Model Checking for Mobile Android Malware Evolution

Aniello Cimitile, Fabio Martinelli, Francesco Mercaldo, Vittoria Nardone, Antonella Santone, Gigliola Vaglini

{fabio.martinelli, francesco.mercaldo}@iit.cnr.it
Institute for Informatics and Telematics, National Research Council of Italy (CNR)

{cimitile, vnardone,santone}@unisannio.it
Department of Engineering, University of Sannio, Italy

gigliola.vaglini@unipi.it
Department of Information Engineering, University of Pisa, Italy

Software Evolution

User needs & The environment change

Malware, as any software, Evolves

User needs:
- to evade detection
- new threats

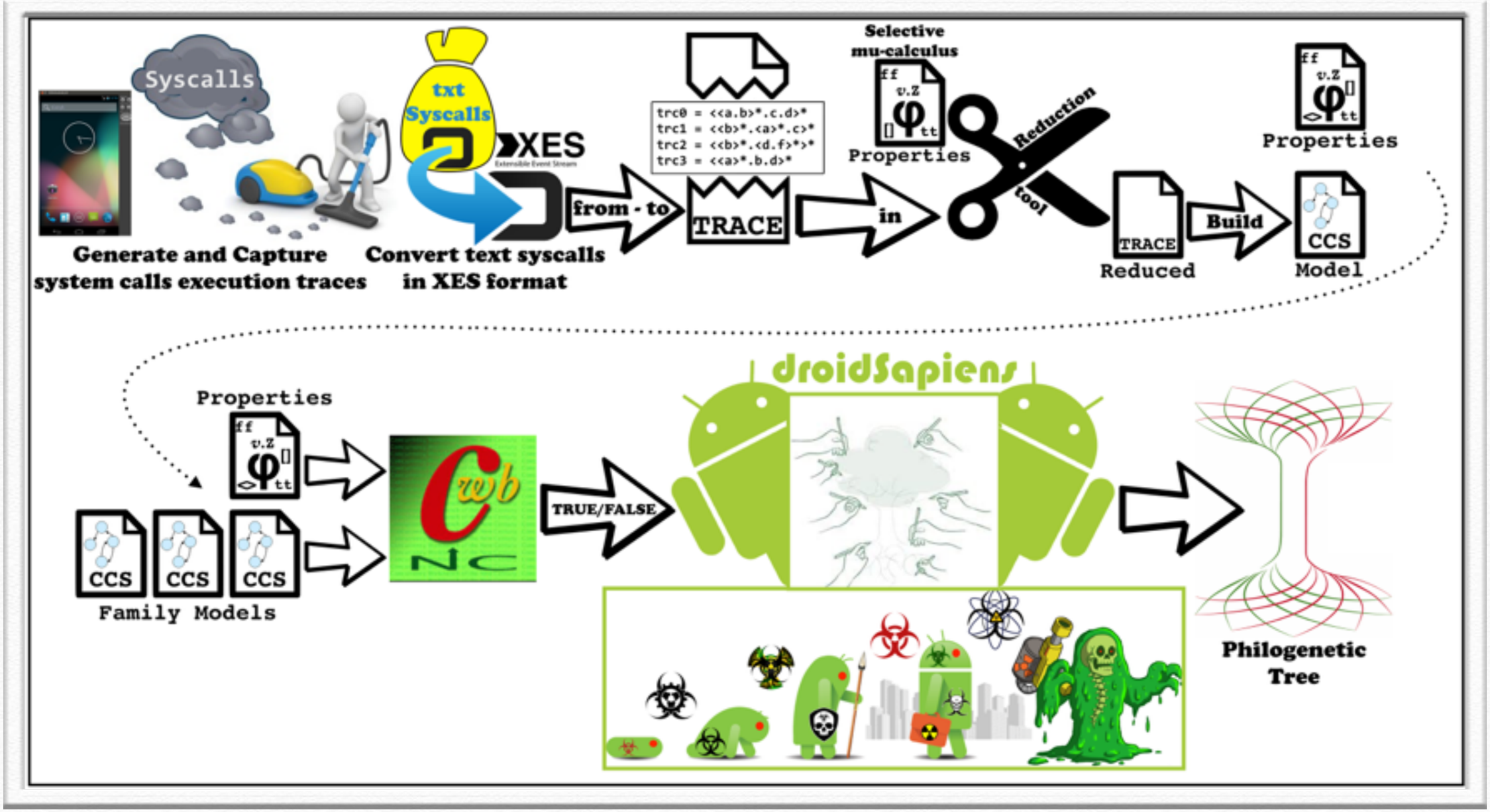
&

