QBinDiff : A modular differ to enhance binary diffing and graph alignment

SSTIC 2024

Mercredi 05 Juin 2024

Roxane Cohen <rcohen@quarkslab.com>, PhD student
Robin David <rdavid@quarkslab.com>, R&D lead
Riccardo Mori <rmori@quarkslab.com>, Security engineer
Florian Yger <florian.yger@dauphine.psl.eu>, Associate professor
Fabrice Rossi <fabrice.rossi@dauphine.psl.eu>, Professor

AGENCE INNOVATION DÉFENSE

Quarkslab

Binary Diffing

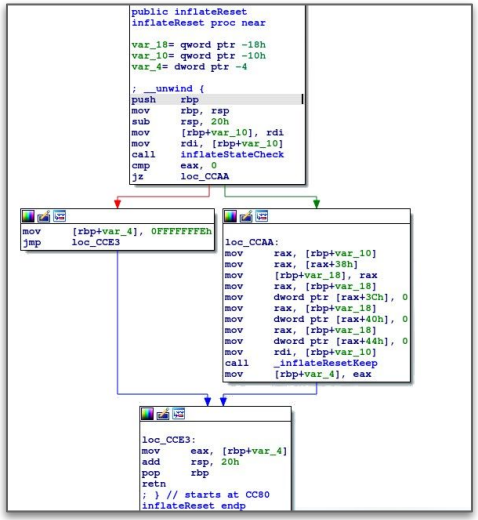
Introduction

Goal is **comparing** two *(or more)* binaries to analyze their differences. It usually done using functions with a `1-to-1` mapping computation.
(which can be problematic when functions are merged or split)

Use-cases:

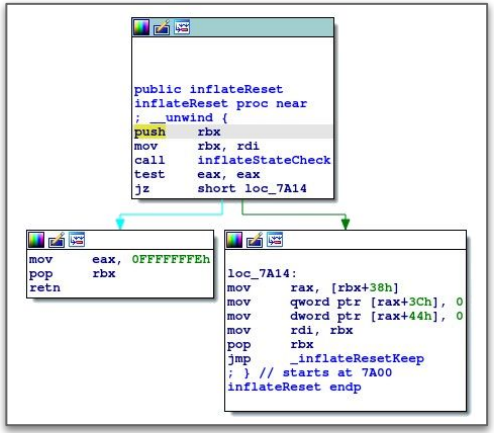
→ malware diffing *(analysing updates, or common components between two variants)*
→ patch analysis / 1-day analysis *(understanding if patch is correct, or what is 1-day about)*
→ anti-plagiarism
→ statically linked libraries identification *(static binary against some libs)*
→ symbol porting *(e.g: IDA annotations to a new version of a binary)*
→ backdoor detection *(legitimate binary against a modified version)*
→ cross-architecture diffing *(for symbol porting etc..)*

Impact of optimization & obfuscation



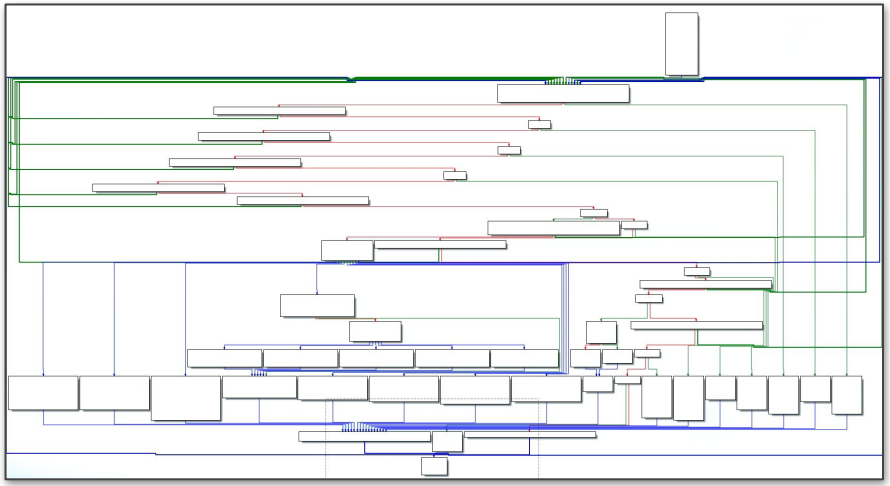
-O0 Compilation

(no optimization)



