Raffinement du diffing binaire par arbitrage de la similarité et du matching

CAID 2024

19 Novembre 2024, Rennes, France

Roxane Cohen <rcohen@quarkslab.com>
Robin David <rdavid@quarkslab.com>
Riccardo Mori <rmori@quarkslab.com>
Florian Yger <florian.yger@insa-rouen.fr>
Fabrice Rossi <rossi@ceremade.dauphine.fr>

Dauphine | PSL UNIVERSITÉ PARIS

AGENCE INNOVATION DÉFENSE

Quarkslab

Agenda

Partie 1: Introduction

- Diffing binaire & similarité pour la cybersécurité

Partie 2: QBinDiff

- Algorithme de diffing par Belief Propagation
- Étude d'ablation

Partie 3: Benchmark

- Benchmark (en cas standard)
- Conclusion

Partie 1

Introduction

Quarkslab

Comparaison de programmes = Diffing binaire

Définition

Comparer deux *(ou plus)* programmes pour analyser leurs différences.
Effectué au niveau des fonctions avec une correspondance `1-to-1`
(problématique quand des fonctions ont été ajoutées ou retirées)

Comparaison de programmes = Diffing binaire

Définition

Comparer deux *(ou plus)* programmes pour analyser leurs différences.
Effectué au niveau des fonctions avec une correspondance `1-to-1`
(problématique quand des fonctions ont été ajoutées ou retirées)

Cas d'usage:

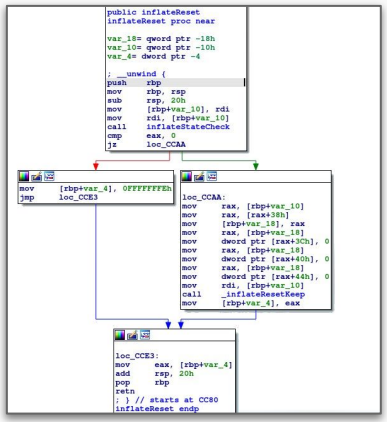
- diffing de malware *(analyse de nouvelles fonctionnalités)*
- analyse de patch / 1-day analyse *(Patch Tuesday, vérifier si un patch est correct)*
- anti-plagiat
- identifier les bibliothèques compilées statiquement *(binaire compilé statiquement vs bibliothèques)*
- porter des symboles *(e.g: annotations d'IDA vers un nouveau binaire)*
- détection de backdoor *(binaire légitime contre une version modifiée suspecte)*
- diffing cross-architecture

Problématique

Comment comparer les fonctions de manière **résiliente** ?
Quels **artefacts** utiliser pour cela ?

Contraintes:

- résilient à des mises à jour du code
- résilient à des options de compilations différentes
- résilient à de l'obfuscation *(de préférence)*
- résilient à des architectures différentes *(de préférence)*



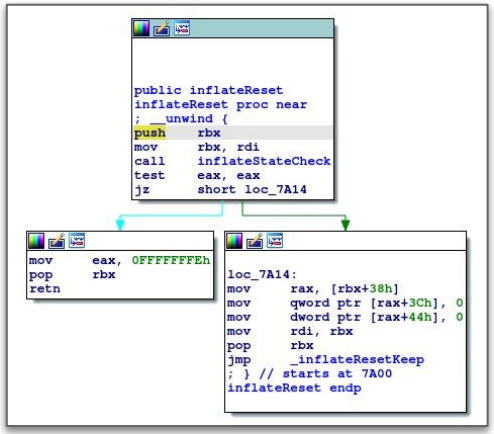
Ces deux fonctions sont-elles les mêmes ?

Représentation de base

Représentations

CFG: Control-Flow Graph

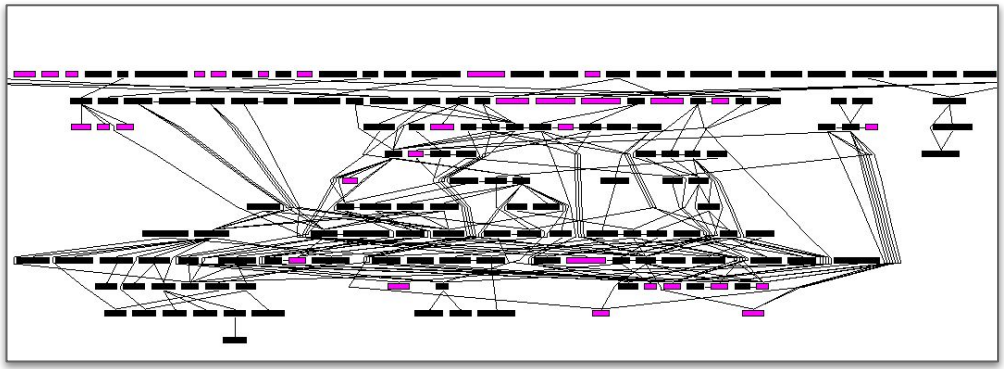
(un par fonction)



Noeuds: basic-blocs et arêtes: chemins possibles dans la fonction.

CG: Call Graph

(un par programme)



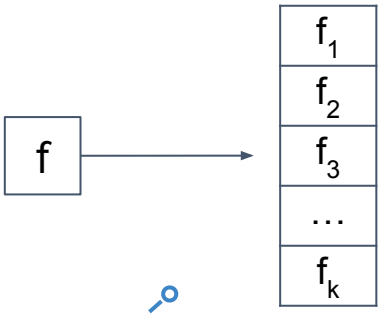
Noeuds: une fonction et arêtes: appels entre les fonctions.

? Comment — En combinant **similarité** et **matching** binaire *(utilisant les deux structures de graphe)*

Diffing vs Similarité

Similarité

Quelle fonction est **la plus proche** de f au sein d'un ensemble de taille k ?

