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

Trans
Inte
st
syste
exe

```
n15  push   0FEh
n16  push   offset dword_4097A4
n17  call   GetSystemDirectoryA
n18  push   0
n19  push   0
n20  lea    eax, [ebp-1Ch]
n21  mov    ebx, eax
n22  push   ebx
n23  push   eax
n24  push   0
n25  call   URLDownloadToFileA
n26  push   5
n27  call   sub_4038B4
n28  push   ebx
n29  call   WinExec
```

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

Trojan Downloader

How to extract such graph automatically!!!

Get the p
folder.

GetSystemDirectoryA

URLDownloadToFileA

Transfer data from an URL address into a file.

WinExec

Malicious API graph

Executing this file in the system folder.

```
push    0FEh
push    offset dword_4097A4
call    GetSystemDirectoryA
push    0
push    0
lea     eax, [ebp-1Ch]
mov     ebx, eax
push    ebx
push    eax
push    0
call    URLDownloadToFileA
push    5
call    sub_4038B4
push    ebx
call    WinExec
```

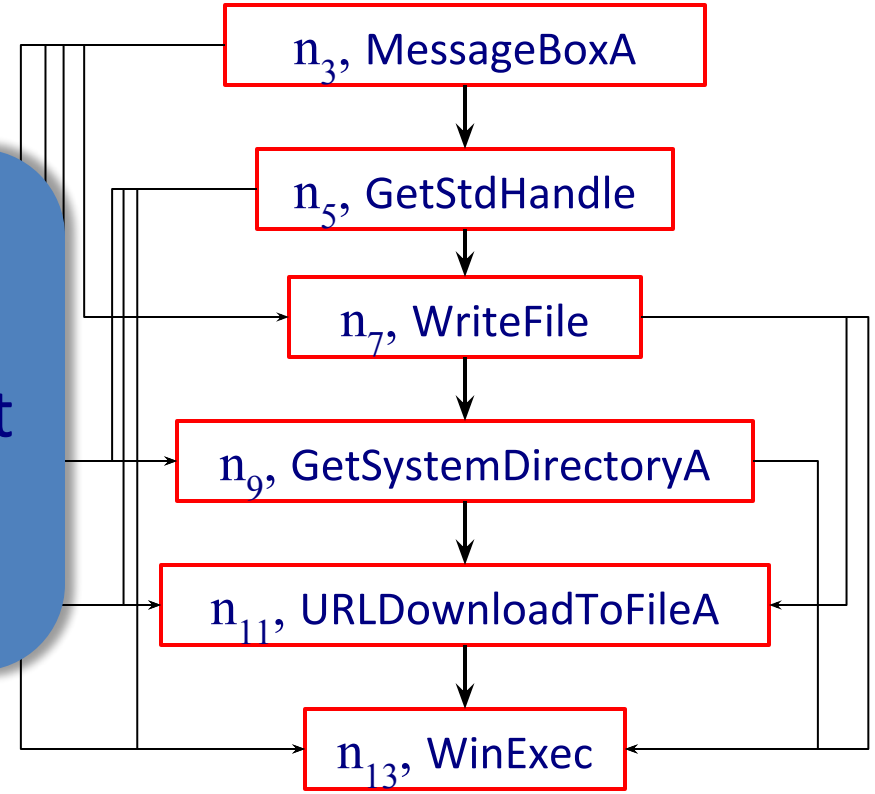
Modeling a program

…
n_1 push offset Text
n_2 push 0

…
n_{12} push ebx
n_{13} call **WinExec**

*An assembly code of Trojan-Downloader.Win32.Delf.abk

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



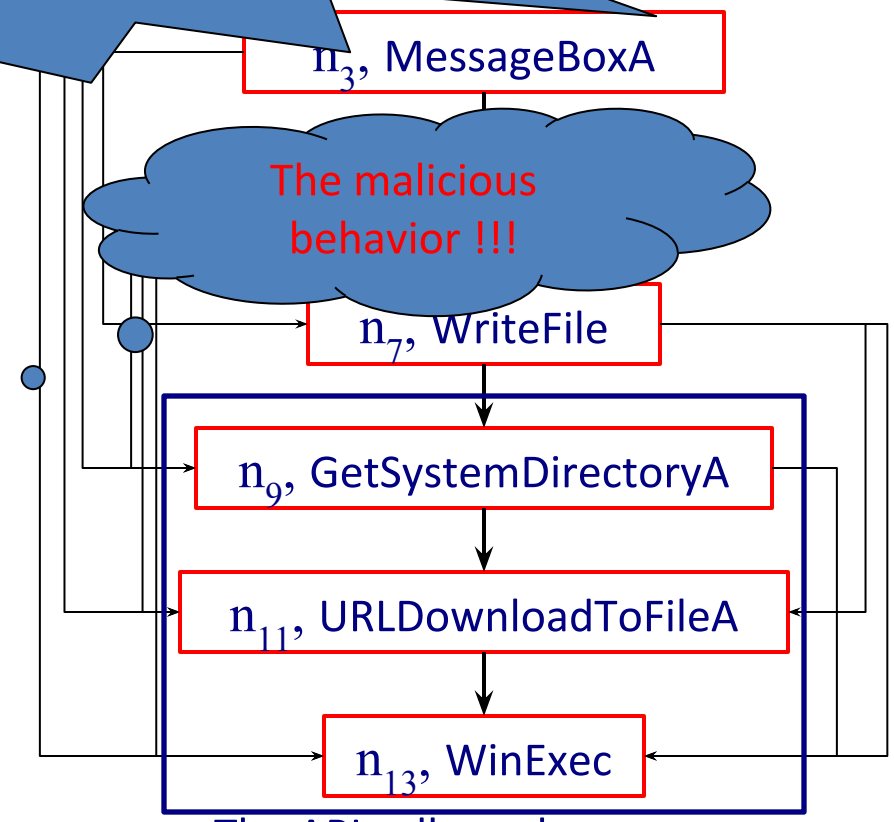
The API call graph

Modeling a program

Our goal is to extract such malicious behavior from this graph.