-O2 Compilation

(optimization)



Obfuscation

(virtualization)

Motivation: Our use-case

Edge-cases

- Two banks *(area)* in the same binary
- Specified subset of functions
- **Obfuscated** binaries

Core Ideas:

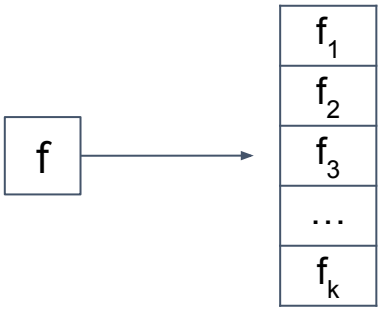
- Observation: Multiple obfuscations alters function contents **but not the overall program call graph** *(because harder to put in practice)*
- Need: We want to diff/relates/compare obfuscated binaries **without** having to deobfuscate them first.
- Wish: We want to bring manually acquired knowledge for the diff e.g: anchors, specific features etc.

⇒ Can improve diff using **resilient** features, analyst knowledge

Diffing ain't Similarity

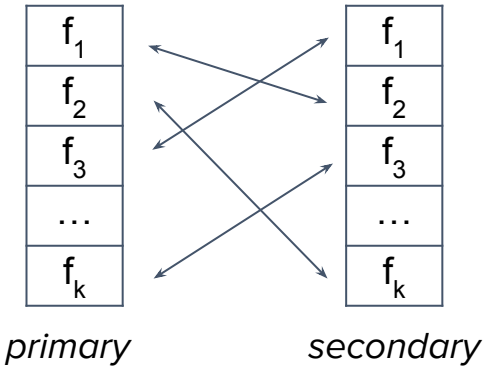
Binary similarity

Which function is the **most similar** to *f* among a pool of size *k* ?



Binary diffing

What is the **best mapping** between functions of primary and secondary ?



Diffing = Similarity + Matching

(from similarity scores, create an assignment…)

QBinDiff

Quarkslab

QBindiff

Algorithm: Solve the Network Alignment Problem using an optimization algorithm based on **message passing** (belief propagation) to arbitrate function similarity and call-graph topology.

Blog ↗

Key Features:

- Disassembler agnostic *(use exported representation)*
- Standalone program
- Python API *(to be used programmatically)*
- Two APIs:
  - High-level for binary diffing
  - Low-level for arbitrary diffing *(matrices as input)*
- Designed to be **modular**!

TL;DR: Anything that can be encoded as features and a graph can be diffed!

Diffing Landscape

- ✔ **decompiler**
- ✘ exporter
- ✔ precision
- ✘ recall

- ✔ **fast**
- ✘ no API *(not modular)*
- ✔ now OSS
- ✔ disass agnostic

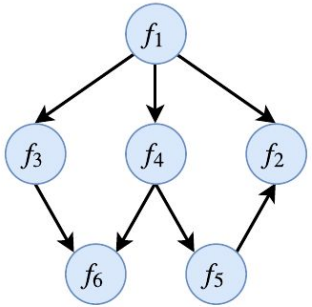
- ✔ **modular**
- ✘ memory & time
- ✘ power-user
- ✔ generic API

		Diaphora	Bindiff	Radiff2	QBindiff	Ghidriff
	Language	Python	Java	C	Python	Python
Disassembler	IDA	✔	✔	✘	✔	✘
	Ghidra	✘	✔	✘	✔	✔
	Binja	✘	✔	✘	✔	✘
	Radare2	✘	✘	✔	✘	✘
	Exporter	SQLite	Binexport	n/c	Binexport Quokka	n/c
	Scripting API	✔	✘	n/c	✔	n/c
	Use decompiler	✔	✘	✘	✘	n/c