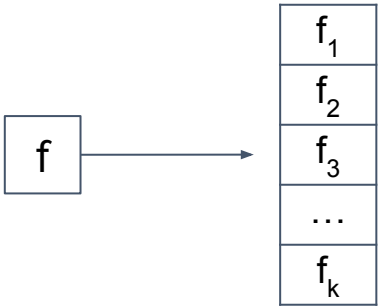


recherche de vulnérabilités, détection de clones, filiation de code

Diffing vs Similarité

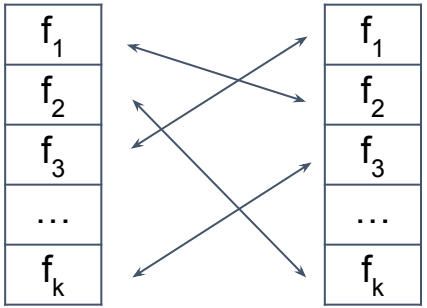
Similarité

Quelle fonction est **la plus proche** de f au sein d'un ensemble de taille k ?



Matching

Quelle est la **meilleure correspondance** entre les fonctions du primary et du secondary ?

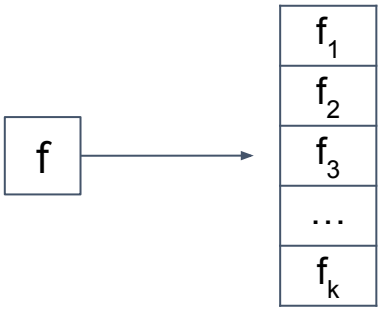


primary *secondary*

Diffing vs Similarité

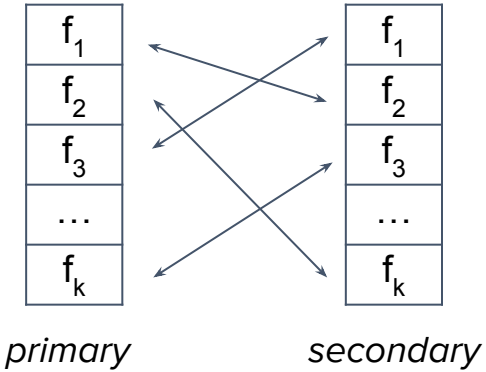
Similarité

Quelle fonction est **la plus proche** de f au sein d'un ensemble de taille k ?



Matching

Quelle est la **meilleure correspondance** entre les fonctions du primary et du secondary ?



Diffing = Similarité + Matching

(avec les scores de similarité, créer une correspondance…)

Differs à l'état-de-l'art

✔ **decompiler**
✘ exporter

✔ **rapide**
✘ pas d'API
✔ Indépendant d'un désassembleur

		Diaphora	Bindiff	Radiff2	Ghidriff
	Langage	Python	Java	C	Python
Disassembler	IDA	✔	✔	✘	✘
	Ghidra	✘	✔	✘	✔
	Binary Ninja	✘	✔	✘	✘
	Radare2	✘	✘	✔	✘
	Exporter	SQLite	Binexport	n/c	n/c
	Algorithm	Matching par règles	Propagation de matches	Modification de bytes	Matching par règles *(+similarité)*

Similarité binaire

Diffing = similarité + matching

1. Utiliser des approches par similarité binaire *(état de l'art ~ deep learning ~ coûteux)*
2. Combiner les résultats avec un algorithme de matching *(algorithme hongrois ~ n^3)*

	GMN [1]	Asm2vec [2]	PalmTree [3]	jTrans [4]
Langage	Python	Python	Python	Python
Technique	GNN	word2vec	transformers	transformers

[1] Li and al.**Graph Matching Networks for Learning the Similarity of Graph Structured Objects.** 2019
[2] Ding and al. **Asm2Vec: Boosting Static Representation Robustness for Binary Clone Search against Code Obfuscation and Compiler Optimization.** 2019
[3] Li and al. **PalmTree: Learning an Assembly Language Model for Instruction Embedding.** 2021
[4] Wang and al. **jTrans: Jump-Aware Transformer for Binary Code Similarity.** 2022

Évaluer un résultat de diffing

Comment comparer les paires de fonctions qui doivent être matchées (vérité-terrain) aux fonctions matchées par un differ pour deux binaires strippés ?

Vrai Positifs
un match correct trouvé

Faux Positifs
un faux match trouvé

Vrai Négatif
deux fonctions ne sont pas matchées *(et c'est correct)*

Faux Négatif
Un match n'est pas trouvé

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

Partie 2

QBinDiff

Quarkslab

QBindiff

QBinDiff: Résoudre le problème d'alignement de graphe en utilisant un algorithme d'optimisation *(propagation de messages)* afin d'arbitrer la similarité de fonctions et la topologie du call-graph

Fonctionnalités:

- Indépendant du désassembleur *(BinExport ou Quokka)*
- Utilisable en ligne de commande
- Python API *(à utiliser programmatiquement)*
- Deux APIs:
  - Haut-niveau pour du diffing binaire
  - Bas-niveau pour du diffing général *(matrice et similarité)*
- Conçu pour être **modulaire**!
- **Open-source**

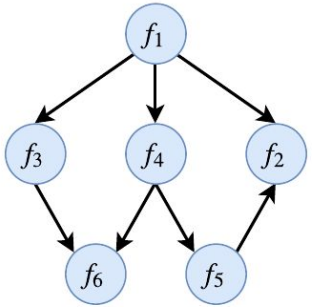
In-house exporter

Blog ↗

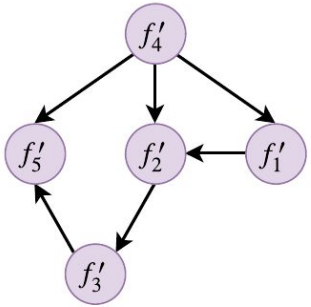
A modular differ to enhance binary diffing and graph alignment, SSTIC 2024

Algorithme

Programme 1 (#M noeuds)

