Binary Function Clone Search in Presence of Code Obfuscation and Optimization over Multi-CPU Architectures

ASIA CCS '23

Summary

- Binary Code Similarity Detection (BCSD) Model
  - Try to infer if two binaries are similar

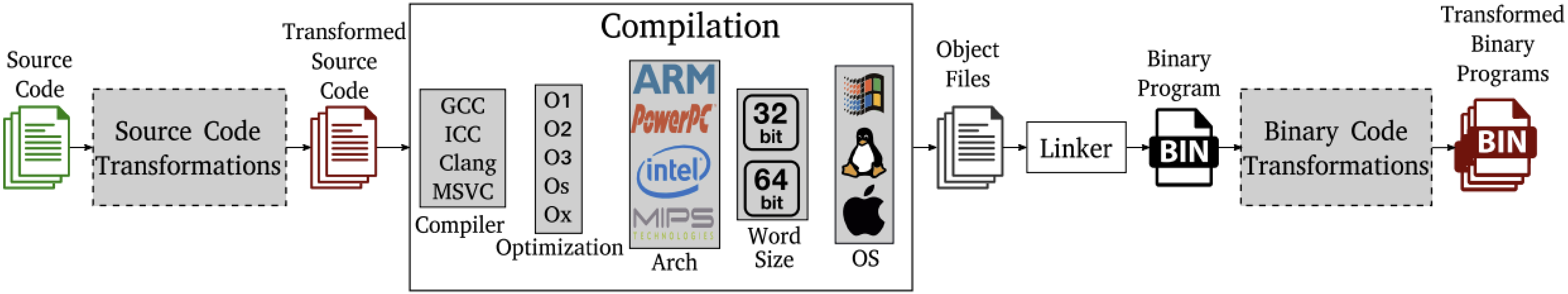
- Resilient Features
- To what?
  - Optimization
  - Obfuscation
  - Cross Architecture

Summary – cont.

- Why this paper
  - The paper evaluates obfuscation techniques including **tigress**

Background

Binary Compilation Pipeline



- Binary is generated via source code
- The final binary depends on the compilation process
- *Optimization (O0-O3)
- *Target Architecture

Irfan Ul Haq and Juan Caballero. 2021. A Survey of Binary Code Similarity. ACM Comput. Surv. 54, 3, Article 51 (April 2022), 38 pages. https://doi.org/10.1145/3446371