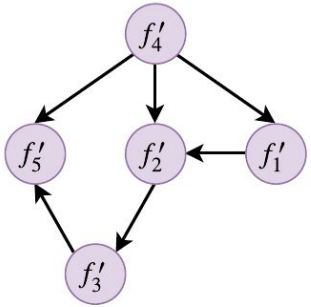
In-house exporter

Algorithm

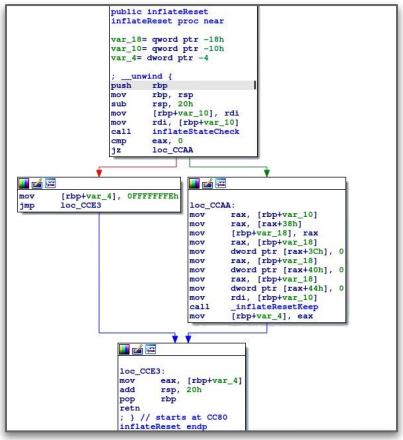
Sample 1 (#M nodes)



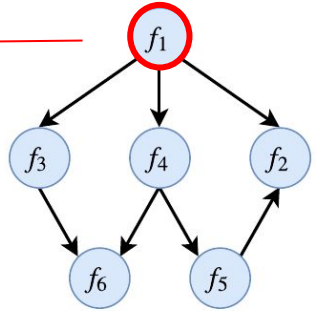
Sample 2 (#N nodes)



Algorithm

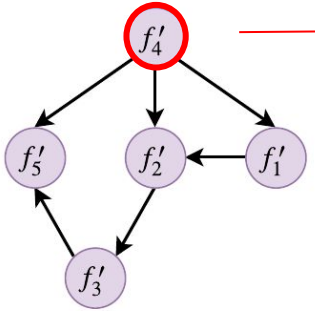


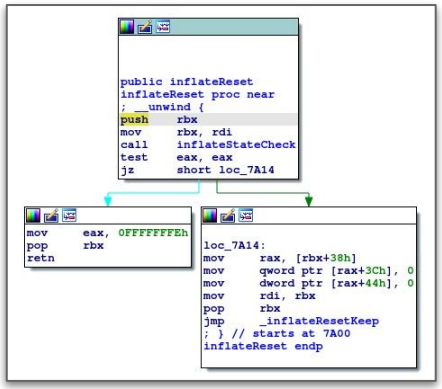
Sample 1 (#M nodes)



N

Sample 2 (#N nodes)





Features
(# nodes, # edges, cyclomatic complexity...)

4, 4, 2...

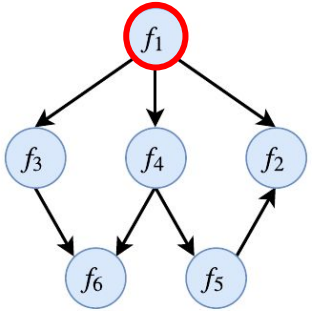
0 < Similarity < 1

Features
(# nodes, # edges, cyclomatic complexity...)

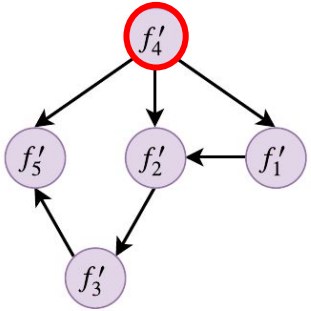
3, 2, 1...