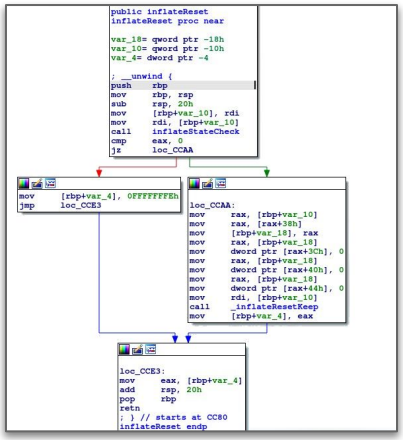


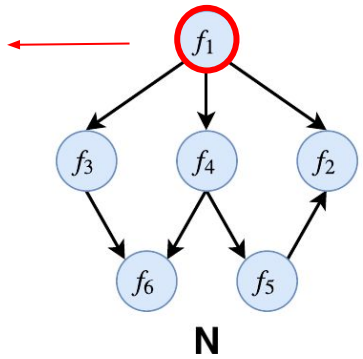
Programme 2 (#N noeuds)



Algorithme

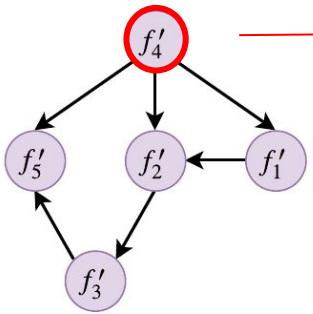
Programme 1 (#M noeuds)

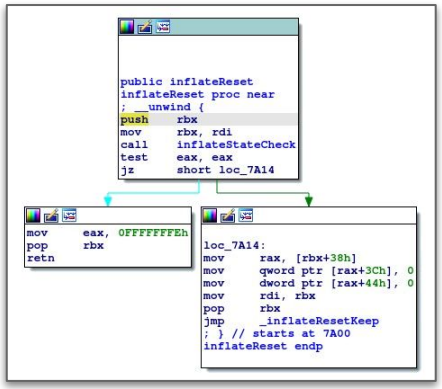




N

Programme 2 (#N noeuds)





Features
(# noeuds, # arêtes, complexité cyclomatique...)

4, 4, 2...

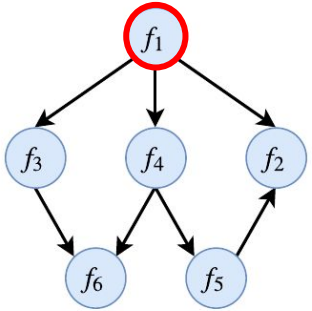
0 < Similarité < 1

Features
(# noeuds, # arêtes, complexité cyclomatique...)

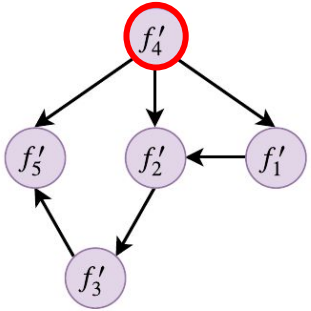
3, 2, 1...

Algorithme

Programme 1 (#M noeuds)



Programme 2 (#N noeuds)



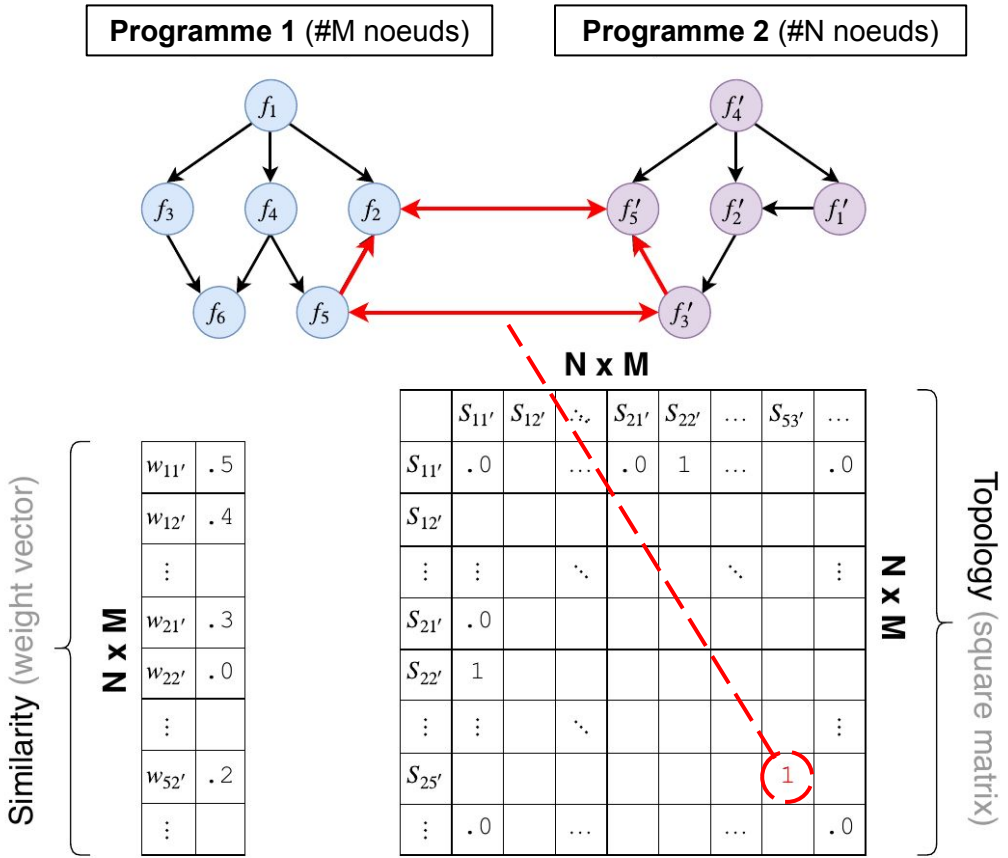
Similarity (weight matrix)