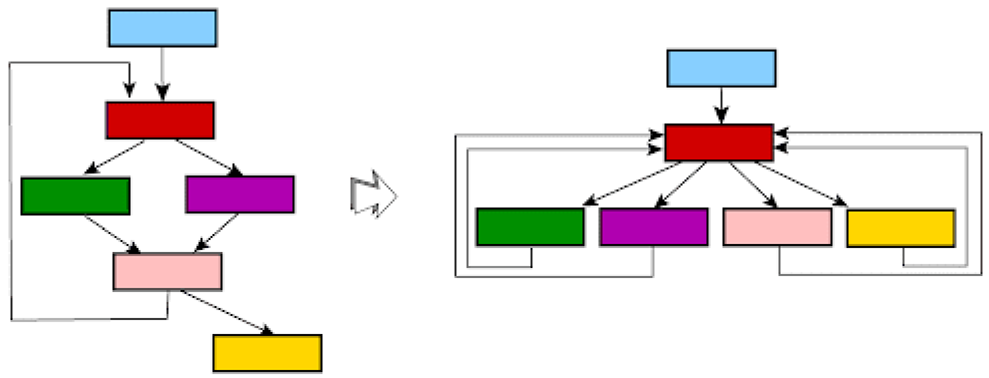
Obfuscation

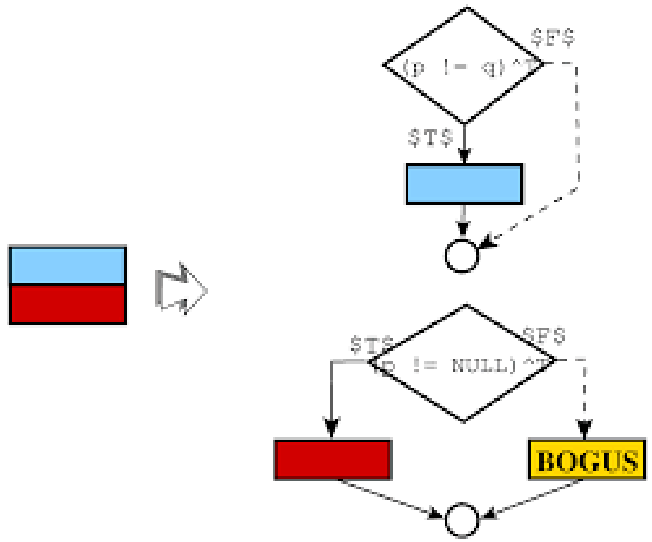
- Pre-Compilation / During Compilation
  - Post Compilation => Packing (out of scope)

- O-LLVM / Tigress

O-LLVM

- Using LLVM clang compiler
- Add new transformation pass (obfuscation)
    - Flattening (fla)
    - Bogus Control Flow (bcf)
    - Substitution (sub)





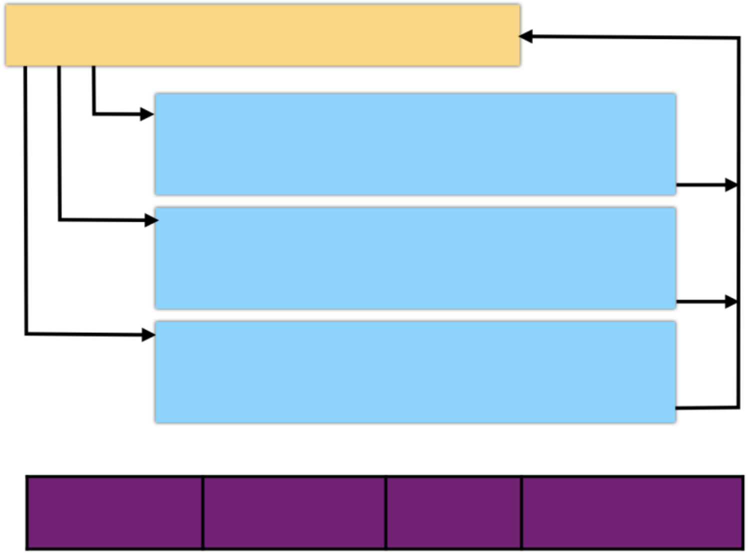
https://tigress.wtf/

Tigress

- Source to Source Obfuscator
  - Source IN Obfuscated Source OUT

- Provide MANY obfuscation transformations

- *Virtualization



Binary Code Similarity Detection

- One source code can result in infinitely many binaries
- “Similar” Binary => Binary that originates from the same source code

