Automatic Extraction of Malicious Behaviors

Khanh-Huu-The Dam
University Paris Diderot and LIPN

Tayssir Touili
LIPN, CNRS and University Paris 13
Motivation

• Symantec reported:

317M malwares in 2014 vs. 431M malwares in 2015

Increased by 36% in one year

More than 1M new malwares released everyday

Malware detection is a big challenge.
Malicious Behavior Extraction

• Extracting malicious behaviors requires a huge amount of engineering effort.
  – a tedious and manual study of the code.
  – a huge time for that study.

The main challenge is how to make this step automatically.
Our goal is …

To extract *automatically* the malicious behaviors!
Model Malicious Behaviors

How?

How does a malicious behavior look like!!

What is a good model for a malicious behavior??
Trojan Downloader

*This code is extracted from Trojan-Downloader.Win32.Delf.abk*
Trojan Downloader

GetSystemDirectoryA
URLDownloadToFileA
WinExec

Malicious API graph
Executing this file in the system folder.

How to extract such graph automatically!!!
Modeling a program

An API call graph represents the order of execution of the different API functions in a program.

*An assembly code of Trojan-Downloader.Win32.Delf.abk*
Our goal is to extract such malicious behavior from this graph.

*An assembly code of Trojan-Downloader.Win32.Delf.abk*
How to extract malicious behaviors?

Our goal: Isolate the few relevant subgraphs (in malwares) from the nonrelevant ones (in benwares).
IR Problem vs. Our Problem

**IR Problem**

Retrieve relevant documents and reject nonrelevant ones in a collection of documents.

**Our Problem**

Isolate the few relevant subgraphs (in malwares) from the nonrelevant ones (in benwares).
Information Retrieval Community

- Extensively studied the problem over the past 35 years.

- Information Retrieval (IR) consists of retrieving documents with relevant information from a collection of documents.
  - Web search, email search, etc.

- Several techniques that were proven to be efficient.
Our goal is …

Adapt and apply this knowledge and experience of the IR community to our malicious behavior extraction problem.
Information Retrieval

• Information retrieval research has focused on the retrieval of text documents and images.
  – based on extracting from each document a set of terms that allow to distinguish this document from the other documents in the collection.
  – measure the relevance of a term in a document by a term weight scheme.
Term weight scheme in IR

• The term weight represents the relevance of a term in a document.
  – The higher the term weight is, the more relevant the term is in the document.

• A large number of weighting functions have been investigated.
  – The TFIDF scheme is the most popular term weighting in the IR community.
Basic TFIDF scheme

- The TFIDF term weight is measured from the occurrences of terms in a document and their appearances in other documents.

\[ w(i,j) = \text{tf}(i,j) \times \text{idf}(i) \]

- \( w(i,j) \) : the weight of term \( i \) in document \( j \).
- \( \text{tf}(i,j) \) : the frequency of term \( i \) in document \( j \).
- \( \text{idf}(i) \) : the inverse document frequency of term \( i \).

\[ \text{idf}(i) = \log \left( \frac{N}{\text{df}(i)} \right) \]

\( N \) is the size of the collection.
\( \text{df}(i) \) is the number of documents containing term \( i \).
Properties of TFIDF scheme

- A term is **relevant to a document** if it **occurs frequently** in this document and **rarely appears** in other documents.
  - Words are terms in a document.
  - Common words like “the”, “a”, “with”, “of”, etc. are terms that can be found in every document are irrelevant.
Basic TFIDF Scheme Issues

- Term frequencies are usually bigger for longer documents.
- For ranking, a document with a higher tf for a relevant term is not placed ahead of other documents which have multiple relevant terms.

\[ w(i,j) = F(\text{tf}(i,j)) \times \text{idf}(i) \]

- Adjust the term frequency by a function \( F(\text{tf}) \)

\( F(\text{tf}) \) takes into account the long document normalization and ensures the high rank for relevant documents.
Some Functions of Term frequency

\[ F_1 = \text{tf}(i, j) \]

\[ F_2 = \frac{(k_1 + 1) \times \text{tf}(i, j)}{S(j)} \]

Depending on the application, one function can be better than the others.

\[ F_4 = \begin{cases} 
\frac{(k_1 + 1)}{k_1 \left( \frac{S(j)}{\text{AVG}(D)} \times b + 1 - b \right) + e^{-\text{tf}(i, j)}} & \text{if } \text{tf}(i, j) > 0 \\
0 & \text{if } \text{tf}(i, j) = 0 
\end{cases} \]
How to apply to our graphs?