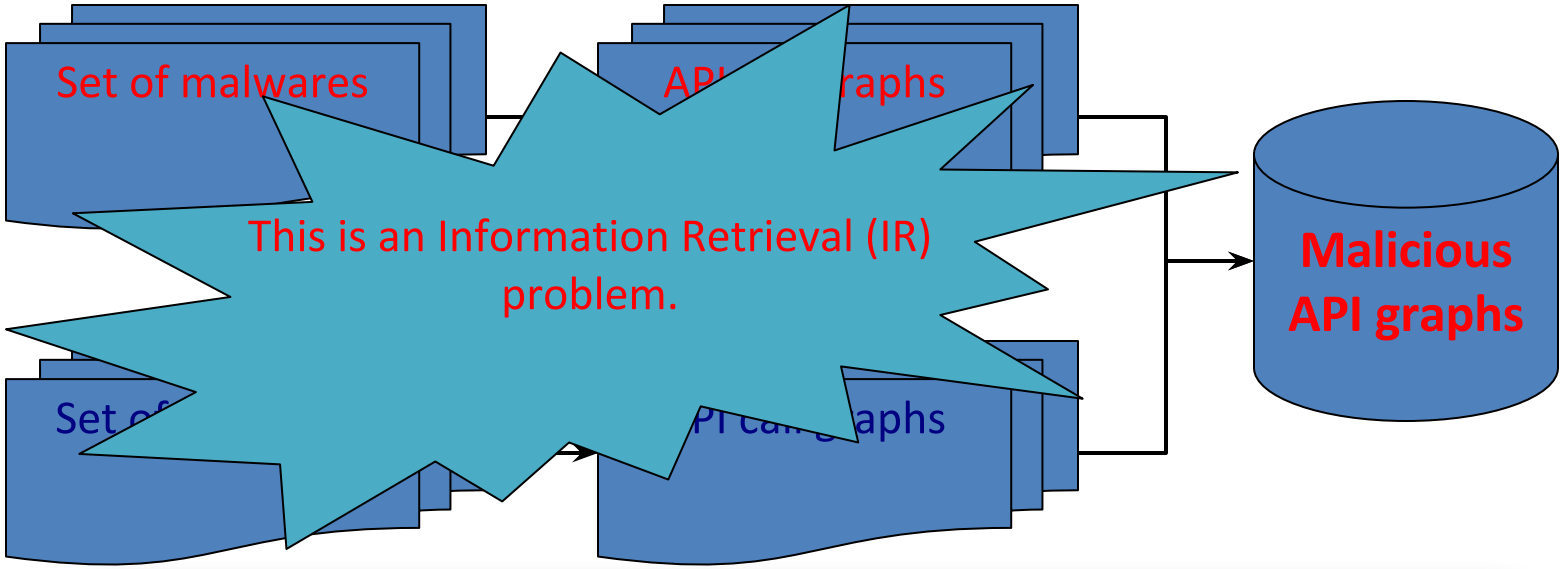
```
…
n1   push   offset Te
n2   push   0
n3   call   MessageBoxA
…
n4   push   0FFFFFFF5h
n5   call   GetStdHandle
n6   push   eax
n7   call   WriteFile
…
n8   push   offset dword_4097A4
n9   call   GetSystemDirectoryA
…
n10  push   0
n11  call   URLDownloadToFileA
…
n12  push   ebx
n13  call   WinExec
```

*An assembly code of Trojan-Downloader.Win32.Delf.abk



The API call graph

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 <u>set of terms</u> that allow to distinguish this document from the other documents in the collection.
  - measure the <u>relevance of a term</u> 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.

$$w(i,j) = \mathrm{tf}(i,j) \times \mathrm{idf}(i)$$

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

    $\mathrm{idf}(i) = \log(N/\mathrm{df}(i))$

N is the size of the collection.

$\mathrm{df}(i)$ is the number of documents containing term i.

Properties of TFIDF scheme

- A term is <u>relevant to a document</u> 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.

$$w(i,j) = F(\mathrm{tf}(i,j)) \times \mathrm{idf}(i)$$

- Adjust the term frequency by a function F

$F(\mathrm{tf})$ takes into account the long document normalization and ensures the high rank for relevant documents.