- Feature Extraction

- NLP

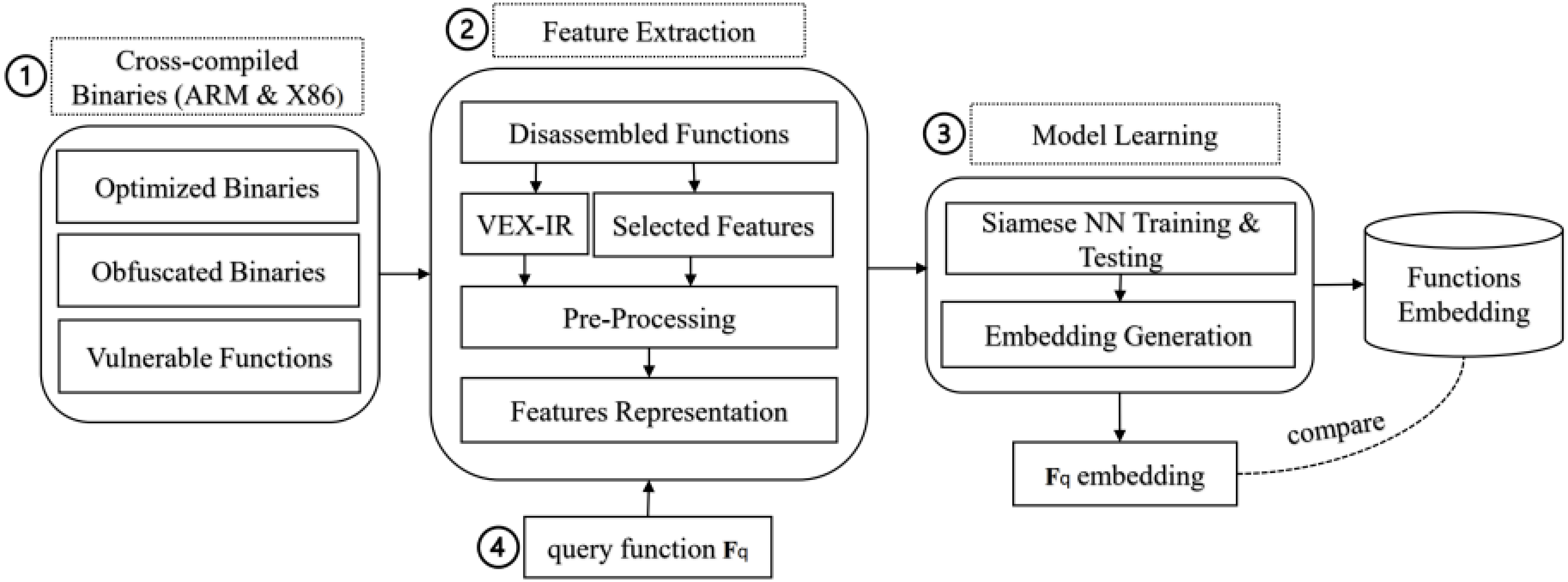


Figure 1: Overview of work flow of *BinFinder*

- Feature Extraction based BCSD model
- Feature -> Resilient Feature
  - Resilient against: Optimization, Obfuscation, Target Architecture

Resilient Features

- Num_callers
- Num_libc_callees
- Num_callees
- Num_unique_callees

- VEX IR Instructions
- LibcCalls
- Constants

Features	Code Obfuscations						Compilers Optimizations					
	O0_fla		*O0_sub*		*O0_bcf*		*O0_O1*		*O0_O2*		*O0_O3*	
	P(0)	*diff_mean*	*P(0)*	*diff_mean*	*P(0)*	*diff_mean*	*P(0)*	*diff_mean*	*P(0)*	*diff_mean*	*P(0)*	*diff_mean*
num_instructions	0.024	2188.365	0.105	615.22	0.027	3928.47	0.049	719.998	0.055	781.255	0.06	1055.09
num_basic block	0.014	42.219	0.37	5.297	0.036	27.731	0.352	5.548	0.361	5.528	0.363	5.578
num_Arithm	0.345	7.063	0.447	4.957	0.318	9.204	0.469	3.812	0.469	3.857	0.474	3.866
num_Logic	0.44	4.832	0.443	10.926	0.051	12.506	0.492	4.713	0.469	4.936	0.475	4.973
num_Cmp	0.283	29.971	0.394	7.143	0.277	14.342	0.407	7.352	0.405	7.471	0.41	7.525
num_ControlTrans	0.023	43.195	0.267	10.348	0.039	26.388	0.25	10.523	0.253	10.615	0.255	10.673
num_InOut	0.258	117.932	0.326	50.601	0.259	90.489	0.35	45.107	0.36	45.356	0.356	45.442
num_constants	0.091	15.464	0.29	3.391	0.133	2.763	0.324	2.228	0.327	2.294	0.326	2.322
num_callers	0.753	0.602	0.787	0.536	0.735	0.694	0.755	0.618	0.747	0.642	0.746	0.643
num_Libc_callees	0.923	0.158	0.932	0.144	0.888	0.224	0.912	0.194	0.912	0.2	0.91	0.203
num_callees	0.66	1.148	0.719	0.956	0.481	2.042	0.689	1.149	0.696	1.2	0.697	1.22
num_Unique_callees	0.764	0.488	0.785	0.46	0.73	0.535	0.764	0.524	0.773	0.528	0.774	0.528
unique_Vex_Instructions	0.314	0.221	0.314	0.221	0.314	0.221	0.106	0.214	0.107	0.218	0.111	0.231