N

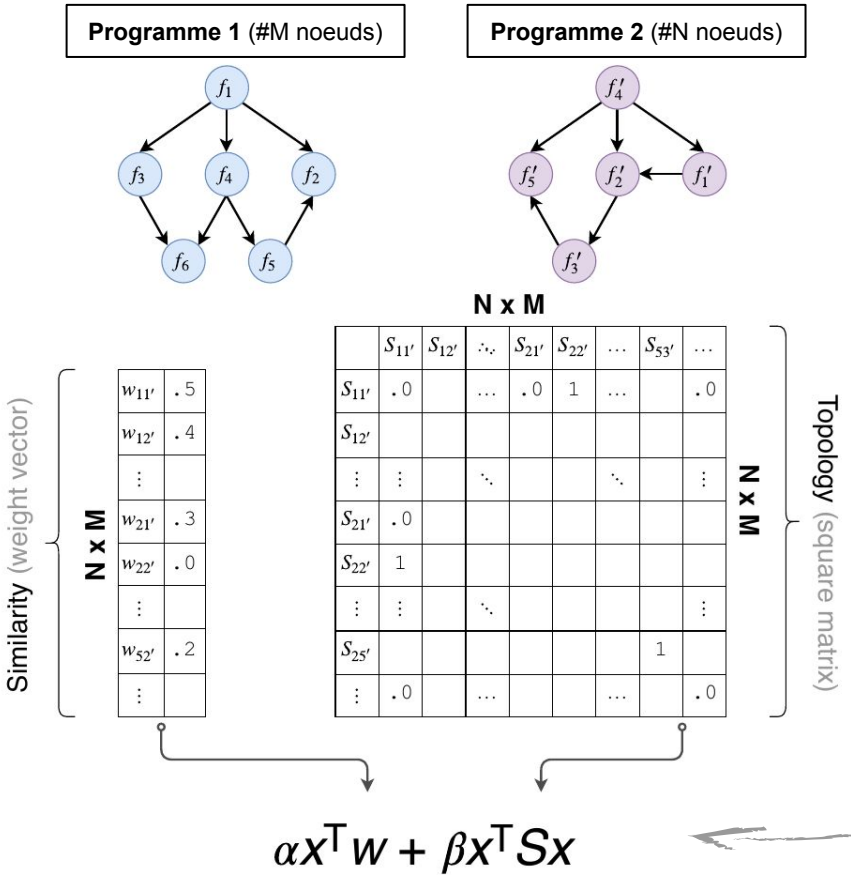
M

	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				

Algorithme


Programme 1 (#M noeuds)
Programme 2 (#N noeuds)
f_1 f_3 f_4 f_2 f_6 f_5
f'_4 f'_5 f'_2 f'_1 f'_3
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
N x M
Topology (square matrix)

Algorithme

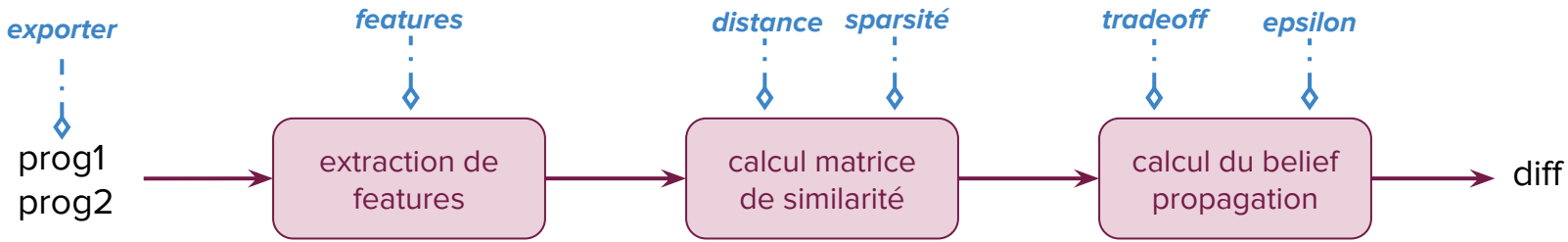


Objectif : **arbitrer** entre la **similarité des fonctions** et la **topologie du call-graph** afin d'être plus résilient si l'un d'entre eux est altéré *(+ utiliser les fonctions importées comme encres)*

Modulaire ?

Paramètres

- **exporter:** représentation exportée du binaire (Quokka, BinExport)
- **features**: 27 features utilisés pour décrire le programme *(certains en commun avec diaphora / bindiff)*
- **distance**: calculer la similarité entre features
- **sparsité**: pourcentage des matchs *(pertinents)* à conserver dans la matrice de similarité
- **tradeoff**: curseur pour arbitrer en la similarité des fonctions et la topologie du call-graph
- **epsilon**: paramètre de relaxation *(pour converger plus vite)*



Ablation

Objectif

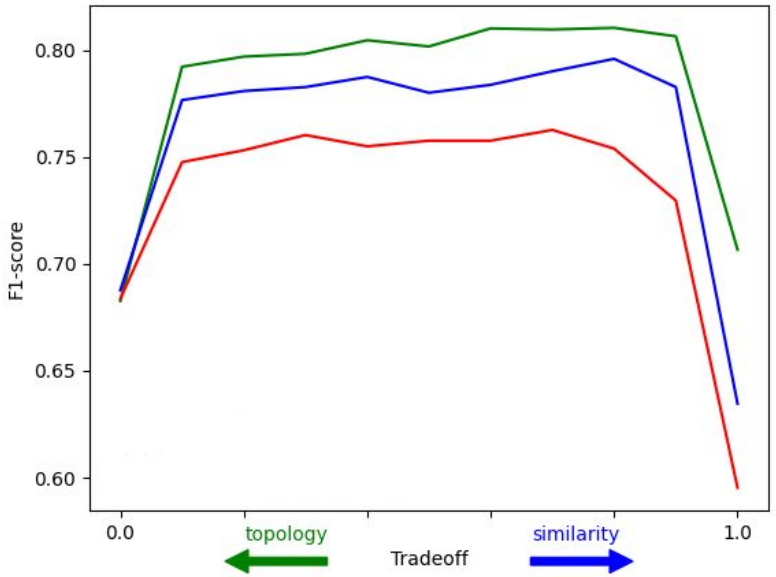
- Utiliser la modularité de QBinDiff pour obtenir de meilleurs résultats en fonction du cas d'usage *(comparé à BinDiff/Diaphora)*
- Trouver les meilleures features *(pour décrire une fonction)*
- Etudier les paramètres de QBinDiff et les fine-tuner
  - distance
  - tradeoff
  - sparsité
  - epsilon
- Trouver les meilleurs paramètres pour un projet donné *(ppb)*
- Etant donné un nouveau binaire, réutiliser des features et des paramètres bons en moyenne *(avb)*