The relevant graph consists of relevant nodes and edges.
How to apply to our graphs?

- Weight of term (node or edge) \( i \) in graph \( j \) is computed by

\[
w(i,j) = F(\ tf(i,j)) \times idf(i)
\]

- \( tf(i,j) \): the frequency of term \( i \) in graph \( j \).
- \( idf(i) \): the inverse graph frequency of term \( i \).

\[
idf(i) = \log(\ N/df(i))
\]

\( N \) is the size of the collection.
\( df(i) \) is the number of graphs containing term \( i \).
Relevance of a term in a graph

Malware graph set M

API call graphs

Graph \( m_1, m_2, \ldots \)

Relevance ?

Given term \( i \) (node or edge)

\[
w(i, m_j) = F\left(\text{tf}(i, m_j)\right) \times \text{idf}(i)
\]

Benware graph set B

API call graphs

Graph \( b_1, b_2, \ldots \)

Relevance ?

Given term \( i \) (node or edge)

\[
w(i, b_j) = F\left(\text{tf}(i, b_j)\right) \times \text{idf}(i)
\]

Relevance of term \( i \) to graph \( m_j \)

Relevance of term \( i \) to graph \( b_j \)
Relevance of a term in a set

Malware graph set M

Benware graph set B

API call graphs

Graph

$W(i, M) = \frac{1}{K} \sum_{j=1}^{M} w(i, m_j)$

Relevance of term $i$ in Malware

$K = \max_{i,j} w(i, m_j)$

Relevance ?

Given term i (node or edge)

$W(i, B) = \frac{1}{K} \sum_{j=1}^{B} w(i, b_j)$

of term i in Benware

$K = \max_{i,j} w(i, b_j)$
Relevance of a term w.r.t M and B

\[ W(i, M, B) \] is high when \[ W(i, M) \] is high and \[ W(i, B) \] is low.

\[ K = \max_{i,j} w(i, m_j) \]

Given term \( i \) (node or edge)

\[ W(i, B) = \frac{1}{K} \sum_{j=1}^{\|B\|} w(i, b_j) \]

\[ K = \max_{i,j} w(i, b_j) \]
Relevance of a term w.r.t $M$ and $B$: 
**Rocchio weight**

- Measured by the distance between the weight of $i$ in the set $M$ and its weight in the set $B$.

\[
W(i, M, B) = 0.75 \frac{W(i, M)}{|M|} - 0.15 \frac{W(i, B)}{|B|}
\]

- $W(i, M, B)$ is high if $W(i, M)$ is high and $W(i, B)$ is low.
Relevance of a term w.r.t $M$ and $B$: Ratio weight

- Measured by the ratio of the weight of term $i$ in $M$ and its weight in $B$.  

$$W(i, M, B) = \frac{W(i, M)}{|M|} \times \frac{0.5 + |B|}{0.5 + W(i, B)}$$

- This is a kind of quotient between $W(i, M)$ and $W(i, B)$.

- $W(i, M, B)$ is high if $W(i, M)$ is high and $W(i, B)$ is low.

Normalizing term weights by the size of the collection.  

$W(i, B) = 0$.  
To avoid a problem in case
Relevance of a term w.r.t M and B

Malware graph set M

Benware graph set B

How to use the term weight to extract malicious graphs?

$W(i, M, B)$

For each term (node or edge) $i$

The high weight means term $i$ is relevant to $M$ and not to $B$. 
Construct malicious API graphs

• A malicious API graph consists of nodes and edges with the highest weight.

• How to link all these nodes and edges in a graph.

There are different possibilities for computing such graph.
Strategy S0

• Take **n nodes** with the highest weight, for n given by the user.

• Choose **out-going edges** with the highest weight to connect these nodes.

![Diagram](image)

Graphs:
- **A** connected to **B**
- **A** connected to **C**

Nodes with the highest weight:
- **A**

Edges with the highest weight:
- From **A** to **B**
- From **A** to **C**

Edges connecting nodes:
- From **B** to **C**

n = 3
Strategy S0

• Take n nodes with the highest weight, for n given by the user.
• Choose out-going edges with the highest weight to connect these nodes.
Strategy S0

• Take \textit{n nodes} with the highest weight, for \textit{n} given by the user.