Some Functions of Term frequency

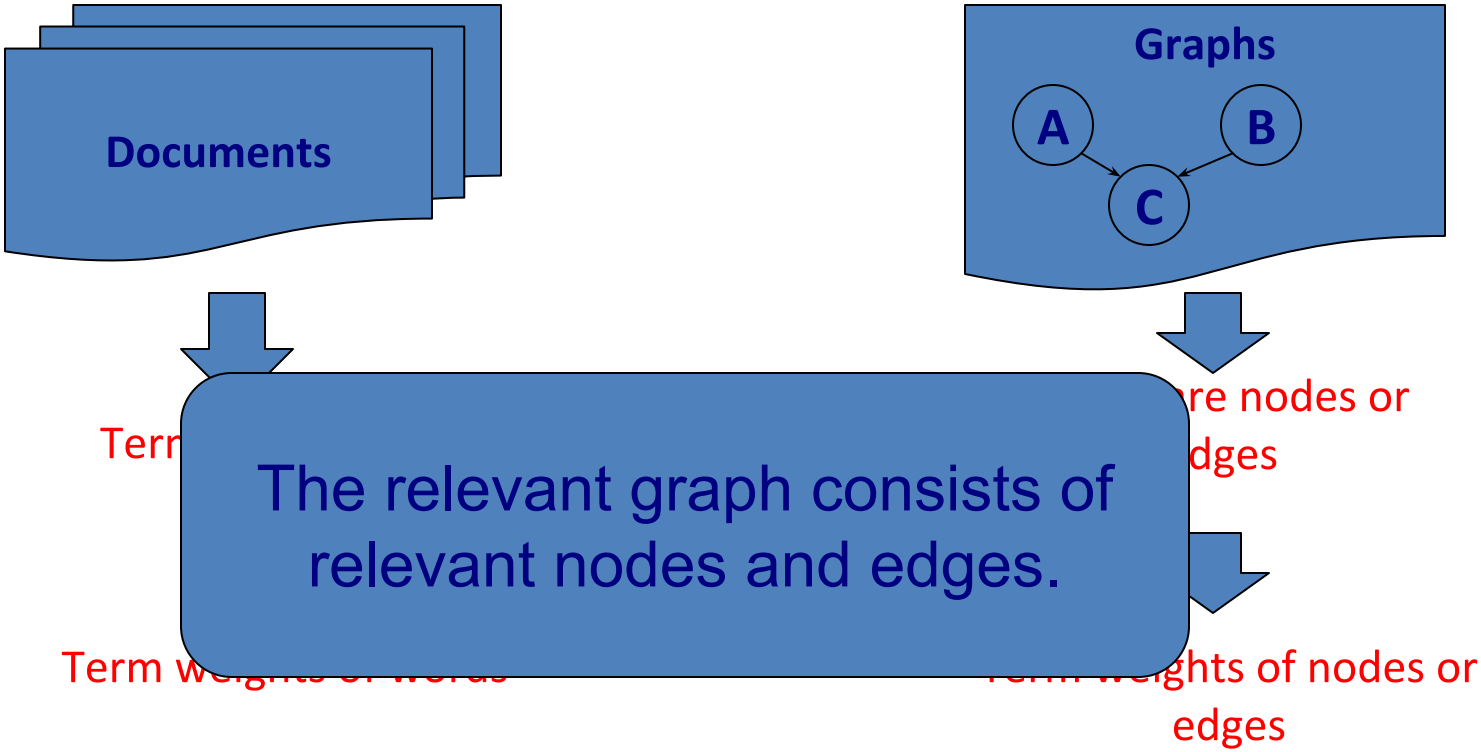
$$F_1 = \mathrm{tf}(i,j)$$

$$F_2 = \frac{(k_1 + 1) \times \mathrm{tf}(i,j)}{S(j)}$$

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

$$F_4 = \begin{cases} \dfrac{(k_1 + 1)}{k_1\left(\dfrac{S(j)}{\mathrm{AVG}(\mathcal{D})} \times b + 1 - b\right) + e^{-\mathrm{tf}(i,j)}} & \text{if } \mathrm{tf}(i,j) > 0 \\ 0 & \text{if } \mathrm{tf}(i,j) = 0 \end{cases}$$

How to apply to our graphs ?



How to apply to our graphs ?

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

$$w(i,j) = F(\mathrm{tf}(i,j)) \times \mathrm{idf}(i)$$

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

$$\mathrm{idf}(i) = \log(N/\mathrm{df}(i))$$

N is the size of the collection.

$\mathrm{df}(i)$ 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, …

Benware graph set B

API call graphs

Graph b_1, b_2 …

Relevance ?

$$w(i, m_j) = F\left(tf(i, m_j)\right) \times idf(i)$$

Releva
term **i** to graph
m_j

$$w(i, b_j) = F\left(tf(i, b_j)\right) \times idf(i)$$

f
term **i** to graph
b_j

Given term i (node or edge)

Relevance of a term in a set

Malware graph set M

API call graphs

Graph

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

Relevance ?

Benware graph set B

API call graphs

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

Relevance of term **i** in Malwares

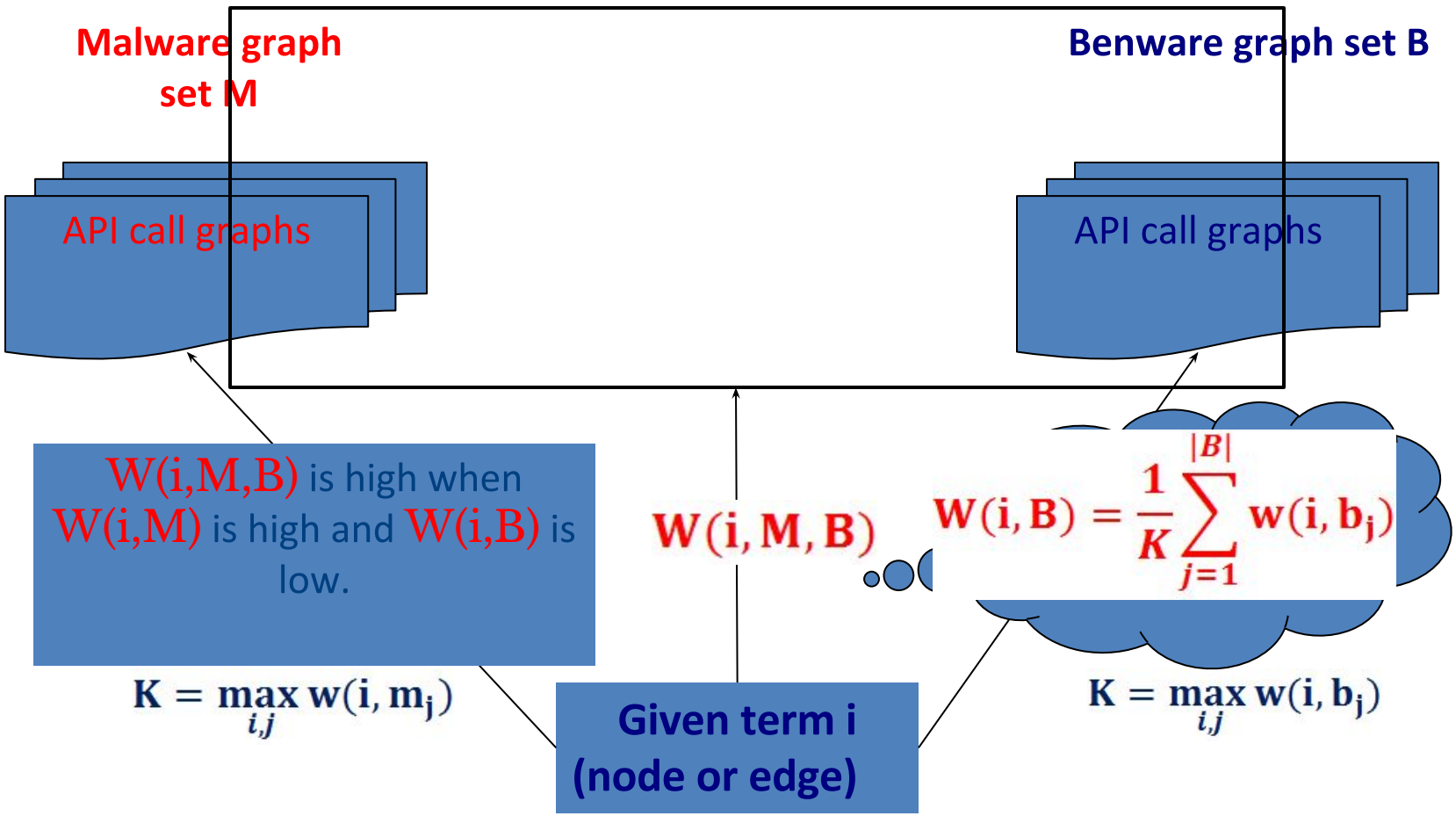
$$K = \max_{i,j} w(i, m_j)$$

Given term i (node or edge)

of term **i** in

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

Relevance of a term w.r.t M and B



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.