Cadre expérimental

- Ablation réalisée sur un dataset à l'état-de-l'art [1]
- Six projets : `zlib`, `unrar`, `curl`, `clamav`, `nmap`, `openssl`
- Meilleur features pour chacun de ces projets *(ppb)*
- Trouver la meilleure configuration "par défaut" pour une utilisation future *(avb)*

[1] Marcelli and al. **How Machine Learning Is Solving the Binary Function Similarity Problem.** 2022

Features

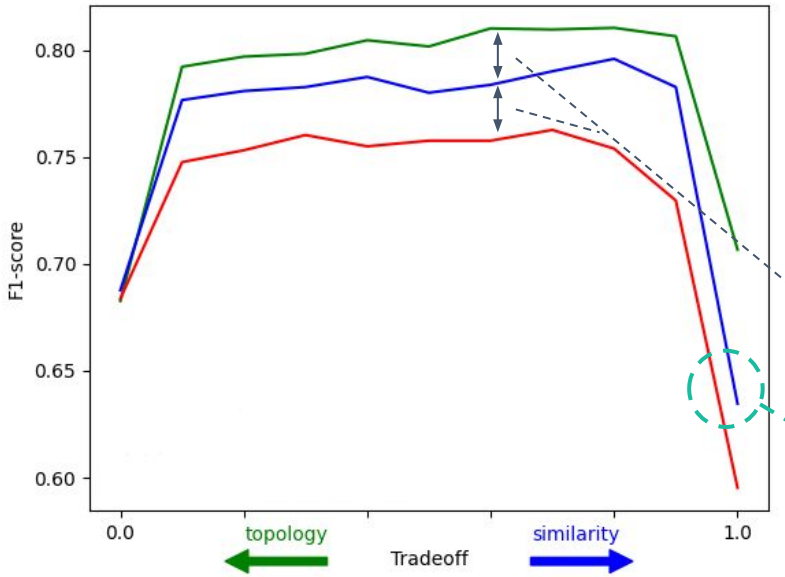


Effet des features *(f1-score)* sur le binaire **zlib** en fonction du tradeoff

Comment choisir de “bons” features ?

- 1er feature set: seulement des features liés aux **données** (#constantes)
- 2ème feature set: features de **données & CFG** (# noeuds du CFG)
- 3ème feature set: features de **données & CFG & CG** (#enfants d’une fonction)

Features



Effet des features *(f1-score)* sur le binaire **zlib** en fonction du tradeoff

Comment choisir de "bons" features ?

- 1er feature set: seulement des features liés aux **données** (#constantes)
- 2ème feature set: features de **données & CFG** (# noeuds du CFG)
- 3ème feature set: features de **données & CFG & CG** (#enfants d'une fonction)

Utiliser des features de nature différente

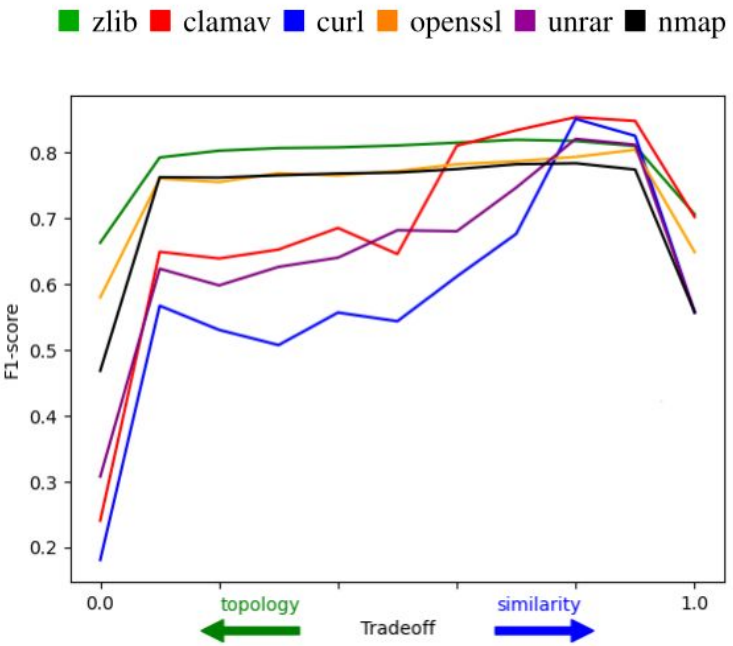
Inclure la structure du call-graph quand le curseur ne considère que la similarité

Tradeoff

Curseur entre similarité et topologie

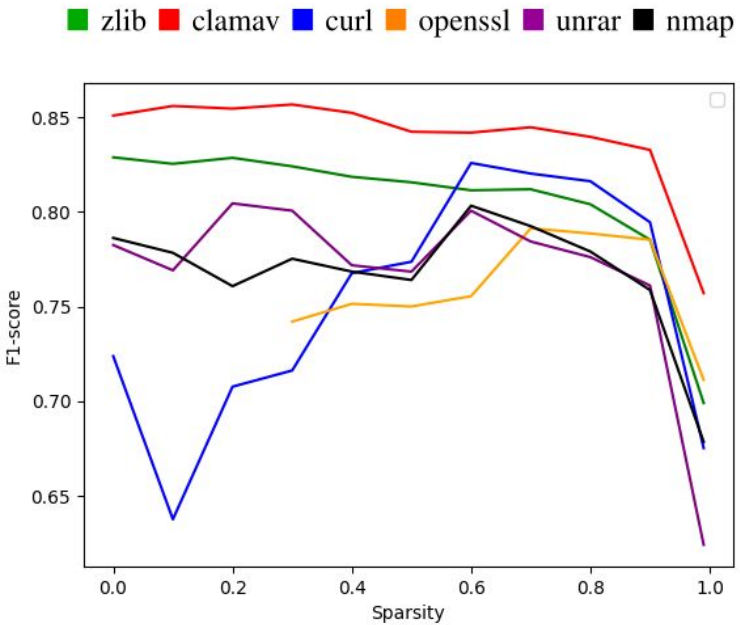
- Tradeoff ~ 1 : similarité des fonctions seulement
- Tradeoff ~ 0 : structure du call-graph seulement