Algorithm

Sample 1 (#M nodes)



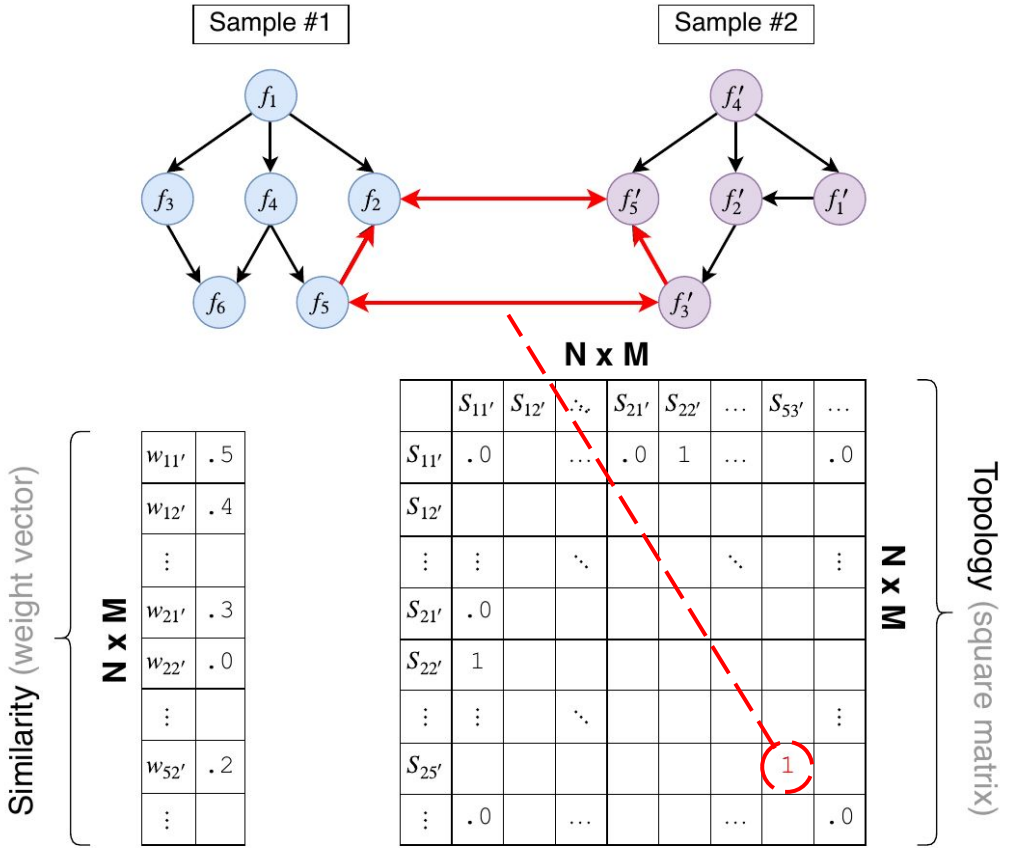
Sample 2 (#N nodes)



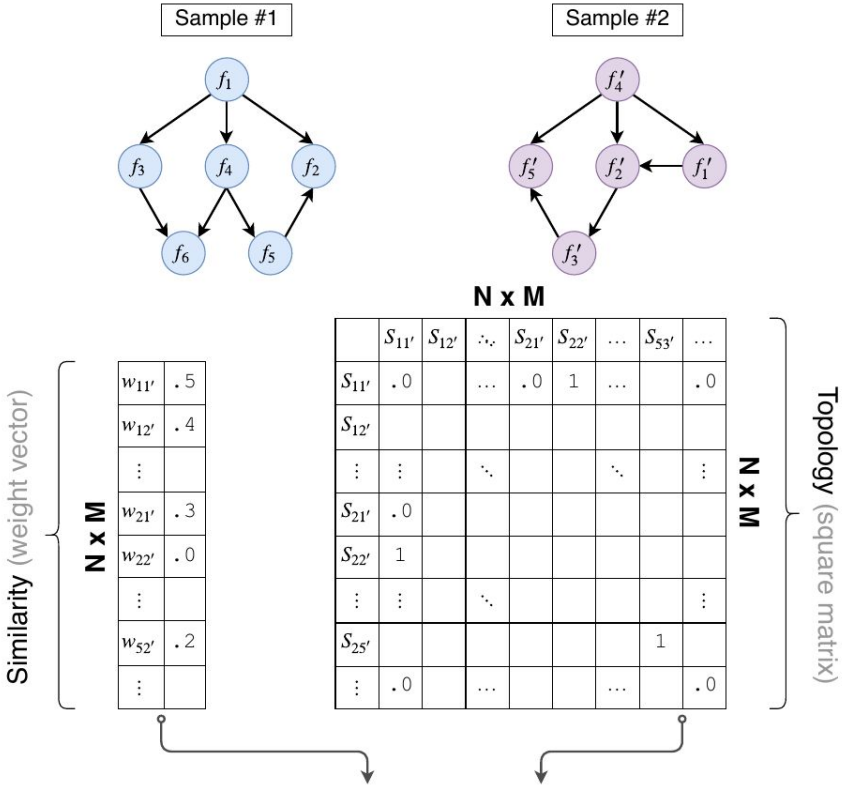
Similarity (weight matrix)