• Choose \textit{out-going edges} with the highest weight to connect these nodes.

\begin{tikzpicture}
  \node[circle,fill=red] (A) at (0,0) {A};
  \node[circle,fill=red] (B) at (2,1) {B};
  \node[circle,fill=red] (C) at (1,-1) {C};
  \path[->,red,thick] (A) edge (B);
  \path[->,red,thick] (B) edge (C);
  \path[->,red,thick] (C) edge (A);
  \node at (2,2) {Nodes with the highest weight};
  \node at (2,1) {Edges with the highest weight};
  \node at (1,-2) {Edges connecting nodes};
  \node at (0,-2) {n = 3};
\end{tikzpicture}
Strategy S1

• Take **n nodes** with the highest weight, for **n** given by the user.

• Choose **edges** with the highest weight that start from one of these nodes.

![Graphs Diagram](image)

- **Nodes in the graph**
- **Nodes with the highest weight**
- **Edges with the highest weight**
- **Edges connecting nodes**

$n = 3$
Strategy S1

• Take \textbf{n nodes} with the highest weight, for \( n \) given by the user.

• Choose \textbf{edges} with the highest weight that start from one of these nodes.

\[ \text{Graphs} \]
\[ n = 3 \]

\begin{itemize}
  \item \textbf{Nodes in the graph:} A, B, C, D
  \item \textbf{Nodes with the highest weight:} A
  \item \textbf{Edges with the highest weight:} A to B, D to C
  \item \textbf{Edges connecting nodes:} A to D
\end{itemize}
Strategy S1

- Take **n nodes** with the highest weight, for n given by the user.
- Choose **edges** with the highest weight that start from one of these nodes.
Strategy S2

- Take **n nodes** with the highest weight, for **n** given by the user.
- Choose **paths** with the highest weight to connect each pairs of these nodes.

```
Graphs

A  D  B

A  D  C

n = 3
```

- Nodes in the graph
- Nodes with the highest weight
- Edges on the path with the highest weight
- Edges connecting nodes
Strategy S3

- Take \( n \) edges with the highest weight, for \( n \) given by the user.

![Graph Diagram]

- Nodes in the graph
- Nodes with the highest weight
- Edges with the highest weight
- Edges connecting nodes

\( n = 3 \)
We use these weights to compute malicious graphs by using different strategies.

For each term (node or edge) i

$W(i, M, B)$

The higher weight means term i is relevant to M and not to B.
Does the program contain any malicious behavior?

A new program API call graph

Check common paths

Malware

Benware

How our graphs can be used for malware detection?

Yes

No
Experiments

• Apply on a dataset of 1980 benign programs and 3980 malwares collected from Vx Heaven.
  – Training set consists of 1000 benwares and 2420 malwares → extract malicious graphs.
  – Test set consists of 980 benwares and 1560 malwares → for evaluating malicious graphs.

• Evaluate different strategies and formulas.
Performance Measurement

• High recall means that most of the relevant items were computed.

\[
\text{Recall} = \frac{\text{True Positives}}{\text{Number of graphs}}
\]

(Detection rate)

• High precision means that the technique correctly identifies relevant items.

\[
\text{Precision} = \frac{\text{True Positives}}{\text{True Positives + False Positives}}
\]
Performance Measurement

- **F-Measure** is a harmonic mean of precision and recall.

\[
F \text{ Measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}
\]

- F-Measure is 1 if all retrieved items are relevant and all relevant items have been retrieved.
Evaluating the performance of the different strategies

<table>
<thead>
<tr>
<th>Strategy</th>
<th>n</th>
<th>Recall</th>
<th>Precision</th>
<th>F-Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0 by F3 with Ratio equation</td>
<td>95</td>
<td>89.68%</td>
<td>99.08%</td>
<td>94.15%</td>
</tr>
<tr>
<td>S0 by F3 with Rocchio equation</td>
<td>85</td>
<td>99.04%</td>
<td>98.16%</td>
<td>98.6%</td>
</tr>
<tr>
<td>S1 by F4 with Ratio equation</td>
<td>65</td>
<td>98.14%</td>
<td>98.71%</td>
<td>98.42%</td>
</tr>
<tr>
<td>S1 by F4 with Rocchio equation</td>
<td>40</td>
<td>96.6%</td>
<td>97.98%</td>
<td>97.3%</td>
</tr>
<tr>
<td>S2 by F3 with Ratio equation</td>
<td>65</td>
<td>78.24%</td>
<td>99.35%</td>
<td>87.56%</td>
</tr>
<tr>
<td>S2 by F3 with Rocchio equation</td>
<td>80</td>
<td>95.77%</td>
<td>97.84%</td>
<td>96.8%</td>
</tr>
<tr>
<td>S3 by F3 with Ratio equation</td>
<td>95</td>
<td>82.56%</td>
<td>99.92%</td>
<td>90.42%</td>
</tr>
<tr>
<td>S3 by F1 with Rocchio equation</td>
<td>90</td>
<td>80.38%</td>
<td>98.35%</td>
<td>88.47%</td>
</tr>
</tbody>
</table>

