=1    TF=1
6!6)61696a6i6q6y6d6q6y6,term5744642 Nt=1    TF=1
6!6)61696d6_6g6,term109642  Nt=1    TF=1
6!6)64696?6i6s6f6k6,term3061166 Nt=1    TF=1
6!6)646a6i6t6a6i6t6,term581079  Nt=11   TF=11
6!6)646a6i6t6d6l6t6,term3027925 Nt=3    TF=3
6!6)646e6m6x6e6m6x6,term5074317 Nt=2    TF=2
6!6)656>6c6i6s6]6m6}6,term2294945   Nt=6    TF=6
6!6)6d6j6,term543508    Nt=5    TF=5
6!6)6h6\6,term6016456   Nt=1    TF=1
6!6*60666g6l6\6a6g6,term2174734 Nt=5    TF=5
6!6*676e6n6[6i6r6,term9240  Nt=9    TF=9
6!6*6?6o6,term6959230   Nt=1    TF=1
6!6+6,term7096406   Nt=2    TF=2
6!6+606:6@6f6q6[6p6,term549100  Nt=1    TF=1
6!6+61676=6d6m6,term6456258 Nt=1    TF=1
6!6+61676=6h6n6z6f6o6t6z6,term2301042   Nt=3    TF=3
6!6+61676b6h6x6_6d6v6,term300084    Nt=8    TF=8
6!6+61686b6g6m6r6x6]6c6j6p6u6{6,term580979  Nt=11   TF=11
6!6+616?6k6o6t6y6~6,term4592117 Nt=3    TF=3
6!6+626,term4744890 Nt=1    TF=1
6!6+656;6a6g6r6]6n6s6y6,term2025886 Nt=3    TF=3
6!6+656;6g6l6w6\6a6l6q6v6,term3409711   Nt=2    TF=2
6!6+656@6j6u6g6z6,term653121    Nt=5    TF=5
6!6+666g6m6s6^6d6o6,term2300851 Nt=3    TF=3
6!6+676b6,term5795037   Nt=1    TF=1
6!6+6f6a6h6o6u6{6,term4956866   Nt=1    TF=1
6!6+6p6w6a6k6,term5729925   Nt=1    TF=1
6!6,6,term2205280   Nt=2    TF=2
6!6,616,term4406740 Nt=2    TF=2
6!6,61676>6i6s6y6_6j6t6,term4407427 Nt=2    TF=2
6!6,626,term6417742 Nt=1    TF=1
6!6,626>6i6t6^6m6y6,term284120  Nt=4    TF=4
6!6,626@6t6`6e6p6v6,term298683  Nt=8    TF=8
6!6,626h6w6b6m6~6,term3044959   Nt=9    TF=9
```

Fig. 2. Parts of a lexicon extracted and labeled in the textual part of a malware. Nt denotes in how many files the token occurs, TF is how many times it occurs. Some tokens appear to be related to the Windows PE executable file with encoding in Base64.. For example, the sequence 6!6,626@6t6‘6e6p6v6 is present 8 times in 8 malware. term298683 instead indicates the entire coding of the term in the system. The phenomenon of obfuscation is evident. The strings in the figure are all generated by the regex "(d.)+d" where d is the digit 6.

3 Classification tasks

Malware classification is a hard task because very few labeled training sets exist for detection. Clustering can be an alternative because it can automatically aggregate malware into different groups. However, the very first step forward for a classification task would be to separate malware files from non-malware ones in a very large document collection. We have released a TREC like collection of malware text that was still missing. This collection allows for researchers from different areas to cooperate and apply IR, Data Mining and Big Data technologies to the problem of malware classification. At this aim a set of classification queries will be soon released.

References

1. Egele, M., Scholte, T., Kirda, E., and Kruegel, C. A survey on automated dynamic malware-analysis techniques and tools. *ACM Comput. Surv. 44*, 2 (Mar. 2008), 6:1–6:42.
2. Schultz, M. G., Eskin, E., Zadok, E., and Stolfo, S. J. Data mining methods for detection of new malicious executables. In *Proceedings of the 2001 IEEE Symposium on Security and Privacy* (Washington, DC, USA, 2001), SP ’01, IEEE Computer Society, pp. 38–.
3. Ye, Y., Li, T., Adjeroh, D., and Iyengar, S. S. A survey on malware detection using data mining techniques. *ACM Comput. Surv. 50*, 3 (June 2017), 41:1–41:40.