M \ N	f'_1	f'_2	…	f'_4	f'_5
f_1	.5	.4		.6	.0
f_2	.3	.0		1	1
⋮	⋮		⋱		
f_5	.6	.2			
f_6	.6				

Algorithm


Sample #1
Sample #2
f_1 f_2 f_3 f_4 f_5 f_6
f'_1 f'_2 f'_3 f'_4 f'_5
N x M
Similarity (weight vector)
N x M
$w_{11'}$ .5
$w_{12'}$ .4
$w_{21'}$ .3
$w_{22'}$ .0
$w_{52'}$ .2
$S_{11'}$ $S_{12'}$ $S_{21'}$ $S_{22'}$ $S_{53'}$
$S_{11'}$ .0 .0 1 .0
$S_{12'}$
$S_{21'}$ .0
$S_{22'}$ 1
$S_{25'}$ 1
.0 .0 .0
N x M
Topology (square matrix)

Algorithm



Goal: **Arbitrate** between **function similarity** and **call-graph topology** to be more resilient if one of them is altered *(+ still use imported functions as anchors)*

We said Modular ?

General Parameters

- **features**: 27 functions features *(some taken from diaphora / bindiff)*
- **distance**: cosine, euclidean, haussmann *(custom one)*
- **tradeoff**: cursor on function similarity or call graph topology
- **sparsity ratio**: percentage of candidate matches to keep in similarity matrix
- **epsilon**: Relaxation parameter *(helps converging faster)*
- **iterations**: max number of belief propagation steps

⇒ *We provide "best" default values for each of them*

Developers Modularity

- custom executable format *(to load arbitrary file)*
- can diff anything by providing low-level matrices
- can develop custom features

"e.g: compilation unit aware feature where functions from the same CU, shall be close to their sibling in terms of addresses or order in the binary"

⇒ Obtaining a good diff might require fine tuning parameters.

QBinDiff Usage

Command line

```
$ qbindiff primary.BinExport \
           secondary.BinExport \
           -ff bindiff -o result.BinDiff \# output in bindiff format
           -a1 CS_ARCH_ARM:CS_MODE_THUMB \# with .BinExport better to
           -a2 CS_ARCH_ARM:CS_MODE_THUMB   # specify arch in capstone
File loading ──────────────────────────────────────────────────────────
100% 0:00:00
Initialization ────────────────────────────────────────────────────────
100% 0:00:00
Matching ──────────────────────────────────────────────────────────────
100% 0:00:00
Saving Results ────────────────────────────────────────────────────────
100% 0:00:00

┌─────────────┬──────────────────────┐
│ Score       │             206.0000 │
│ Similarity  │             108.0000 │
│ Squares     │                   98 │
│ Nb matches  │                  108 │
├─────────────┼──────────────────────┤
│ Node cover  │ 100.000% / 100.000%  │
│ Edge cover  │ 100.000% / 100.000%  │
└─────────────┴──────────────────────┘
```