The environment change

Motivation

- To propose a novel approach that use temporal logic formula to infer malware evolution.
- To demonstrate that Android malware is not developed by zero
- To propose an useful method to malware analysts to predict future threats.
- To contribute to the current mobile malware research by pointing to the evolution of possible vulnerabilities concerning the Android platform.

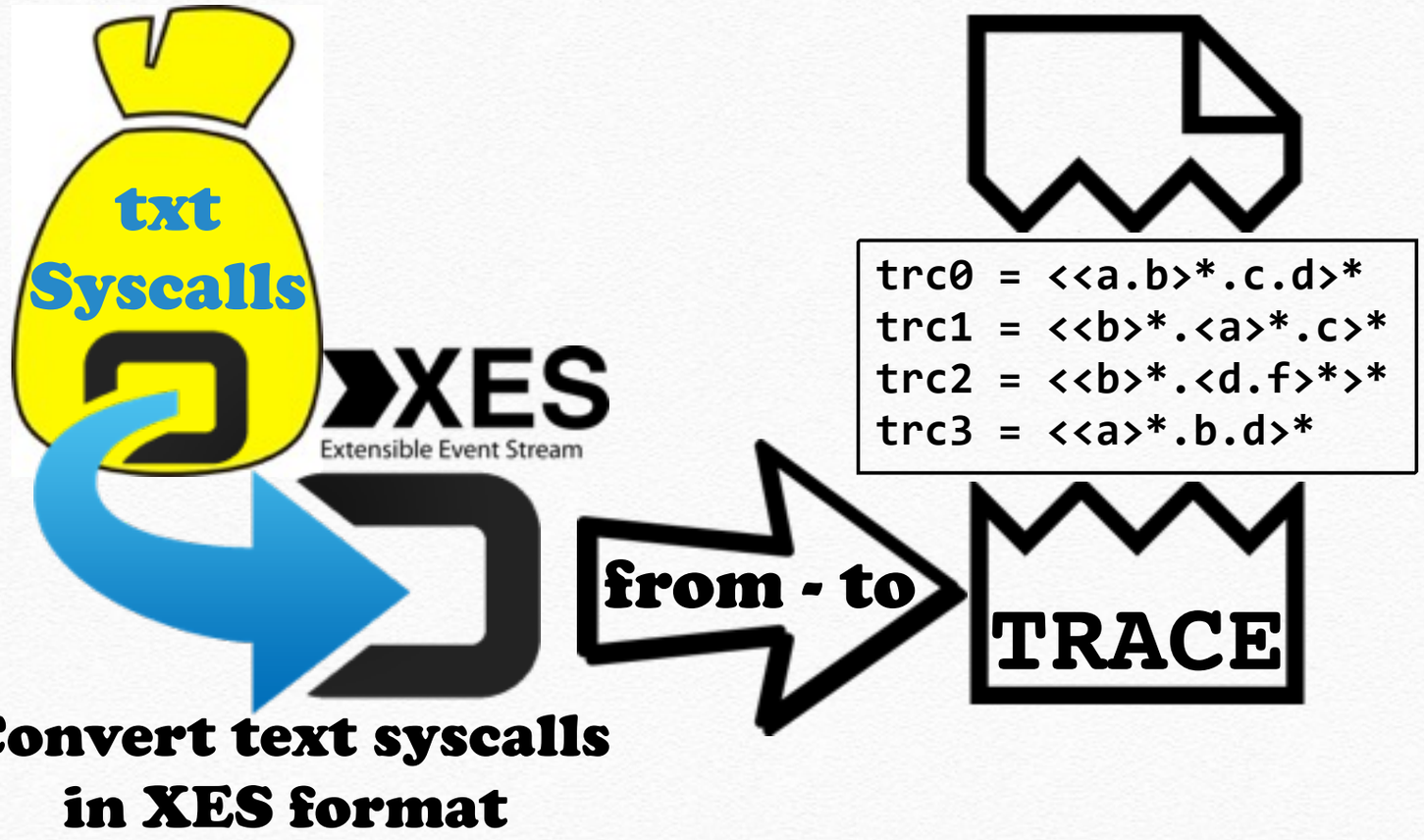
Our Approach



Process 1: System Call Extraction

- The APK is installed and started on an Android Device Emulator
- BOOT_COMPLETED event is generated
- The corresponding sequence of system call is gathered in a textual format