- Tradeoff extrêmes (0 ou 1) à éviter
- Tradeoff > 0.6 permet de prendre en compte les fonctions complexes



Effet du tradeoff *(f1-score)* sur différents binaires

Sparsité



Effet de la sparsité *(f1-score)* sur différents binaires

Indice de sparsité

- Sparsité ~ 1 : seuls les matchs avec une très forte similarité seront conservés (restrictifs)
- Sparsité = 0 : tous les matchs candidats sont conservés

- Sur les petits programmes, une sparsité élevée décroît légèrement les performances
- Sur les larges programmes, une sparsité élevée donne de meilleures résultats grâce à une convergence plus rapide

Sparsité

- Augmenter la sparsité diminue le coût mémoire et le temps d'exécution
- Très avantageux sur les programmes assez lourds *(performances de diffing & temps/RAM)*

		clamav		nmap		openssl	
		Time	RAM	Time	RAM	Time	RAM
BinDiff		110	298	69	834	56	388
Diaphora3		151	29	651	30	265	30
QBinDiff	$s = 0$	672	823	8675	8339	-	-
	$s = 0.1$	689	801	8702	8329	-	-
	$s = 0.2$	684	712	7424	7167	-	-
	$s = 0.3$	639	617	6712	6171	3844	4884
	$s = 0.4$	656	523	4933	5071	3784	4022
	$s = 0.5$	541	430	4549	3902	2279	3121
	$s = 0.6$	516	347	3146	2900	1509	2301
	$s = 0.7$	489	272	2188	2009	1060	1569
	$s = 0.8$	461	204	1725	1294	1135	999
	$s = 0.9$	447	153	1205	762	528	535
	$s = 0.99$	440	134	880	673	398	408

Effet de la sparsité *(f1-score)* en terme de temps de calcul / coût mémore

Synthèse de l'ablation

Paramètres

- **features**: utiliser des features liés aux données, au CFG et au CG
- **distance**: Canberra ou Haussmann
- **tradeoff**: > 0.6, à tuner en fonction du bruit dans le binaire *(obfuscation)*
- **indice de sparsité**: le choisir grand pour de larges projets
- **epsilon**: > 0.8
- Meilleure combinaison spécifique à chaque projet **(ppb)** ou en moyenne **(avb)**

