Malware Comparison with Frequency Analysis

Gabriel Duque

LSE - EPITA

gabriel.duque@epita.fr

February 13, 2018

Overview

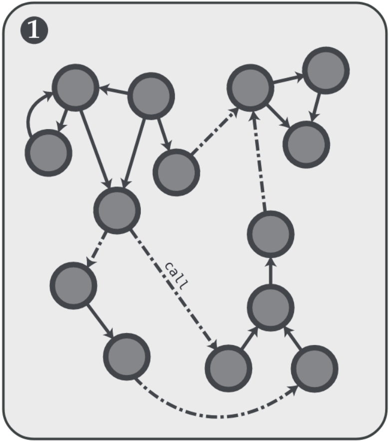
1. Individual Analysis
   - Function Detection
   - Denoising
2. The Malware Cluster
   - Term Frequency - Inverse Document Frequency
   - Lucene Indexes and Elastic Search

Nucleus

1. Interprocedural control flow graph (ICFG) construction
2. Basic block cluster detection
3. Directly called function detection
4. Unreachable function detection

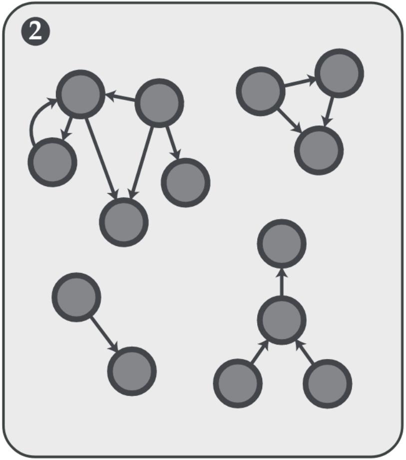
ICFG Construction (1)

- Disassembling with capstone
- Analyzing the flow of the binary
- Generating the ICFG



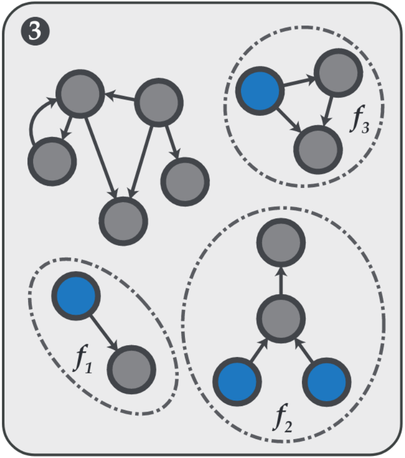
ICFG Construction (2)

- Ignoring the *call* edges
- Basic blocks connected through intraprocedural edges
- Detecting the basic block clusters



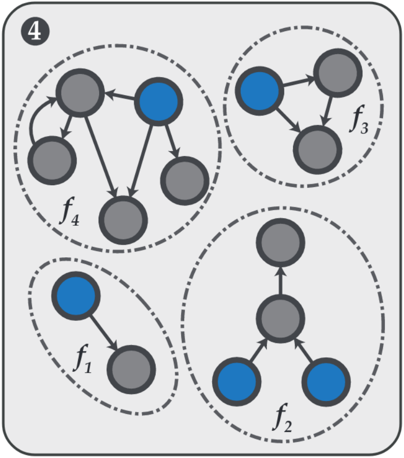
ICFG Construction (3)

- Reintroducing the *call* edges
- Following flow until complete block is formed
- Isolating the directly called functions



ICFG Construction (4)

- Iterating over basic blocks to find an isolated one
- Following the flow until complete block is formed
- Isolating the indirectly called functions



Denoising

```
import capstone

from .x86instr import JUMP_CALL_IDS
from .x86instr import STACK_REGISTERS

def is_call_or_jump(instr, jump_call_ids=JUMP_CALL_IDS):
    return instr['id'] in jump_call_ids

def is_mem_deref(op, register_blacklist=STACK_REGISTERS):
    return (op['op_type'] == capstone.x86.X86_OP_MEM
            and op['mem'] not in register_blacklist)

def is_imm_deref(op, data_range_list):
    if op['op_type'] == capstone.x86.X86_OP_IMM:
        for start, end in data_range_list:
            if start <= op['imm'] <= end:
                return True
    return False
```