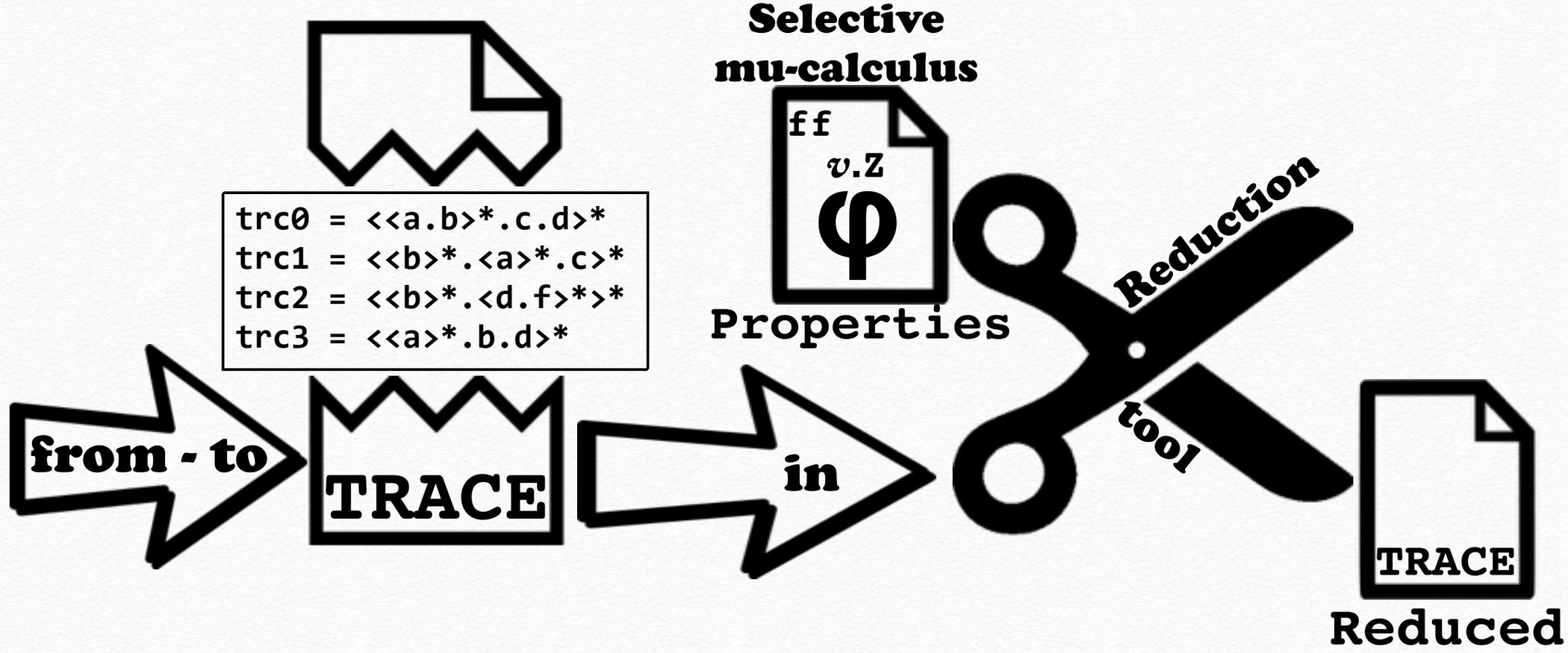
Process 2: XES-based Event Stream Generation