Exemple

Ligne de commande

```
$ 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

```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
```

⇒ **Résultats : CSV ou fichier .BinDiff**

Partie 3

Benchmarks

Quarkslab

Comparaison entre differs

		BinDiff	Diaphora3	QBinDiff-ppb (BinExport)	QBinDiff-ppb (Quokka)	QBinDiff-avb (BinExport)	QBinDiff-avb (Quokka)	GMN	Asm2vec	PalmTree	JTrans
zlib	libz.so.1.2.11	0.85	0.65	0.84	**0.89**	0.82	0.88	0.71	0.19	0.67	0.69
openssl	libssl.so.3	0.81	0.64	0.85	0.85	0.83	**0.86**	0.56	0.17	0.63	0.67
	openssl	0.95	0.68	0.96	**0.98**	0.92	**0.98**	0.59	0.54	0.76	0.72
	libcrypto.so.3	0.76	0.78	0.63	0.80	0.67	**0.82**	0.58	0.01	0.55	0.46
nmap	nping	0.59	0.52	0.74	**0.77**	0.73	**0.77**	0.17	0.17	0.41	0.52
	ncat	0.73	0.58	0.86	**0.92**	0.86	**0.92**	0.24	0.17	0.56	0.67
	nmap	0.8	0.8	0.73	**0.82**	0.73	**0.82**	0.66	0.10	0.61	0.43
clamav	libclamav	0.58	0.46	0.77	**0.81**	0.76	**0.81**	0.43	0.10	0.51	0.53
curl		0.65	0.56	0.83	**0.88**	0.83	**0.88**	0.24	0.22	0.50	0.57
unrar		0.68	0.62	0.82	**0.88**	0.81	0.87	0.22	0.14	0.57	0.69
Averaged		0.74	0.63	0.80	**0.86**	0.80	**0.86**	0.44	0.18	0.58	0.60

F1-score en fonction des differs & des projets

Comparaison entre differs

		BinDiff	Diaphora3	QBinDiff-ppb (BinExport)	QBinDiff-ppb (Quokka)	QBinDiff-avb (BinExport)	QBinDiff-avb (Quokka)	GMN	Asm2vec	PalmTree	JTrans
zlib	libz.so.1.2.11	0.85	0.65	0.84	**0.89**	0.82	0.88	0.71	0.19	0.67	0.69
openssl	libssl.so.3	0.81	0.64	0.85	0.85	0.83	**0.86**	0.56	0.17	0.63	0.67
	openssl	0.95	0.68	0.96	**0.98**	0.92	**0.98**	0.59	0.54	0.76	0.72
	libcrypto.so.3	0.76	0.78	0.63	0.80	0.67	**0.82**	0.58	0.01	0.55	0.46
nmap	nping	0.59	0.52	0.74	**0.77**	0.73	**0.77**	0.17	0.17	0.41	0.52
	ncat	0.73	0.58	0.86	**0.92**	0.86	**0.92**	0.24	0.17	0.56	0.67
	nmap	0.8	0.8	0.73	**0.82**	0.73	**0.82**	0.66	0.10	0.61	0.43
clamav	libclamav	0.58	0.46	0.77	**0.81**	0.76	**0.81**	0.43	0.10	0.51	0.53
curl		0.65	0.56	0.83	**0.88**	0.83	**0.88**	0.24	0.22	0.50	0.57
unrar		0.68	0.62	0.82	**0.88**	0.81	0.87	0.22	0.14	0.57	0.69
Averaged		0.74	0.63	0.80	**0.86**	0.80	**0.86**	0.44	0.18	0.58	0.60

F1-score en fonction des differs & des projets

Utiliser Quokka permet d'obtenir de meilleurs résultats *(meilleur export)*

Comparaison entre differs

		BinDiff	Diaphora3	QBinDiff-ppb (BinExport)	QBinDiff-ppb (Quokka)	QBinDiff-avb (BinExport)	QBinDiff-avb (Quokka)	GMN	Asm2vec	PalmTree	JTrans
zlib	libz.so.1.2.11	0.85	0.65	0.84	**0.89**	0.82	0.88	0.71	0.19	0.67	0.69
openssl	libssl.so.3	0.81	0.64	0.85	0.85	0.83	**0.86**	0.56	0.17	0.63	0.67
	openssl	0.95	0.68	0.96	**0.98**	0.92	**0.98**	0.59	0.54	0.76	0.72
	libcrypto.so.3	0.76	0.78	0.63	0.80	0.67	**0.82**	0.58	0.01	0.55	0.46
nmap	nping	0.59	0.52	0.74	**0.77**	0.73	**0.77**	0.17	0.17	0.41	0.52
	ncat	0.73	0.58	0.86	**0.92**	0.86	**0.92**	0.24	0.17	0.56	0.67
	nmap	0.8	0.8	0.73	**0.82**	0.73	**0.82**	0.66	0.10	0.61	0.43
clamav	libclamav	0.58	0.46	0.77	**0.81**	0.76	**0.81**	0.43	0.10	0.51	0.53
curl		0.65	0.56	0.83	**0.88**	0.83	**0.88**	0.24	0.22	0.50	0.57
unrar		0.68	0.62	0.82	**0.88**	0.81	0.87	0.22	0.14	0.57	0.69
Averaged		0.74	0.63	0.80	**0.86**	0.80	**0.86**	0.44	0.18	0.58	0.60

F1-score en fonction des differs & des projets

La similarité binaire (avec matching) n'arrive pas à concurrencer des differs standard beaucoup plus simples

Comparaison entre differs

		BinDiff	Diaphora3	QBinDiff-ppb (BinExport)	QBinDiff-ppb (Quokka)	QBinDiff-avb (BinExport)	QBinDiff-avb (Quokka)	GMN	Asm2vec	PalmTree	JTrans
zlib	libz.so.1.2.11	0.85	0.65	0.84	**0.89**	0.82	0.88	0.71	0.19	0.67	0.69
openssl	libssl.so.3	0.81	0.64	0.85	0.85	0.83	**0.86**	0.56	0.17	0.63	0.67