API usage

```python
from qbindiff import QBinDiff, Program
from qbindiff.features import CyclomaticComplexity # etc

p1 = Program("primary.BinExport")
p2 = Program("secondary.BinExport")

differ = QBinDiff(p1, p2)
differ.register_feature_extractor(CyclomaticComplexity, 1.0)
# register your features

differ.process()
mapping = differ.compute_matching()
# do anything you want if the result
```

⇒ **Output is either a CSV or a .BinDiff file** *(to open it with bindiff)*

Benchmarks

Quarkslab

Diffing Evaluation

How can we compare the functions pair that should be matched *(Ground-Truth)* **and the functions that are matched by a differ on stripped binaries ?**

True Positives
good match
correctly identified

False Positives
wrong match
identified

True Negative
Not a match
considered as-is

False Negative
Good match **not**
identified

Precision = $\frac{\square}{\square + \square}$ **Recall** = $\frac{\square}{\square + \square}$ $\Rightarrow$ **F1-score** = $2 \times \frac{P \times R}{P + R}$

Benchmarks

		BinDiff	Diaphora3	QBinDiff (BinExport)	QBinDiff (Quokka)
zlib	libz.so.1.2.11	0.85	0.65	0.82	**0.88**
openssl	libssl.so.3	0.81	0.64	0.83	**0.86**
	openssl	0.95	0.68	0.92	**0.98**
	libcrypto.so.3	0.76	0.78	0.67	**0.82**
nmap	nping	0.59	0.52	0.73	**0.77**
	ncat	0.73	0.58	0.86	**0.92**
	nmap	0.8	0.8	0.73	**0.82**
clamav	libclamav	0.58	0.46	0.76	**0.81**
curl		0.65	0.56	0.83	**0.88**
unrar		0.68	0.62	0.81	**0.87**
Averaged		0.74	0.63	0.80	**0.86**

Standard differ f1-score comparison in a cross-optimizer/optimization setting

(differ robustness against compilation variation)

Benchmarks

		BinDiff	Diaphora3	QBinDiff (BinExport)	QBinDiff (Quokka)
zlib	libz.so.1.2.11	0.85	0.65	0.82	**0.88**
openssl	libssl.so.3	0.81	0.64	0.83	**0.86**
	openssl	0.95	0.68	0.92	**0.98**
	libcrypto.so.3	0.76	0.78	0.67	**0.82**
nmap	nping	0.59	0.52	0.73	**0.77**
	ncat	0.73	0.58	0.86	**0.92**
	nmap	0.8	0.8	0.73	**0.82**
clamav	libclamav	0.58	0.46	0.76	**0.81**
curl		0.65	0.56	0.83	**0.88**
unrar		0.68	0.62	0.81	**0.87**
Averaged		0.74	0.63	0.80	**0.86**

Quokka is better than BinExport
(exporter matters)

Standard differ f1-score comparison in a cross-optimizer/optimization setting
(differ robustness against compilation variation)

Benchmarks

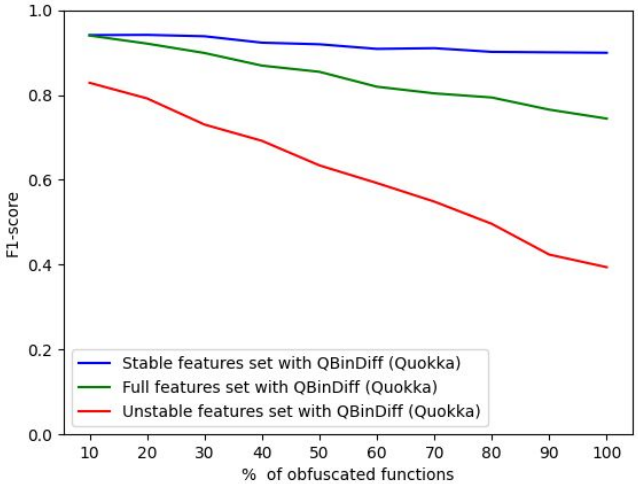
		BinDiff	Diaphora3	QBinDiff (BinExport)	QBinDiff (Quokka)
zlib	libz.so.1.2.11	0.85	0.65	0.82	**0.88**
openssl	libssl.so.3	0.81	0.64	0.83	**0.86**
	openssl	0.95	0.68	0.92	**0.98**
	libcrypto.so.3	0.76	0.78	0.67	**0.82**
nmap	nping	0.59	0.52	0.73	**0.77**
	ncat	0.73	0.58	0.86	**0.92**
	nmap	0.8	0.8	0.73	**0.82**
clamav	libclamav	0.58	0.46	0.76	**0.81**
curl		0.65	0.56	0.83	**0.88**
unrar		0.68	0.62	0.81	**0.87**
Averaged		0.74	0.63	0.80	**0.86**