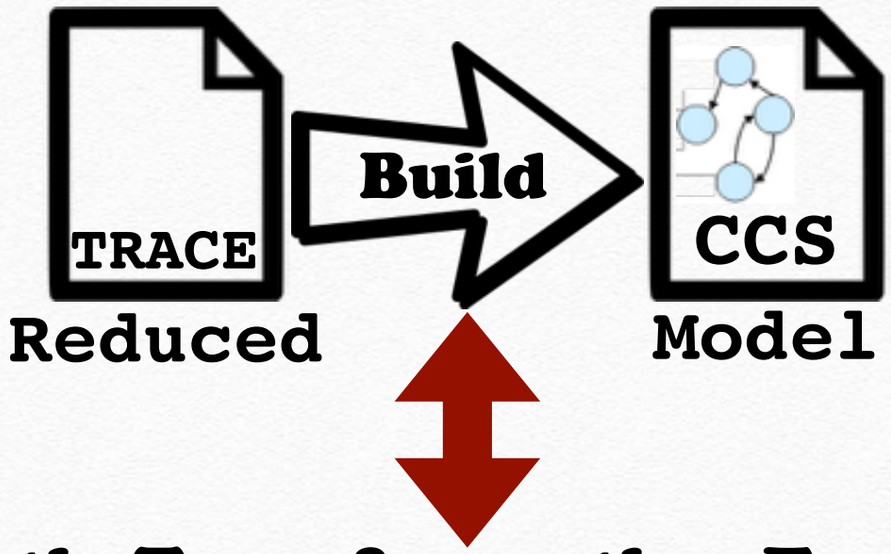
<u>syntax</u>: t::=e|t.t| <t>* |λ
where **e** ∈ **A** and λ is the empty sequence.

The operator "." represents trace concatenation.
The operator "*" represents the iteration of a trace.

Process 3: Property Based Reduction



Process 4: Model Discovery



Syntactic Transformation Function $\mathcal{T}$

- $\mathcal{F}irst(T) = \{e \mid e.t \in T \text{ or } < e.t >^* .t' \in T\}$
- $\mathcal{R}est(T) = \{t \mid e.t \in T\} \cup \{t_1 . < e.t_1 >^* .t_2 \mid < e.t_1 >^* .t_2 \in T\}$
- $Cont(T) = \{t_2 \mid < t_1 >^* .t_2 \in T\}$

Process 5: Formal Analysis of Malware Evolution

droidSapiens

considers the family **X** as "ancestor" of the family **Y** if the formula φ_X, characterizing the family **X**, is TRUE on more than the 35% of the apps belonging to **Y**.

The Dataset

Family	*#samples*	*date*
Geinimi	73	12-2010
Plankton	81	06-2011
DroidKungFu	183	08-2011
Opfake	423	2013
FakeInstaller	98	2014