	openssl	0.95	0.68	0.96	**0.98**	0.92	**0.98**	0.59	0.54	0.76	0.72
	libcrypto.so.3	0.76	0.78	0.63	0.80	0.67	**0.82**	0.58	0.01	0.55	0.46
nmap	nping	0.59	0.52	0.74	**0.77**	0.73	**0.77**	0.17	0.17	0.41	0.52
	ncat	0.73	0.58	0.86	**0.92**	0.86	**0.92**	0.24	0.17	0.56	0.67
	nmap	0.8	0.8	0.73	**0.82**	0.73	**0.82**	0.66	0.10	0.61	0.43
clamav	libclamav	0.58	0.46	0.77	**0.81**	0.76	**0.81**	0.43	0.10	0.51	0.53
curl		0.65	0.56	0.83	**0.88**	0.83	**0.88**	0.24	0.22	0.50	0.57
unrar		0.68	0.62	0.82	**0.88**	0.81	0.87	0.22	0.14	0.57	0.69
Averaged		0.74	0.63	0.80	**0.86**	0.80	**0.86**	0.44	0.18	0.58	0.60

F1-score en fonction des differs & des projets

Parmi les differs, QBinDiff (peu importe l'exporter) donne les meilleures performances

Comparaison entre differs

		BinDiff	Diaphora3	QBinDiff-ppb (BinExport)	QBinDiff-ppb (Quokka)	QBinDiff-avb (BinExport)	QBinDiff-avb (Quokka)	GMN	Asm2vec	PalmTree	JTrans
zlib	libz.so.1.2.11	0.85	0.65	0.84	**0.89**	0.82	0.88	0.71	0.19	0.67	0.69
openssl	libssl.so.3	0.81	0.64	0.85	0.85	0.83	**0.86**	0.56	0.17	0.63	0.67
	openssl	0.95	0.68	0.96	**0.98**	0.92	**0.98**	0.59	0.54	0.76	0.72
	libcrypto.so.3	0.76	0.78	0.63	0.80	0.67	**0.82**	0.58	0.01	0.55	0.46
nmap	nping	0.59	0.52	0.74	**0.77**	0.73	**0.77**	0.17	0.17	0.41	0.52
	ncat	0.73	0.58	0.86	**0.92**	0.86	**0.92**	0.24	0.17	0.56	0.67
	nmap	0.8	0.8	0.73	**0.82**	0.73	**0.82**	0.66	0.10	0.61	0.43
clamav	libclamav	0.58	0.46	0.77	**0.81**	0.76	**0.81**	0.43	0.10	0.51	0.53
curl		0.65	0.56	0.83	**0.88**	0.83	**0.88**	0.24	0.22	0.50	0.57
unrar		0.68	0.62	0.82	**0.88**	0.81	0.87	0.22	0.14	0.57	0.69
Averaged		0.74	0.63	0.80	**0.86**	0.80	**0.86**	0.44	0.18	0.58	0.60

F1-score en fonction des differs & des projets

Peu de différence entre ppb et avb : on peut donc réutiliser les paramètres généraux et garantir de bons résultats

Conclusion

Takeaways

- QBinDiff dépasse les autres differs *(en modulant les paramètres)*
- Etude ablation : paramètres robustes, utilisables pour de nouveaux binaires
- Benchmark: comparaison entre differs standards et outils de similarité
- Similarité: l'efficacité de ces approches adaptées au diffing reste limitée *(entraînement coûteux, fonction seulement)*

⇒ Quarkslab utilise QbinDiff pour des malware obscurcis *(APT & co..)*

QBinDiff est open-source

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

Portail: diffing

https://diffing.quarkslab.com/

TL;DR Utiliser QbinDiff pour "tuner" et obtenir des résultats de diff plus précis

Thank you

Contact information:

Email: contact@quarkslab.com

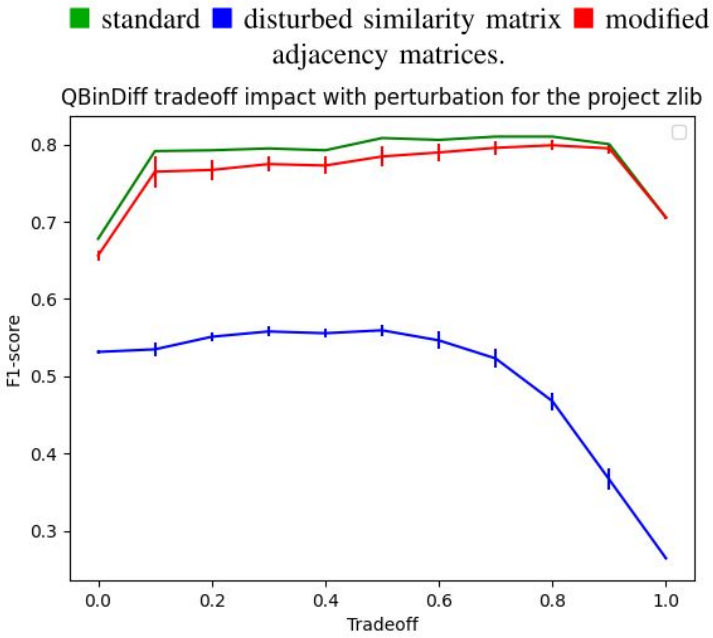
Phone: +33 1 58 30 81 51

Website: quarkslab.com

@quarkslab

Quarkslab

Tradeoff



Conséquence du bruit en fonction du tradeoff *(f1-score)* sur le binaire **zlib**

Curseur entre similarité et topologie

- Tradeoff ~ 1 : similarité des fonctions seulement
- Tradeoff ~ 0 : structure du call-graph seulement

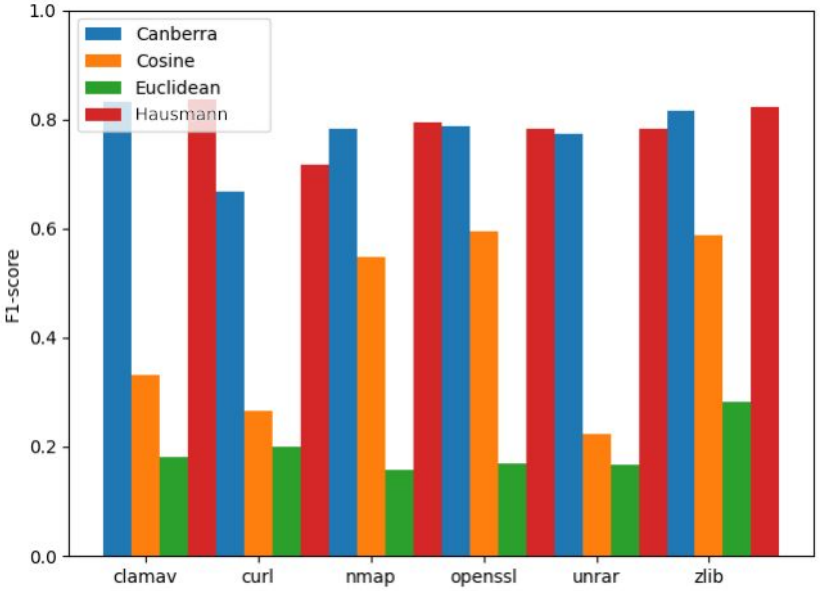
- Structure bruitée *(MH)* & tradeoff = 0 : performances dégradées
- Similarité bruitée *(bruit uniforme)* & tradeoff > 0.5 : performances dégradées
- Tradeoff = 0 ou tradeoff = 1 : manque d'informations sur le programme

Distance

A quoi sert la distance ?

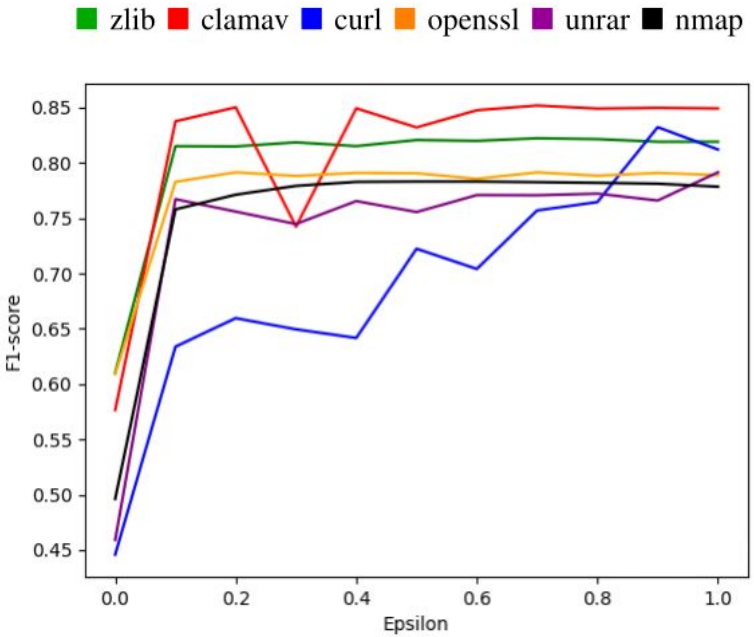
- Sert à **mesurer la similarité** entre différents features de fonctions
- **Haussmann:** Quarkslab new distance, combining Canberra and Jaccard-index

⇒ Les distances **Canberra** et **Haussmann** donnent les meilleurs résultats.



Effet de la distance *(f1-score)* sur différents binaires

Epsilon



Effet d'epsilon *(f1-score)* sur différents binaires

⇒ Epsilon aide à converger plus vite donc choisir une valeur > 0.8.