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

Values to adjust the effect of term weights in **M** and in **B**.

Normalizing term weights by the size of the collection.

- $\mathrm{W(i,M,B)}$ is high if $\mathrm{W(i,M)}$ is high and $\mathrm{W(i,B)}$ 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.

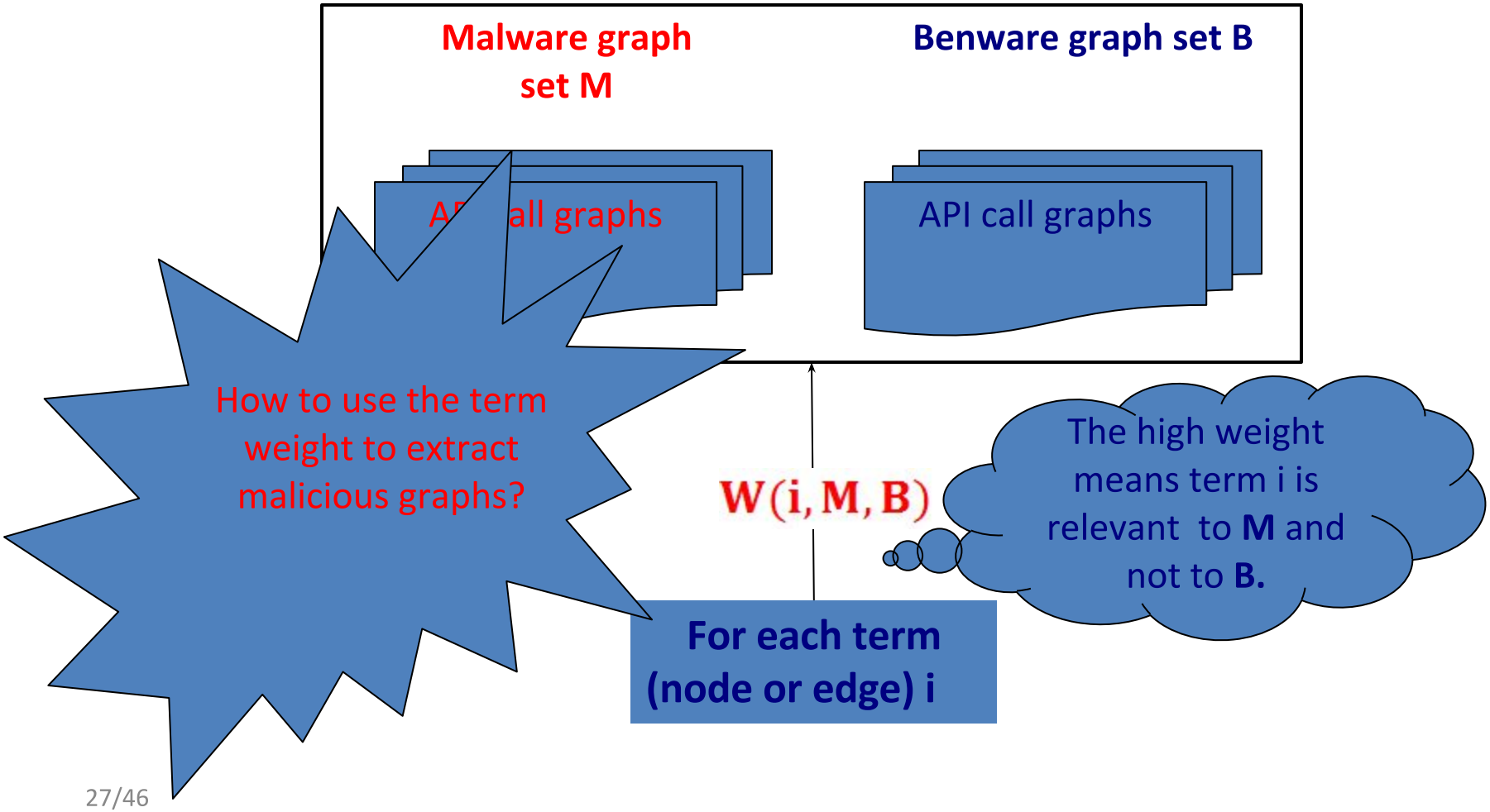
$$\mathbf{W(i,M,B)} = \frac{\mathbf{W(i,M)}}{|M|} \times \frac{\mathbf{0.5} + |B|}{\mathbf{0.5} + \mathbf{W(i,B)}}$$

- 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.

To avoid a problem in case W(i,B)=0.

Relevance of a term w.r.t M and B



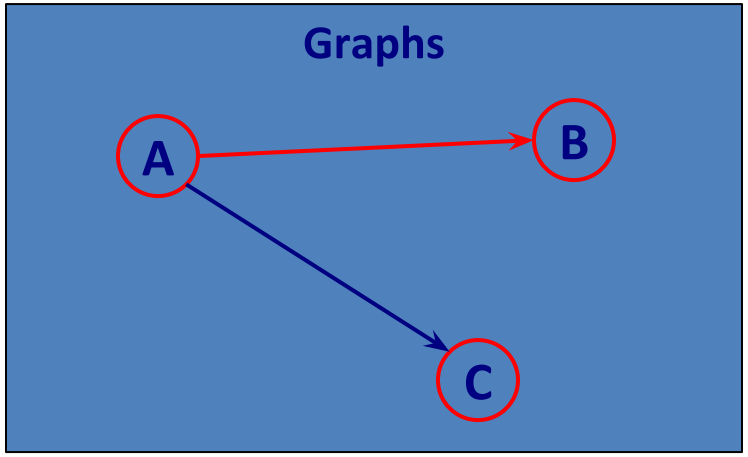
Construct malicious API graphs

- A malicious API graph consists of nodes and edges with the highest weight.
- How to link all these nodes and edges in a graph.