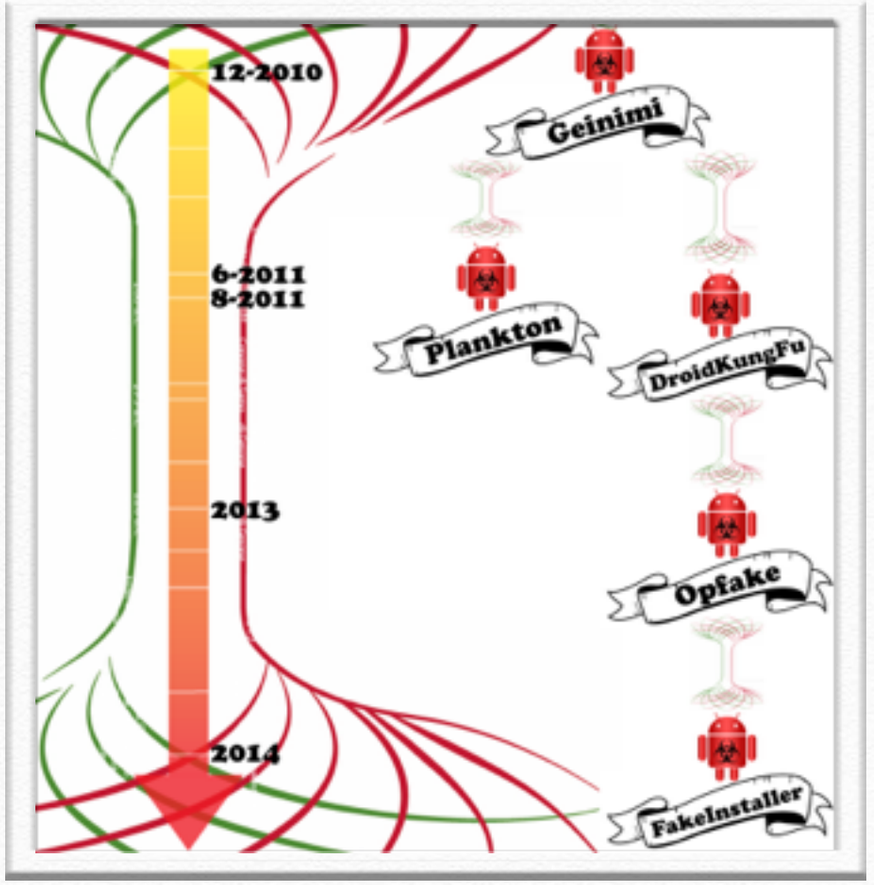
858 sample
5 malware families

We retrieved the Android malware applications from both **Genoma**[1] and **Drebin**[2] dataset

[1] Y. Zhou and X. Jiang. Dissecting android malware: Characterization and evolution. In Security and Privacy (SP), 2012 IEEE Symposium on, pages 95–109. IEEE, 2012

[2] D. Arp, M. Spreitzenbarth, M. Huebner, H. Gascon, and K. Rieck. Drebin: Efficient and explainable detection of android malware in your pocket. In NDSS, 2014.

Preliminary Results



Family (Number of apps) / Formulae	Geimini (73)	Plankton (81)	DroidKungFu (183)	OpFake (423)	FakeInstaller (98)
φ_G	**60 (82%)**	38 (46%)	72 (39%)	135 (31%)	14 (14%)
φ_P	5 (6%)	**54 (66%)**	12 (6%)	20 (4%)	13 (13%)
φ_{DKF}	8 (11%)	13 (16%)	**145 (79%)**	155 (36%)	16(16%)
φ_{OF}	20 (27%)	23 (28%)	55 (30%)	**229 (54%)**	45 (45 %)
φ_{FI}	18(24%)	15 (18%)	51 (27%)	140(33%)	**51 (52%)**

Further Evaluation

Formulae \ Family (Number of apps)	Geimini (73)	Plankton (81)	DroidKungFu (183)	OpFake (423)	FakeInstaller (98)
$\varphi_G \vee \varphi_P$	65 (89%)	59 (72%)	81 (44%)	136 (32%)	27 (27%)
$\varphi_G \vee \varphi_{DKF}$	65 (89%)	50 (61%)	152 (83%)	286 (67%)	30 (30%)
$\varphi_{DKF} \vee \varphi_{OF}$	28 (38%)	30 (37%)	151 (82%)	250 (59%)	60 (61%)
$\varphi_{OF} \vee \varphi_{FI}$	21 (28%)	23 (28%)	61 (33%)	248 (58%)	51 (52%)
$\varphi_P \vee \varphi_{DKF}$	13 (17%)	65 (80%)	150 (81%)	173 (40%)	16 (16%)
$\varphi_P \vee \varphi_{OF}$	21 (28%)	62 (76%)	65 (35%)	230 (54%)	58 (59%)
$\varphi_P \vee \varphi_{FI}$	19 (26%)	60 (74%)	61 (33%)	141 (33%)	64 (65%)
$\varphi_G \vee \varphi_{DKF} \vee \varphi_{OF} \vee \varphi_{FI}$	70 (95%)	57 (70%)	160 (87%)	335 (79%)	76 (77%)

We combine the specified formulae to validate the inferred phylogenetic tree

Further Evaluation

Formulae \ Family (Number of apps)	Geimini (73)	Plankton (81)	DroidKungFu (183)	OpFake (423)	FakeInstaller (98)
$\varphi_G \vee \varphi_P$	65 (89%)	59 (72%)	81 (44%)	136 (32%)	27 (27%)
$\varphi_G \vee \varphi_{DKF}$	65 (89%)	50 (61%)	152 (83%)	286 (67%)	30 (30%)
$\varphi_{DKF} \vee \varphi_{OF}$	28 (38%)	30 (37%)	151 (82%)	250 (59%)	60 (61%)
$\varphi_{OF} \vee \varphi_{FI}$	21 (28%)	23 (28%)	61 (33%)	248 (58%)	51 (52%)
$\varphi_P \vee \varphi_{DKF}$	13 (17%)	65 (80%)	150 (81%)	173 (40%)	16 (16%)
$\varphi_P \vee \varphi_{OF}$	21 (28%)	62 (76%)	65 (35%)	230 (54%)	58 (59%)
$\varphi_P \vee \varphi_{FI}$	19 (26%)	60 (74%)	61 (33%)	141 (33%)	64 (65%)
$\varphi_G \vee \varphi_{DKF} \vee \varphi_{OF} \vee \varphi_{FI}$	70 (95%)	57 (70%)	160 (87%)	335 (79%)	76 (77%)