Table 1: Empirical distribution of binary function features $P(0)_S$

- P(0): probability of a pair of similar binary functions to have the same targeted feature value
- Diff_mean: (absolute difference mean) indicates to which extent the selected feature value is affected

LibcCalls, Constants

- Several instances where dissimilar functions have the same num_libc_callee and ~~num_callee~~ (**num_constant** typo?)
  - But, different LibcCall, Constant

VEX IR Instructions

- Some instances (generally small fnc) exist where
  - No Constant
  - No LibcCall
- Lift asm to VEX IR through angr
  - VEX is an Intermediate Representation (lang in-between asm and source)
  - Angr is a binary analysis platform
- Normalize VEX IR
  - Register=REG, variable=TMP, Memory=MEM, number=CONST
  - Unique Normalized (take only unique instr)

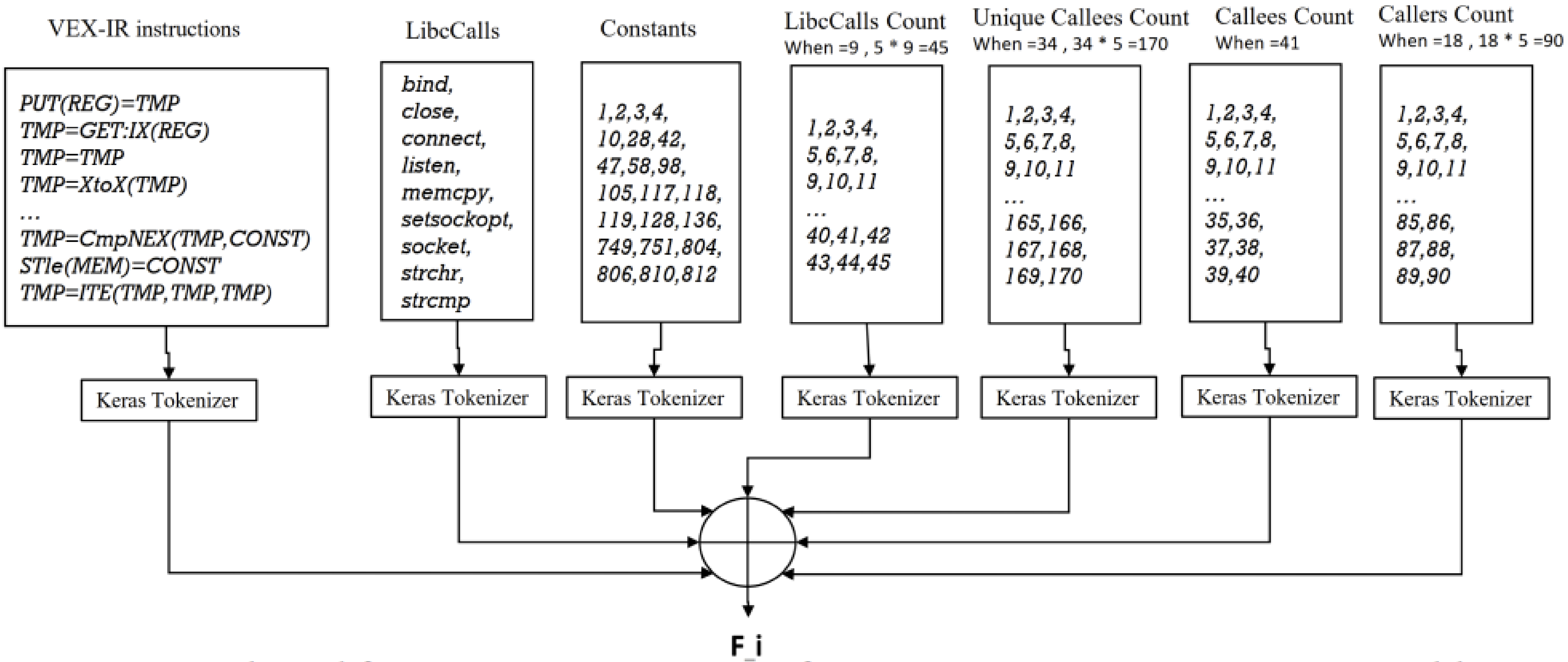


Figure 2: Selected features Representation of `BIO_get_accept_socket` in OpenSSL library

Model Architecture

- Siamese Neural Network

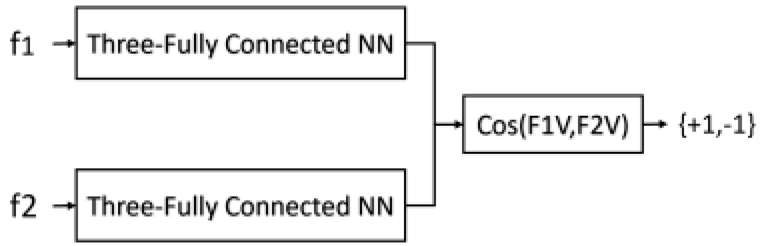


Figure 3: Siamese Neural Network Architecture

  - Twin Neural Network that shares weights