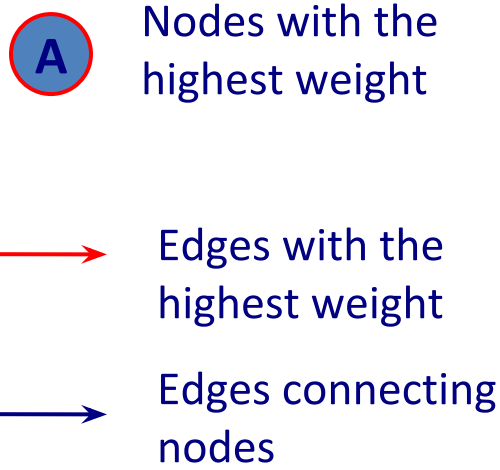
There are different possibilities for computing such graph.

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.

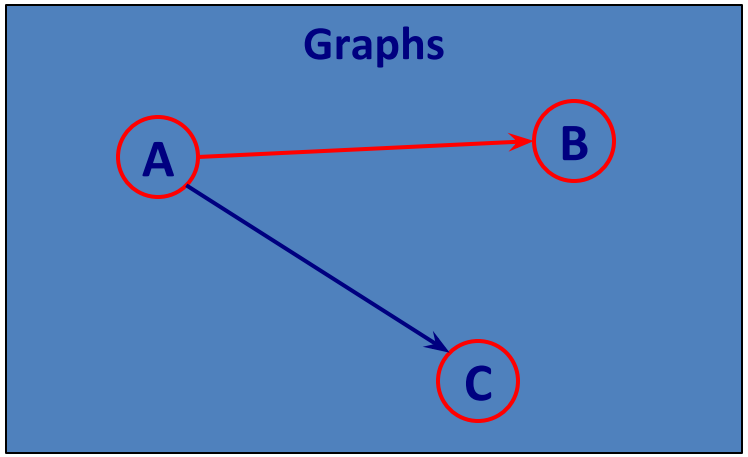


n = 3

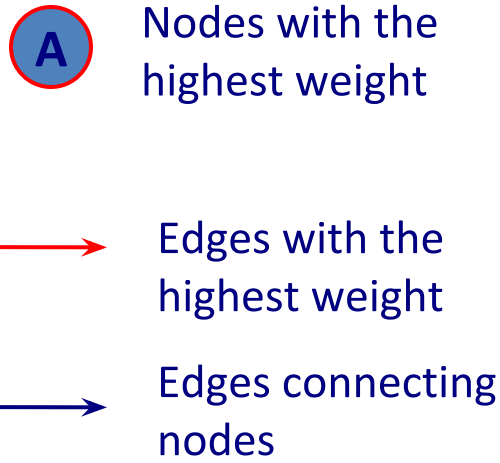


Strategy S0

- 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.

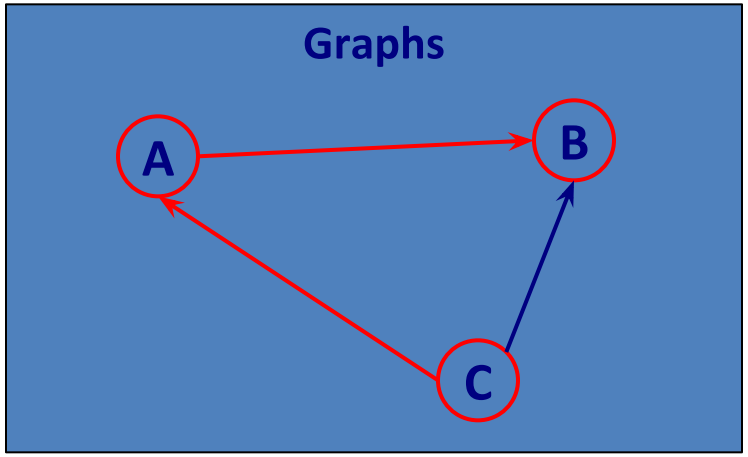


n = 3



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.



n = 3

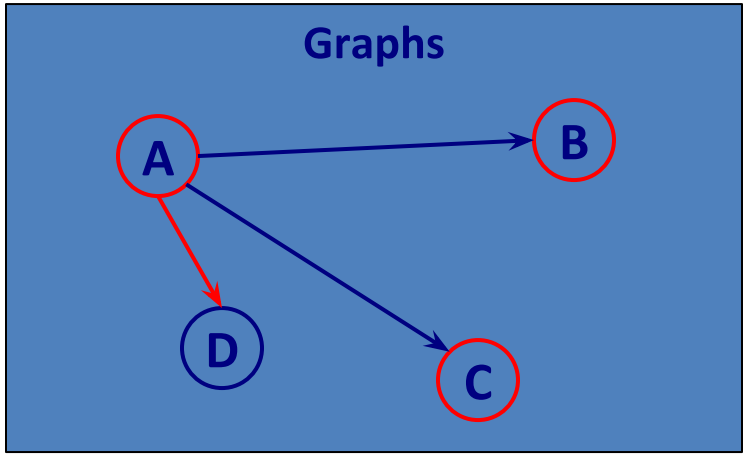
Nodes with the highest weight

Edges with the highest weight

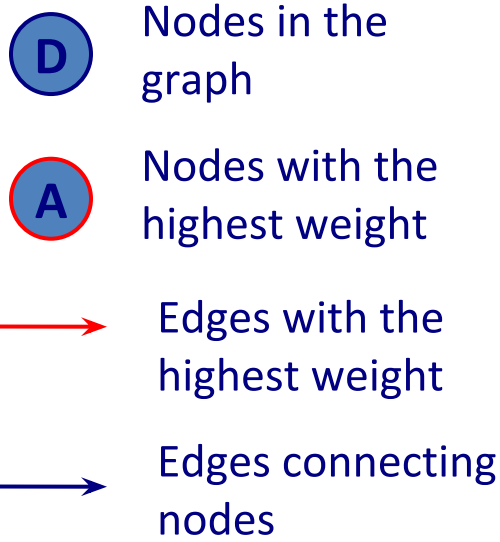
Edges connecting nodes

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.