ancestor \/ descendant

Further Evaluation

Formulae \ Family (Number of apps)	Geimini (73)	Plankton (81)	DroidKungFu (183)	OpFake (423)	FakeInstaller (98)
$\varphi_G \vee \varphi_P$	65 (89%)	59 (72%)	81 (44%)	136 (32%)	27 (27%)
$\varphi_G \vee \varphi_{DKF}$	65 (89%)	50 (61%)	152 (83%)	286 (67%)	30 (30%)
$\varphi_{DKF} \vee \varphi_{OF}$	28 (38%)	30 (37%)	151 (82%)	250 (59%)	60 (61%)
$\varphi_{OF} \vee \varphi_{FI}$	21 (28%)	23 (28%)	61 (33%)	248 (58%)	51 (52%)
$\varphi_P \vee \varphi_{DKF}$	13 (17%)	65 (80%)	150 (81%)	173 (40%)	16 (16%)
$\varphi_P \vee \varphi_{OF}$	21 (28%)	62 (76%)	65 (35%)	230 (54%)	58 (59%)
$\varphi_P \vee \varphi_{FI}$	19 (26%)	60 (74%)	61 (33%)	141 (33%)	64 (65%)
$\varphi_G \vee \varphi_{DKF} \vee \varphi_{OF} \vee \varphi_{FI}$	70 (95%)	57 (70%)	160 (87%)	335 (79%)	76 (77%)