QBinDiff performs better compared to other differs

Standard differ f1-score comparison in a cross-optimizer/optimization setting

(differ robustness against compilation variation)

Glimpse of Obfuscation *(results)*



QBinDiff feature impact : stable, full and unstable features

(Control-Flow Graph Flattening f1-score evolution)

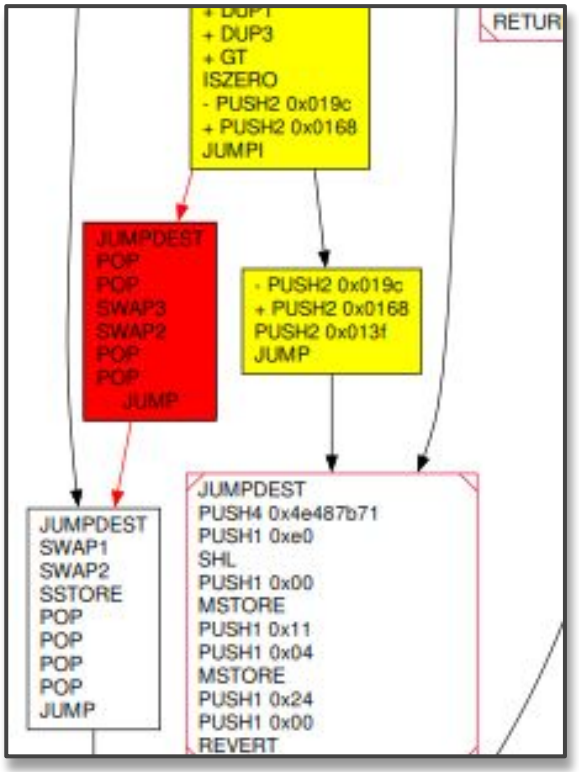
What about diffing other things ?

Quarkslab

Smart Contracts

⇒ Diffing two closed-source **smart contracts** using `pyevmasm` for disassembly, `EtherSolve` for CFG and call graph reconstruction.

⇒ Low-level API diffing with Qbindiff !



(examples by Elouan Wauquier)

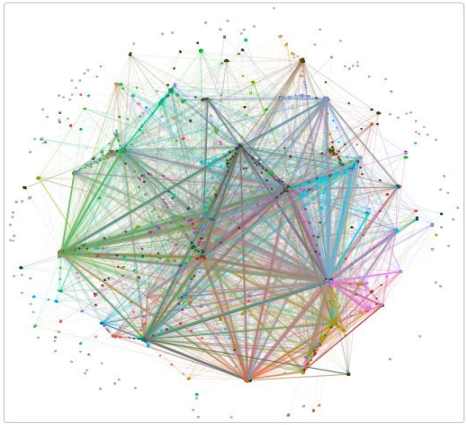
Application to other fields

⇒ Low-Level API enables diffing anything, inputs are similarity matrix and relationship graphs.