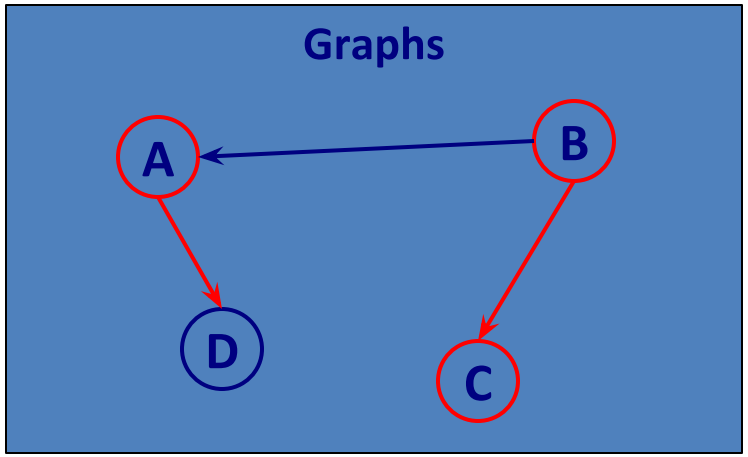


n = 3

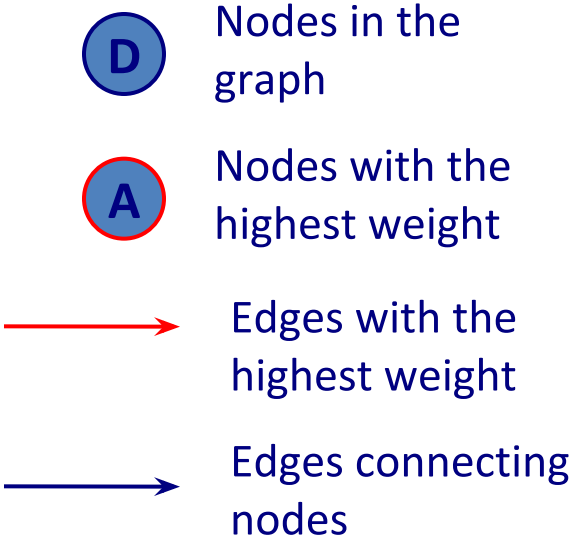


Strategy S1

- 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.

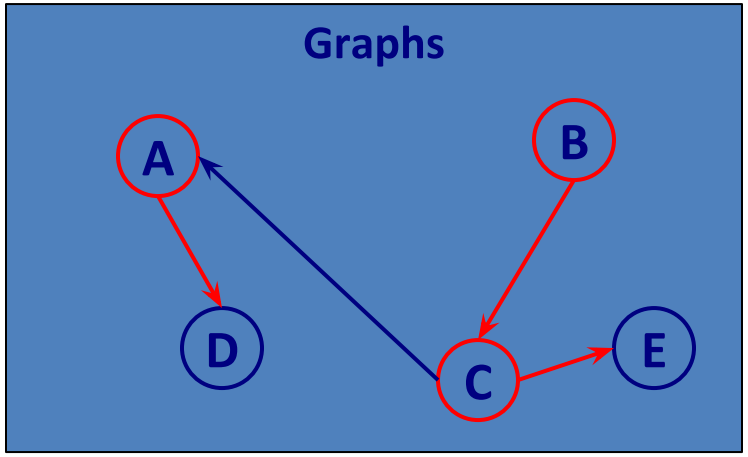


n = 3

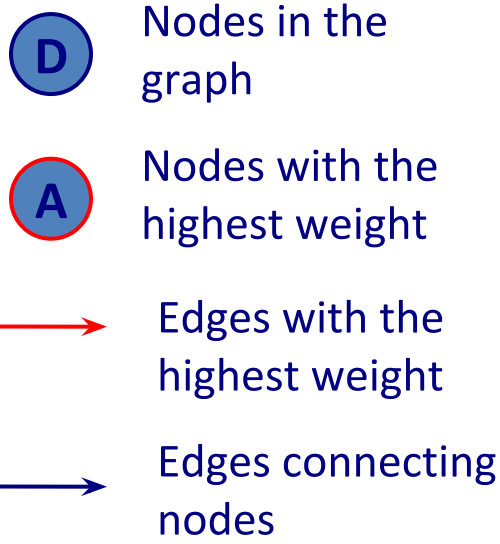


Strategy S1

- 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.

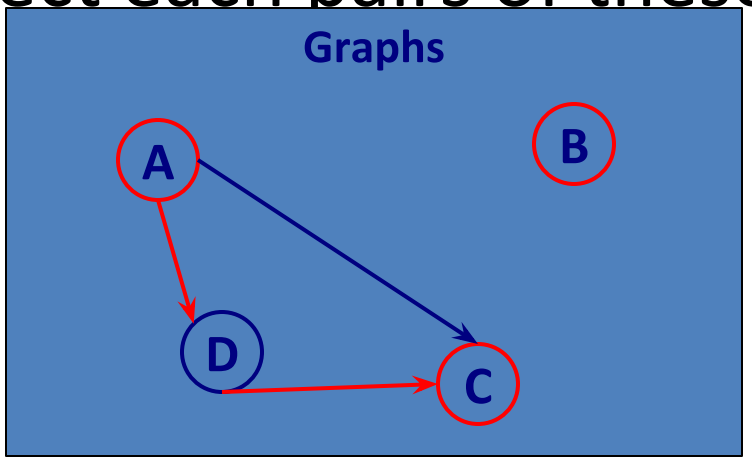


n = 3

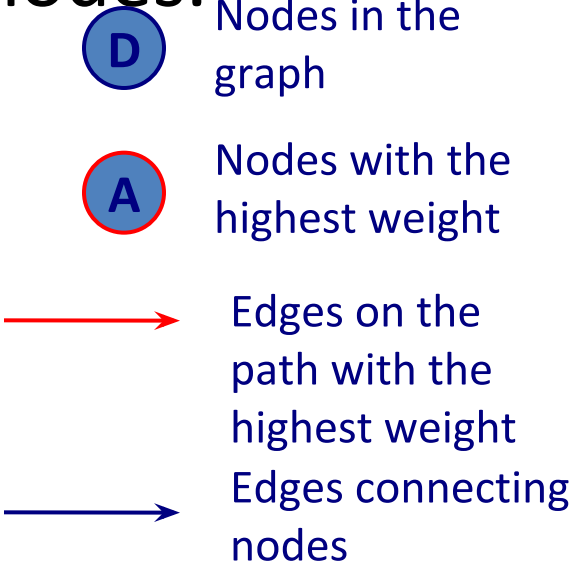


Strategy S2

- 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.