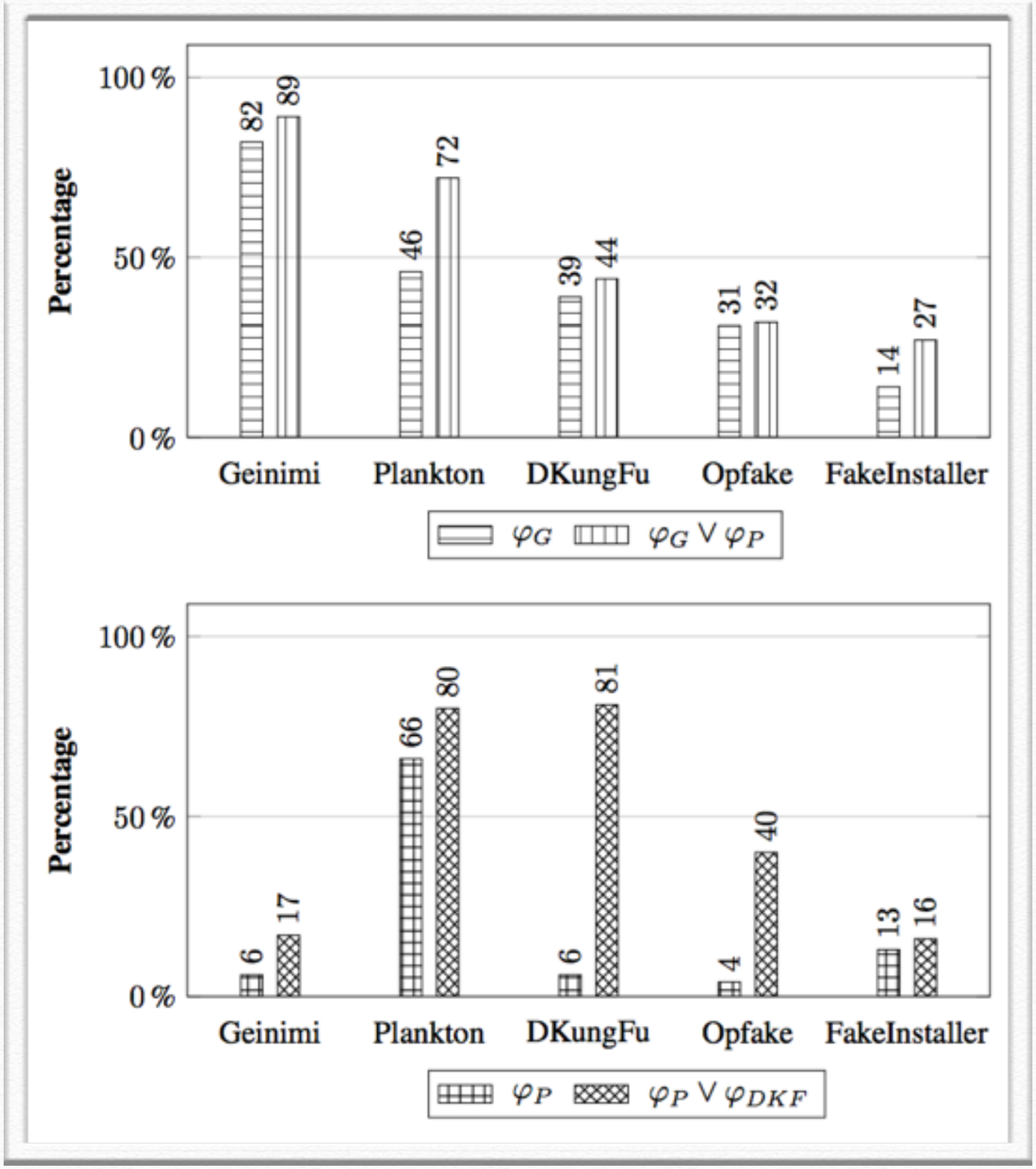
no relation found

Further Evaluation

Formulae \ Family (Number of apps)	Geimini (73)	Plankton (81)	DroidKungFu (183)	OpFake (423)	FakeInstaller (98)
$\varphi_G \vee \varphi_P$	65 (89%)	59 (72%)	81 (44%)	136 (32%)	27 (27%)
$\varphi_G \vee \varphi_{DKF}$	65 (89%)	50 (61%)	152 (83%)	286 (67%)	30 (30%)
$\varphi_{DKF} \vee \varphi_{OF}$	28 (38%)	30 (37%)	151 (82%)	250 (59%)	60 (61%)
$\varphi_{OF} \vee \varphi_{FI}$	21 (28%)	23 (28%)	61 (33%)	248 (58%)	51 (52%)
$\varphi_P \vee \varphi_{DKF}$	13 (17%)	65 (80%)	150 (81%)	173 (40%)	16 (16%)
$\varphi_P \vee \varphi_{OF}$	21 (28%)	62 (76%)	65 (35%)	230 (54%)	58 (59%)
$\varphi_P \vee \varphi_{FI}$	19 (26%)	60 (74%)	61 (33%)	141 (33%)	64 (65%)
$\varphi_G \vee \varphi_{DKF} \vee \varphi_{OF} \vee \varphi_{FI}$	70 (95%)	57 (70%)	160 (87%)	335 (79%)	76 (77%)

ancestor-descendant line tree

Comparison between formulae



Time Verification

Family	T_{ex}	T_{mod}	T_{chk}	T_{TOT}
Geinimi	4380	2.173	3.386	4385.559
Plankton	4860	1.386	2.481	4863.867
DroidKungFu	10980	7.791	8.141	10995.932
Opfake	25380	2.34	11.576	25393,916
FakeInstaller	5880	1.266	2.403	5883.669

- **Tex** is the time employed to retrieve system calls (i.e., 60 seconds for each application)
- **Tmod** is the time required to build the model
- **Tchk** is the time to verify the properties.
- **TTOT** value is the sum of all these contributes.

Remarks and Future Works

- We use model checking in order to investigate Android malware evolution. We build the phylogenetic tree identifying the ancestor and the descendant between mobile malware families.
- We obtain encouraging results and they suggest that the approach is remarkably accurate.
- As future work we intend to investigate the use of the k-bsimulation to measure the similarity among malware families.
- Furthermore, we intend to investigate the multiple ancestors.

Thanks for your attention

We are grateful for receiving comments, observations, suggestions, and collaborations with other research groups which could improve our research.