TF-IDF

$$TFIDF_{w,d,D} = TF_{w,d} \times IDF_{w,D}$$

$$TFIDF_{w,d,D} = \frac{N_{w,d}}{N_d} \times \log \frac{N_D}{N_{w,D}}$$

$TF_{w,d}$: term frequency
$IDF_{w,D}$: inverse document frequency of w in D
$N_{w,d}$: number of times w appears in d
N_d: number of words in d
N_D: number of documents
$N_{w,D}$: number of documents containing w

TF-IDF: Simple Example

Document 1

Term	Term Count
this	1
is	1
a	2
sample	1

Document 2

Term	Term Count
this	1
is	1
another	2
example	3

TF

- Simply the raw frequency of a word in a document
- Represents the weight of the word in a single document

$$\mathrm{tf}(''\textsf{this}'', d_1) = \frac{1}{5} = 0.2 \qquad \mathrm{tf}(''\textsf{example}'', d_1) = \frac{0}{5} = 0$$

$$\mathrm{tf}(''\textsf{this}'', d_2) = \frac{1}{7} \approx 0.14 \qquad \mathrm{tf}(''\textsf{example}'', d_2) = \frac{3}{7} \approx 0.429$$

Document 1

Term	Term Count
this	1
is	1
a	2
sample	1

Document 2

Term	Term Count
this	1
is	1
another	2
example	3

IDF

- The inverse document frequency
- Represents the weight of the presence of the word in a document

$$\mathrm{idf}(''\text{this}'', D) = \log\left(\frac{2}{2}\right) = 0 \quad \mathrm{idf}(''\text{example}'', D) = \log\left(\frac{2}{1}\right) = 0.301$$

Document 1

Term	Term Count
this	1
is	1
a	2
sample	1

Document 2

Term	Term Count
this	1
is	1
another	2
example	3

TF-IDF

- The product of both
- To what extent this word is representative of its document

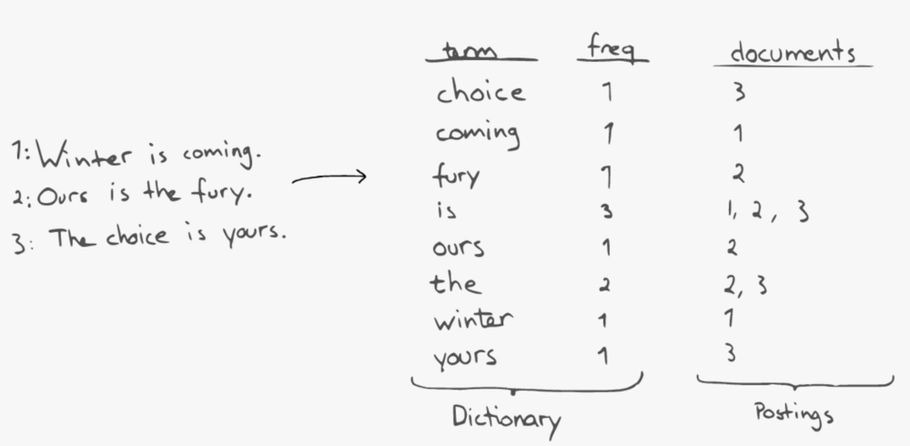
$$\text{tfidf}(''\textsf{this}'', d_1) = 0.2 \times 0 = 0$$
$$\text{tfidf}(''\textsf{this}'', d_2) = 0.14 \times 0 = 0$$

$$\text{tfidf}(''\textsf{example}'', d_1) = \text{tf}(''\textsf{example}'', d_1) \times \text{idf}(''\textsf{example}'', D) = 0 \times 0.301 = 0$$
$$\text{tfidf}(''\textsf{example}'', d_2) = \text{tf}(''\textsf{example}'', d_2) \times \text{idf}(''\textsf{example}'', D) = 0.429 \times 0.301 \approx 0.13$$

Elastic Search

- Elastic Search index: multiple Lucene indices (called shards in ES)
- Lucene index: multiple small inverted indices

Lucene Indexes



Problems and conclusion

- Too much memory (especially for Elastic Search)
- TF-IDF computation keeps getting longer
- Denoising & comparison are too simple
- Some malwares break the libraries I used (invalid section and segment names)
- Completely ignoring non-binary files and packed binaries

Questions ?