  - Output vector is compared (Cosine distance)

  - Often used for this kind of task

Dataset

(glibc, gmp, binutils, libcurl, openssl, ImageMagic, zlib)

- D-1 [gcc clang] | [O0-3 fla bcf sub] | [x86]
- D-2 [gcc clang] | [O0-3] [x86 ARM]
- D-3 [gcc clang] | [O0-3 fla bcf sub] | [x86 ARM]

(openssl, zlib, coreutils)

- D-4 [gcc] | [O0 O3] | [x86] | [Tigress]

BinKit

- D-5 [clang] | [O0-3 fla bcf sub]

Results

Vs O-LLVM

	gmp		OpenSSL		zlib		ImageMagic		Binutils		Coreutils		Findutils		Plotutils		Inetutils		Avg	
	M1	*M2*	*M1*	*M2*	*M1*	*M2*	*M1*	*M2*	*M1*	*M2*	*M1*	*M2*	*M1*	*M2*	*M1*	*M2*	*M1*	*M2*	*M1*	*M2*
O_0 Vs BCF	0.79	0.84	0.71	0.75	0.87	0.94	-	-	0.8	0.44	0.55	0.57	0.61	0.68	0.78	0.55	0.54	0.59	**0.71**	**0.67**
O_0 Vs FLA	0.92	0.84	0.77	0.72	0.92	0.92	0.8	0.79	0.86	0.86	0.93	0.92	0.84	0.85	1	0.78	0.93	0.94	**0.88**	**0.85**
O_0 Vs SUB	0.93	0.92	0.92	0.88	0.91	0.95	0.85	0.84	0.9	0.89	0.97	0.98	0.93	0.93	0.97	0.87	0.95	0.95	**0.92**	**0.91**
Avg	**0.88**	**0.87**	**0.8**	**0.78**	**0.9**	**0.94**	**0.825**	**0.818**	**0.84**	**0.66**	**0.75**	**0.76**	**0.75**	**0.78**	**0.88**	**0.69**	**0.74**	**0.77**	**0.84**	**0.81**