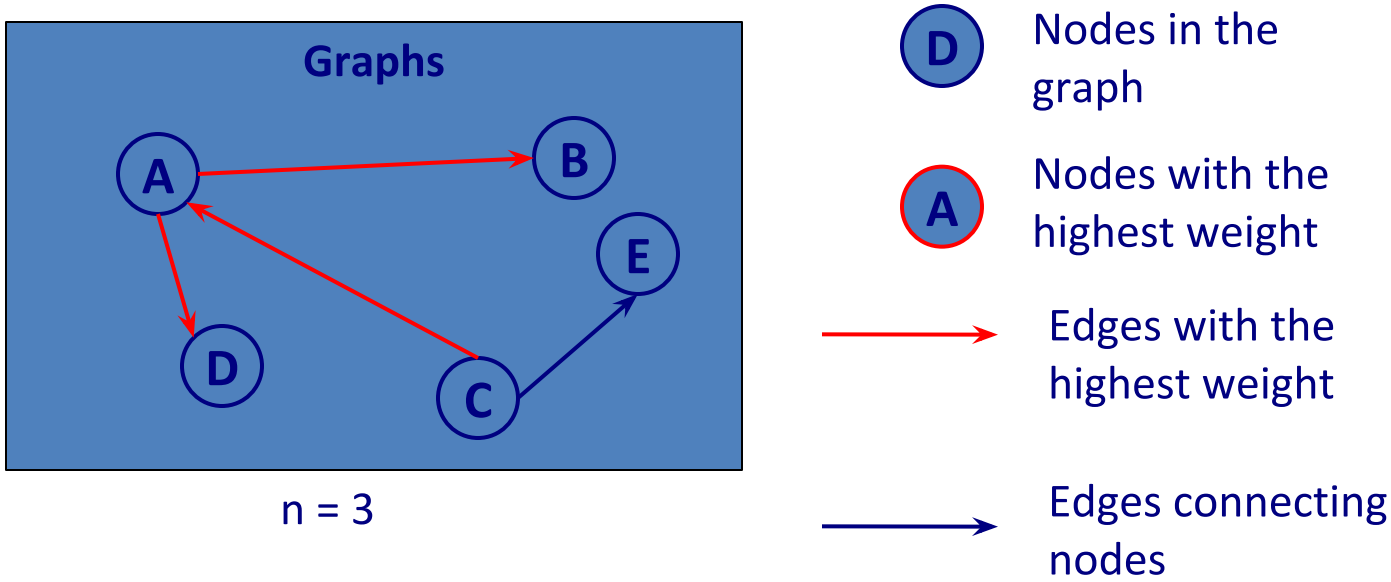


n = 3

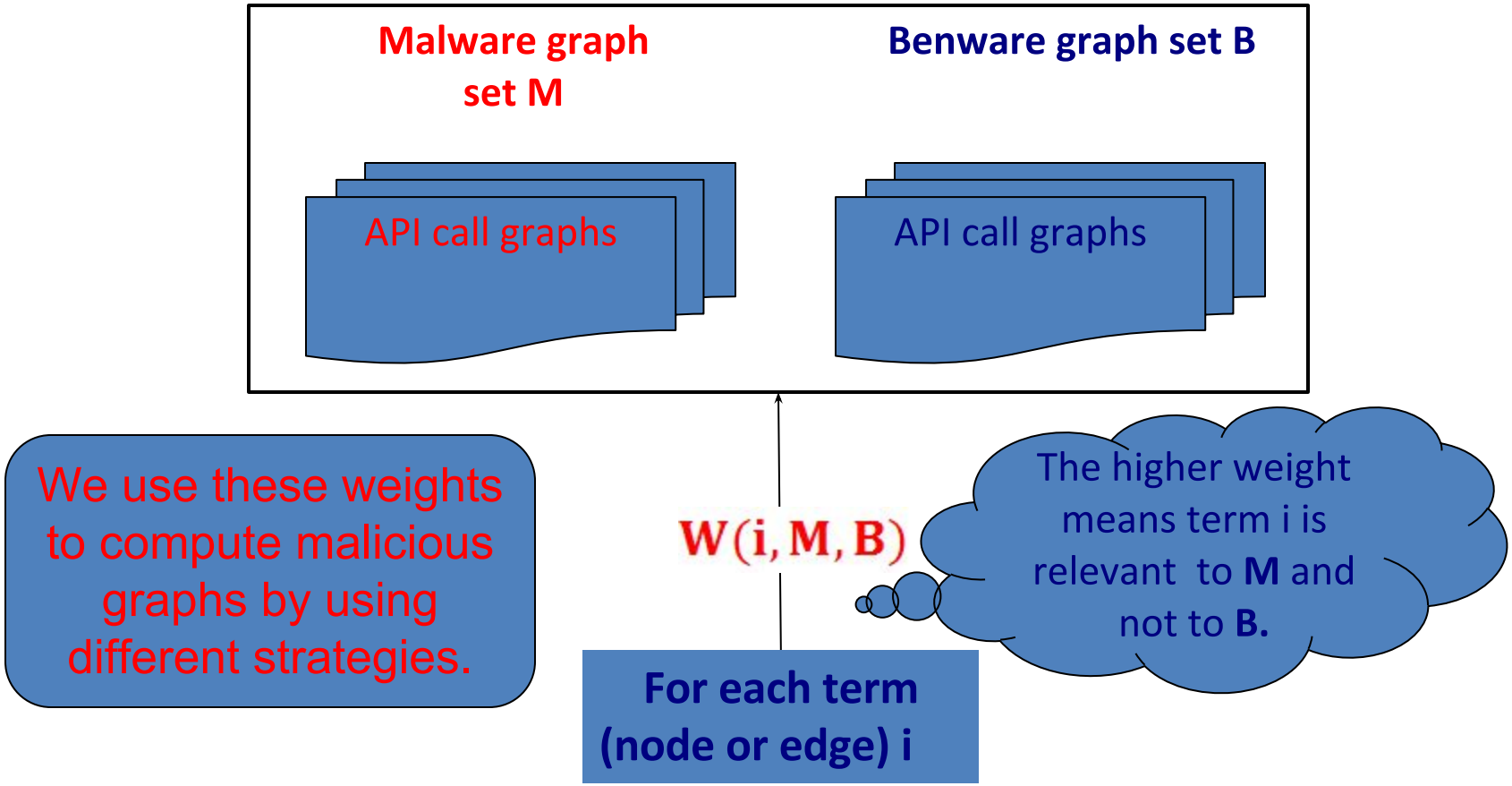


Strategy S3

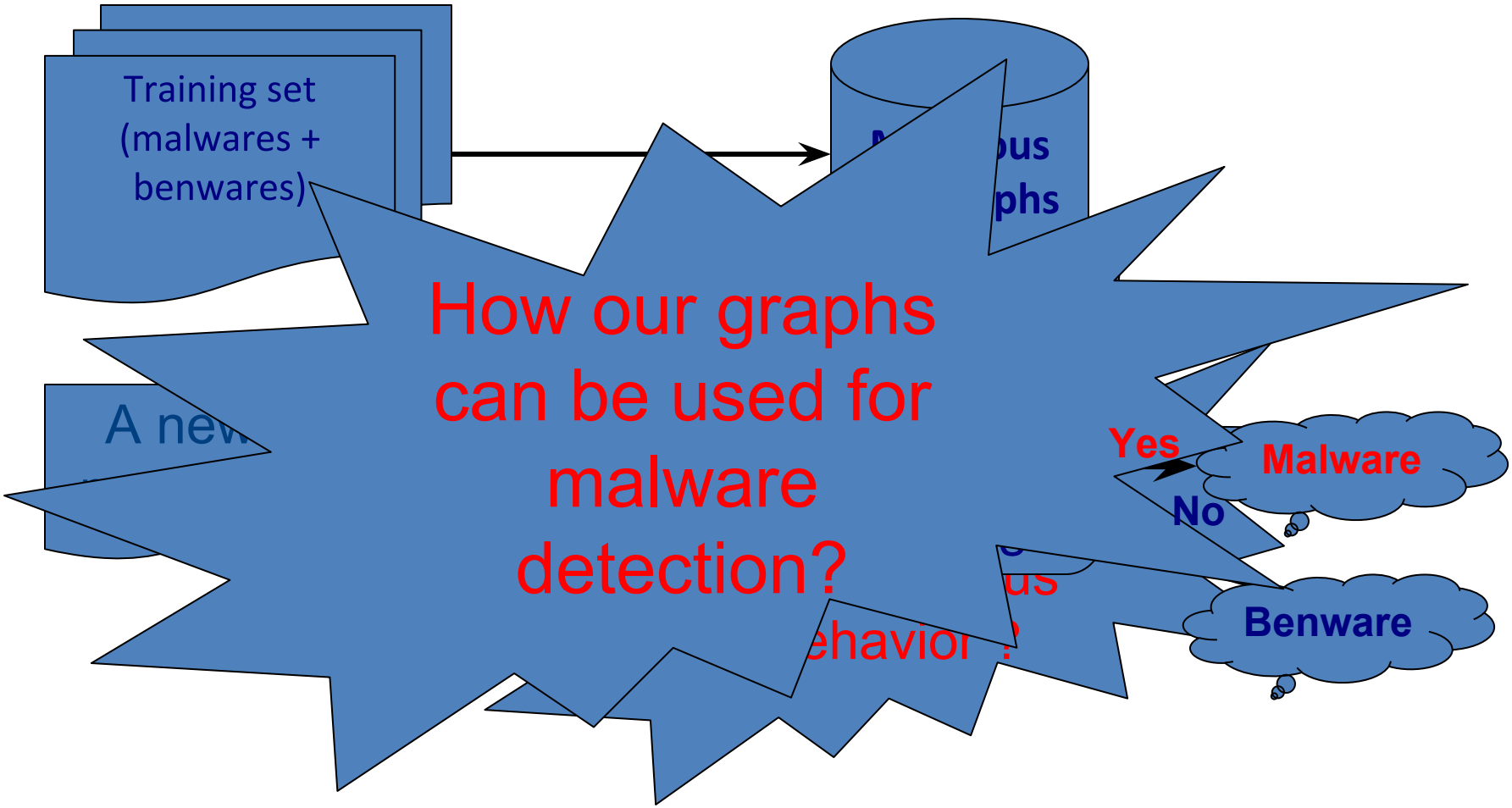
- Take n edges with the highest weight, for n given by the user.



Summary


Malware graph set M
Benware graph set B
API call graphs
API call graphs
We use these weights to compute malicious graphs by using different strategies.
W(i, M, B)
The higher weight means term i is relevant to M and not to B.
For each term (node or edge) i

How to detect malwares?



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 $\longrightarrow$ extract malicious graphs.
  - Test set consists of 980 benwares and 1560 malwares $\longrightarrow$ for evaluating malicious graphs.
- Evaluate different strategies and formulas.

Performance Measurement

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

$$\text{Recall} = \frac{\text{True Positives}}{\textit{Number of graphs}}$$

(Detection rate)

- High precision means that the technique com irrel

$$\text{Precision} = \frac{\text{True Positives}}{\textit{True Postives} + \textit{False Positives}}$$

Performance Measurement

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

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

  – F-Measure is 1 if all retrieved items are relevant and all relevant items have been retrieved.

Evaluating the performance of the different strategies

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

The best performance of each strategy.

Evaluating the performance of the different strategies

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

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

Antivirus	New malwares from Internet	New generated malwares	Antivirus	New malwares from Internet	New generated malwares
Our tool	**100%**	**100%**	Panda	25%	19%
Avira	50%	16%	Kaspersky	35%	81%
Avast	45%	87%	Qihoo-360	80%	96%
McAfee	40%	96%	AVG	40%	82%
BitDefender	40%	87%	ESET-NOD32	65%	87%
F-Secure	40%	87%	Symantec	40%	14%

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!