The best performance of each strategy.
Evaluating the performance of the different strategies

<table>
<thead>
<tr>
<th>Strategy</th>
<th>n</th>
<th>Recall</th>
<th>Precision</th>
<th>F-Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0 by F3 with Ratio equation</td>
<td>95</td>
<td>89.68%</td>
<td>99.08%</td>
<td>94.15%</td>
</tr>
<tr>
<td>S0 by F3 with Rocchio equation</td>
<td>85</td>
<td>99.04%</td>
<td>98.16%</td>
<td>98.6%</td>
</tr>
<tr>
<td>S1 by F4 with Ratio equation</td>
<td>65</td>
<td>98.14%</td>
<td>98.71%</td>
<td>98.42%</td>
</tr>
<tr>
<td>S1 by F4 with Rocchio equation</td>
<td>40</td>
<td>96.6%</td>
<td>97.98%</td>
<td>97.3%</td>
</tr>
<tr>
<td>S2 by F3 with Ratio equation</td>
<td>65</td>
<td>78.24%</td>
<td>99.35%</td>
<td>87.56%</td>
</tr>
<tr>
<td>S2 by F3 with Rocchio equation</td>
<td>80</td>
<td>95.77%</td>
<td>97.84%</td>
<td>96.8%</td>
</tr>
<tr>
<td>S3 by F3 with Ratio equation</td>
<td>95</td>
<td>82.56%</td>
<td>99.92%</td>
<td>90.42%</td>
</tr>
<tr>
<td>S3 by F1 with Rocchio equation</td>
<td>90</td>
<td>80.38%</td>
<td>98.35%</td>
<td>88.47%</td>
</tr>
</tbody>
</table>

The best performance is the Rocchio equation, strategy S0, formula F3, and n = 85.
Comparison with well-known antiviruses

• Detect new unknown malwares
  
  – 180 new malwares generated by NGVCK, RCWG and VCL32 which are the best known virus generators.

  – 32 new malwares from Internet*.

* https://malwr.com/
Comparison with well-known antiviruses

<table>
<thead>
<tr>
<th>Antivirus</th>
<th>New malwares from Internet</th>
<th>New generated malwares</th>
<th>Antivirus</th>
<th>New malwares from Internet</th>
<th>New generated malwares</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our tool</td>
<td>100%</td>
<td>100%</td>
<td>Panda</td>
<td>25%</td>
<td>19%</td>
</tr>
<tr>
<td>Avira</td>
<td>50%</td>
<td>16%</td>
<td>Kaspersky</td>
<td>35%</td>
<td>81%</td>
</tr>
<tr>
<td>Avast</td>
<td>45%</td>
<td>87%</td>
<td>Qihoo-360</td>
<td>80%</td>
<td>96%</td>
</tr>
<tr>
<td>McAfee</td>
<td>40%</td>
<td>96%</td>
<td>AVG</td>
<td>40%</td>
<td>82%</td>
</tr>
<tr>
<td>BitDefender</td>
<td>40%</td>
<td>87%</td>
<td>ESET-NOD32</td>
<td>65%</td>
<td>87%</td>
</tr>
<tr>
<td>F-Secure</td>
<td>40%</td>
<td>87%</td>
<td>Symantec</td>
<td>40%</td>
<td>14%</td>
</tr>
</tbody>
</table>

A comparison of our method against well-known antiviruses.
Summary

• Apply TFIDF scheme for extracting automatically malicious behaviors from the collection of malwares and benwares.

• Compare different formulas and strategies.

• Detection rate is 99.04 %.

• Our tool is able to detect malwares that well-known antiviruses could not detect.
Thank you!