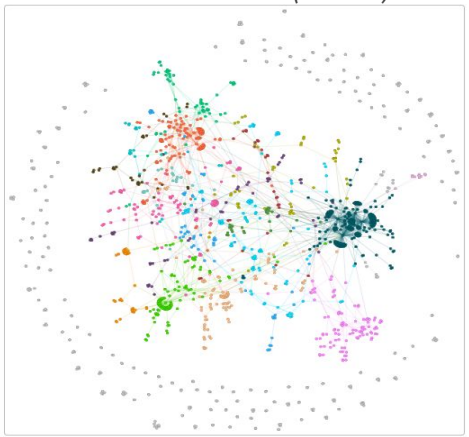
Application to other fields

⇒ Low-Level API enables diffing anything, inputs are similarity matrix and relationship graphs.

Homo-sapiens



Mus musculus *(mouse)*



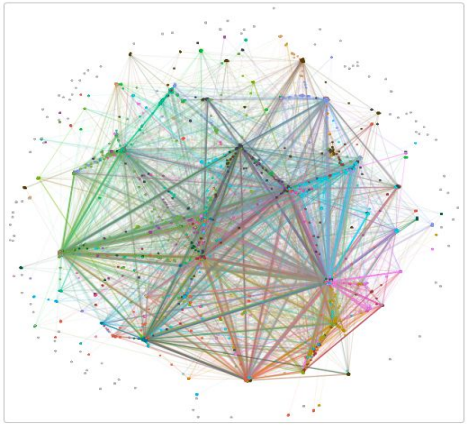
Bioinformatics: Protein-protein interactions

(node proteins, edge interactions between them)

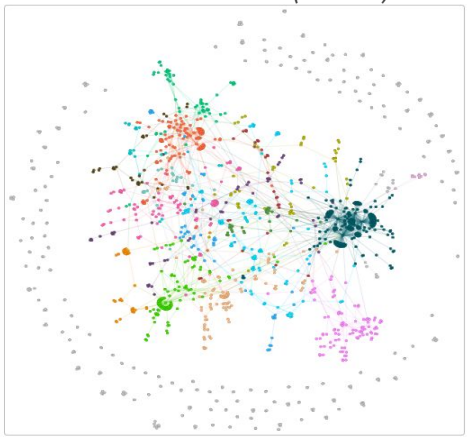
Application to other fields

⇒ Low-Level API enables diffing anything, inputs are similarity matrix and relationship graphs.

Homo-sapiens

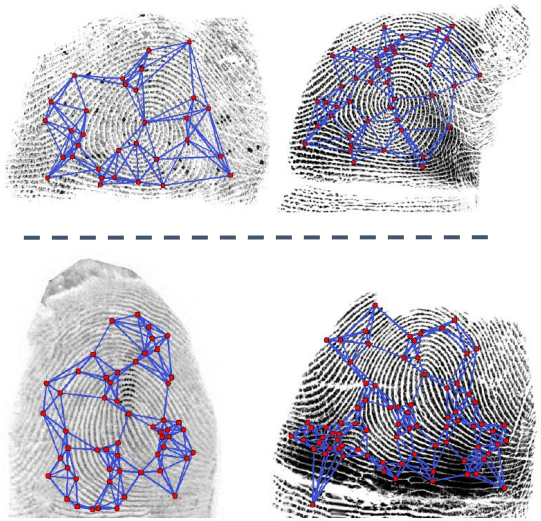


Mus musculus *(mouse)*



Bioinformatics: Protein-protein interactions

(node proteins, edge interactions between them)



Matching fingerprints using Minutiae

(Data from NIST dataset 302)

⇒ **Generic optimization algorithms** *(binary diffing just a reification of the problem)*

Conclusion

Binary Diffing

On common cases standard differs do the job great !
For more specific use-cases Qbindiff is more adapted !

General Diffing

QbinDiff can diff anything (as long as you can compute similarity between objects and determine relationships between objects)

Notes:

- `$ pip install qbindiff`
- Platform for experiments
- Actively maintained & answer questions

Huge Thanks:

- current contributors
- past contributors (Alexis Challande, Elie Mengin)

To contribute..

https://github.com/quarkslab/qbindiff/

Thank you

Contact information:

Email: contact@quarkslab.com

Phone: +33 1 58 30 81 51

Website: quarkslab.com

@quarkslab

Quarkslab