Table 2: Impact of obfuscation on *BinFinder* using precision at *top-1*

- M1: Trained tested on D-1 (only opt O0-3)
- M2: Trained tested on D-1 (both opt and obf)
- Test:
  - Query for every original binary in D-1 and few selected pkg in D-5 against its obfuscated variants
  - Query: (clang O0 x86) Look for (clang sub x86) (clang fla x86) (clang bcf x86)

Vs Tigress

	openssl				zlib				Coreutils				Avg			
	top-1		top-10		top-1		top-10		top-1		top-10		top-1		top-10	
	O0	O3	O0	O3	O0	O3	O0	O3	O0	O3	O0	O3	O0	O3	O0	O3
Add Opaque	0.4	0.58	0.56	0.77	0.89	0.44	1	0.59	0.13	0.17	0.2	0.27	**0.47**	**0.4**	**0.59**	**0.54**
EncodeArithmetic	0.42	0.52	0.772	0.78	0.54	0.35	0.84	0.76	0.71	0.43	0.91	0.81	**0.56**	**0.43**	**0.84**	**0.78**
EncodeLitrals	0.65	0.75	0.85	0.9	0.85	0.84	1	0.98	0.91	0.72	0.99	0.95	**0.8**	**0.77**	**0.95**	**0.94**
Flatten	0.08	0.45	0.14	0.63	0.19	0.75	0.43	0.96	0.21	0.66	0.29	0.84	**0.16**	**0.62**	**0.29**	**0.81**
virtualization	0.18	0.12	0.33	0.17	0.3	0.04	0.55	0.33	0.33	0.25	0.56	0.4	**0.27**	**0.14**	**0.48**	**0.3**
Avg	**0.346**	**0.48**	**0.53**	**0.65**	**0.55**	**0.48**	**0.76**	**0.72**	**0.45**	**0.45**	**0.59**	**0.65**	**0.45**	**0.47**	**0.63**	**0.68**

Table 4: Clone search between original binary functions and obfuscated ones by tigress

	UniqueVex		*#constants*		*constantsList*		*num_callers*		*num_libc_callees*		*num_callees*		*num_unique_callees*	
	O0	**O3**	**O0**	**O3**	**O0**	**O3**	**O0**	**O3**	**O0**	**O3**	**O0**	**O3**	**O0**	**O3**
Add Opaque	0.96	0.96	1	0.98	0.99	0.97	0.88	0.89	0.95	0.88	0.56	0.82	0.56	0.82
EncodeLiterals	0.7	0.88	0.96	0.93	0.93	0.93	0.91	0.84	0.97	0.96	0.84	0.9	0.95	0.94
Flatten	0.09	0.73	0.6	0.79	0.58	0.79	0.4	0.62	0.88	0.94	0.21	0.79	0.21	0.85
Virtualization	0.2	0.2	0.46	0.66	0.45	0.66	0.82	0.52	0.95	0.87	0.34	0.29	0.38	0.3
EncodeArithmetic	0.33	0.42	0.95	0.86	0.92	0.86	0.91	0.81	0.97	0.94	0.84	0.84	0.95	0.91

Table 5: Compiler optimizations effect against *tigress* obfuscation techniques over OPENSSL; each value represents P(0)

Vs Tigress

- Add Opaque – openssl, recall improves for O3
  - Manual analysis -> O3 removes junk callee

	UniqueVex		*#constants*		*constantsList*		*num_callers*		*num_libc_callees*		*num_callees*	
	O0	**O3**	**O0**	**O3**	**O0**	**O3**	**O0**	**O3**	**O0**	**O3**	**O0**	**O3**
Add Opaque	0.96	0.96	1	0.98	0.99	0.97	0.88	0.89	0.95	0.88	0.56	0.82

- Flatten – O3 performs better
  - Unique VEX

	UniqueVex	
	O0	**O3**
Add Opaque	0.96	0.96
EncodeLiterals	0.7	0.88
Flatten	0.09	0.73
Virtualization	0.2	0.2
EncodeArithmetic	0.33	0.42

Marcelli et al. USENIX '22

- Kind of benchmark for BCSD

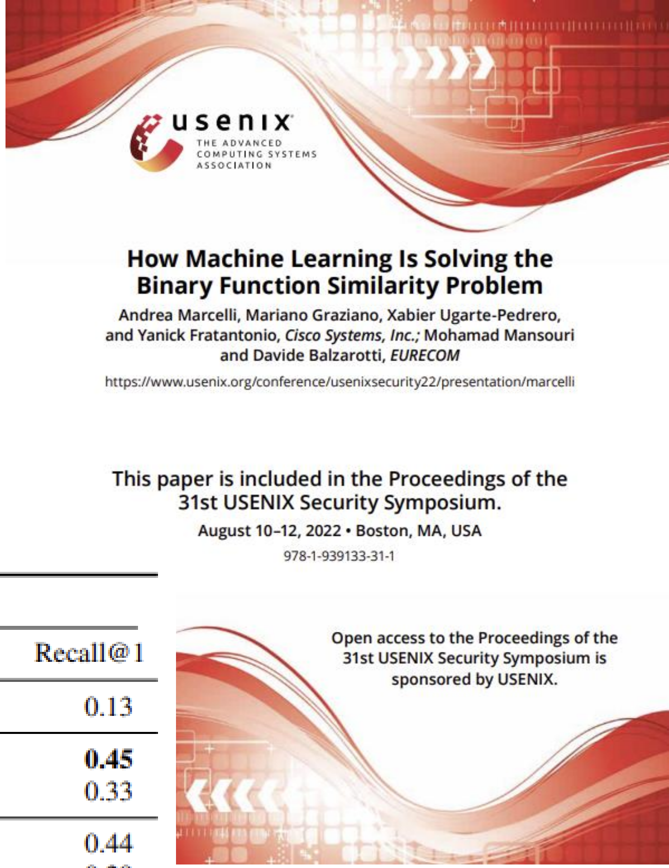


Table 3: Comparison of machine-learning models on Dataset-1.

						XM				
	Description	XC	XC+XB	XA	XM	small	medium	large	MRR10	Recall@1
[67] Zeek (direct comparison)	Strands	0.84	0.85	0.84	0.84	0.85	0.83	**0.87**	0.28	0.13
[40] GMN (direct comparison)	CFG + BoW opc 200	0.85	0.86	**0.86**	0.86	**0.89**	0.82	0.79	**0.53**	**0.45**
[40] GMN (direct comparison)	CFG + No features	**0.86**	**0.87**	**0.86**	**0.87**	0.88	**0.85**	0.84	0.43	0.33
[40] GNN	CFG + BoW opc 200	**0.86**	**0.87**	**0.86**	**0.87**	**0.89**	0.84	0.76	0.52	0.44
[40] GNN	CFG + No features	0.82	0.83	0.82	0.82	0.85	0.80	0.76	0.37	0.29
[76] GNN (s2v)	CFG + BoW opc 200	0.81	0.82	0.78	0.81	0.82	0.78	0.74	0.36	0.26
[76] GNN (s2v)	CFG + manual	0.81	0.82	0.80	0.81	0.84	0.77	0.79	0.36	0.28
[76] GNN (s2v)	CFG + No features	0.69	0.70	0.69	0.70	0.70	0.69	0.75	0.12	0.07
[45] w2v + AVG + GNN (s2v)	CFG + N. asm 150	0.79	0.79	0.74	0.77	0.78	0.75	0.73	0.24	0.16
[45] w2v + wAVG + GNN (s2v)	CFG + N. asm 150	0.79	0.79	0.76	0.77	0.78	0.76	0.76	0.29	0.20
[45] w2v + RNN + GNN (s2v)	CFG + N. asm 150	0.79	0.80	0.79	0.80	0.82	0.77	0.80	0.27	0.17
[49] w2v + SAFE	N. asm 150	0.80	0.81	0.80	0.81	0.83	0.77	0.77	0.17	0.07
[49] w2v + SAFE	N. asm 250	0.82	0.83	0.82	0.83	0.84	0.81	0.82	0.22	0.09
[49] w2v + SAFE + trainable	N. asm 150	0.80	0.81	0.80	0.81	0.83	0.76	0.74	0.29	0.16
[49] rand + SAFE + trainable	N. asm 150	0.79	0.80	0.79	0.80	0.83	0.75	0.74	0.28	0.17
[14] Asm2Vec	10 CFG random walks	0.77	0.69	0.60	0.65	0.63	0.70	0.78	0.12	0.07
[38] PV-DM	10 CFG random walks	0.77	0.70	0.50	0.62	0.63	0.62	0.61	0.11	0.08
[38] PV-DBOW	10 CFG random walks	0.78	0.70	0.50	0.63	0.63	0.62	0.61	0.11	0.09

Marcelli et al.

- Dataset-A and Dataset-B (represent different challenges)
  - Different compiler versions
  - Different optimization levels
  - Different CPU architectures
- Dataset-A:
  - Train and test
- Dataset-B:
  - Validate the resulting model on a miscellaneous and extensive group

Approach	AUC				XM				
	XC	XC+XB	XA	XM	small	medium	large	MRR10	Recall@1
BinFinder	**0.98**	**0.97**	**0.98**	**0.98**	**0.98**	**0.98**	**0.93**	**0.8**	**0.73**
GMN_OPC-200_e16	0.86	0.85	0.86	0.86	0.89	0.82	0.79	0.53	0.45
GNN-s2v_GeminiNN_OPC-200_e5	0.78	0.81	0.82	0.81	0.84	0.77	0.79	0.36	0.28
SAFE_ASM-list_e5	0.8	0.8	0.81	0.81	0.83	0.77	0.77	0.17	0.27
Zeek	0.84	0.84	0.85	0.84	0.85	0.83	0.87	0.28	0.13
asm2vec	0.62	0.81	0.74	0.69	0.63	0.7	0.78	0.12	0.07

Table 7: Comparison of state-of-the-art models with BinFinder on Dataset-A tasks

Approach	AUC			MRR10			Recall@1		
	XA	XA+XO	XO	XO	XA	XA+XO	XO	XA	XA+XO
BinFinder	**0.99**	0.95	0.96	**0.77**	0.83	0.68	**0.72**	**0.77**	0.58
GMN_OPC-200_e16	0.98	**0.96**	**0.97**	0.75	**0.84**	**0.71**	0.66	**0.77**	**0.61**
GNN-s2v_GeminiNN_GeminiFeatures_e5	0.96	0.93	0.93	0.57	0.74	0.57	0.47	0.63	0.49
SAFE_ASM-list_e5	0.9	0.88	0.88	0.27	0.3	0.31	0.14	0.17	0.19
Trex	0.94	0.94	0.94	0.61	0.50	0.53	0.5	0.37	0.46
Zeek	0.94	0.91	0.92	0.41	0.45	0.36	0.28	0.30	0.21
asm2vec_e10	0.69	0.75	0.94	0.60	0.06	0.22	0.49	0.015	0.17

Table 8: Comparison of state-of-the-art models with BinFinder on Dataset-B tasks

- XO: different opt. same compiler, compiler version, arch.
- XC: different compiler compiler version and opt. same arch and bitness
- XC+XB: different compiler, compiler version, opt, and bitness. Same arch
- XA: function pair have different arch and bitness but same compiler, compiler version, opt
- XA+XO: function pair have different arch, bitness, and opt. Same compiler and compiler version
- XM: function pair comes from arbitrary arch, bitness, opt, compiler, compiler version.
- XM-S/M/L : size of fnc

Cont.

- Classification (Similar? Dissimilar?)
- AUC: aggregate measure of the performance of a model across all possible classification threshold

- Ranking (top similar)
- MRR (mean reciprocal rank)
- Recall@K