A Presentation on

MimeoDroid: Large Scale Dynamic App analysis on Cloned Devices via Machine Learning Classifiers

Presented by: Akka Zemmari

Parvez Faruki, Akka Zemmari, Vijay Laxmi, M S Gaur & Mauro Conti

The 3rd IEEE International Workshop on Reliability & Security Data Analysis
Toulouse, France

June 28-30, 2016

Outline

Outline

Smartphone and Smart Devices

Smartphone Growth

- Gartner reports Smartphone accounts 57% of the mobile phone sales in Q3_2015, Android leads with 82% share. [a] [b] [c]
- Android devices touching 1 billion with 1.5m new activations per month![d]
- Fuelled by cost reduction and silicon Integration for IoTs.
- Android app market likely to grow $70 Billion in 2017 from existing $25 billion.[e]

---

[a] "*State of Mobile App Security*" - Volume 3, November 2015, ArXan Research

[b] "*KINDSIGHT SECURITY LABS*" Q3_2015, Android Malware Growth

[c] "*Gartner Annual Smartphone Sales Report 2015*" - http://www.gartner.com/newsroom/id/2665715

[d] "*State of Mobile App Security*" - Volume 3 November 2015, ArXan Research

[e] "*Gartner Says Annual Smartphone Sales Surpassed Sales of Feature Phones for the First Time in 2013*" - http://www.gartner.com/newsroom/id/2665715

Malware Threat Landscape

Android Malware threat

- Increasing exponentially [3, 5, 9]
- Prevalence of obfuscation, code protection and evasion techniques.
- Signature database exploding.
- Repackaged malware signatures increasing rapidly [8, 9].

Malware Threat Landscape

Android Malware threat

- Increasing exponentially [3, 5, 9]
- Prevalence of obfuscation, code protection and evasion techniques.
- Signature database exploding.
- Repackaged malware signatures increasing rapidly [8, 9].

Analysis Techniques are vulnerable against

- Android malware evading analysis Sandbox.
- Evolving malicious apps [2] armed with anti-analysis environment.
- Covert malicious behavior evading the anti-malware.

Malware Threat Landscape

Android Malware threat

- Increasing exponentially [3, 5, 9]
- Prevalence of obfuscation, code protection and evasion techniques.
- Signature database exploding.
- Repackaged malware signatures increasing rapidly [8, 9].

Analysis Techniques are vulnerable against

- Android malware evading analysis Sandbox.
- Evolving malicious apps [2] armed with anti-analysis environment.
- Covert malicious behavior evading the anti-malware.

Why MimeoDroid ?

- Automated dynamic analysis with machine learning improves analysis coverage [6, 7, 1].

Android Malware Statistics

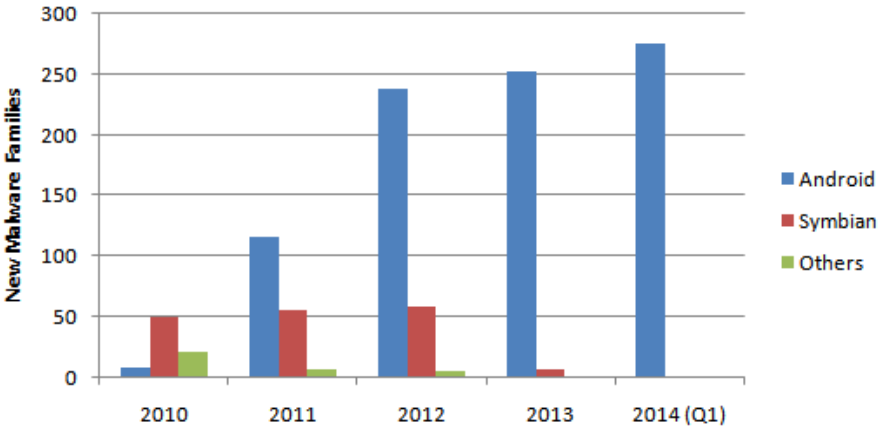


Figure: Source: F-Secure

Android Malware Statistics

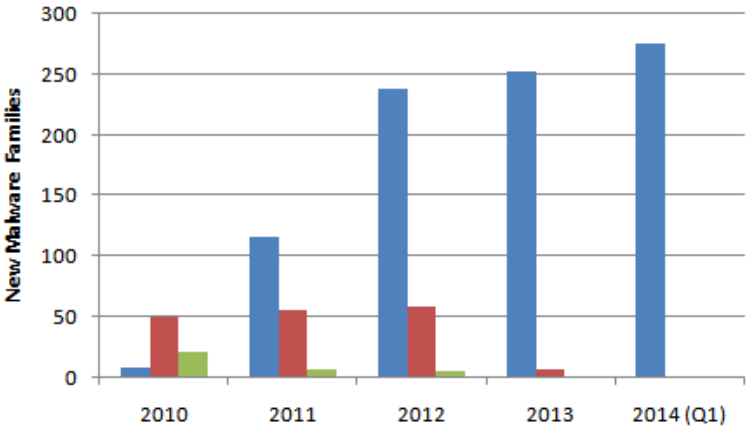


Figure: Source: F-Secure

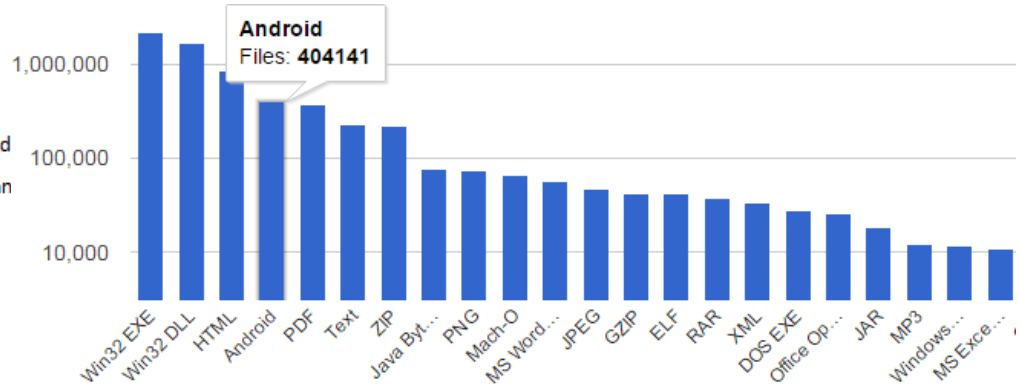


Figure: Submissions @ VirusTotal: June 18-24, 2016

Android Malware Statistics

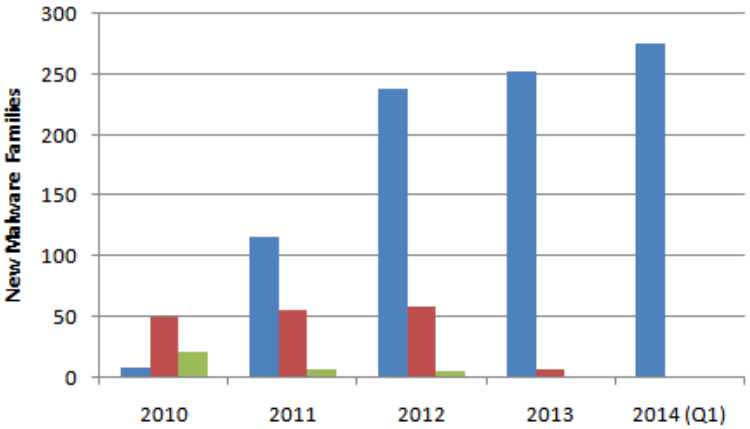


Figure: Source: F-Secure

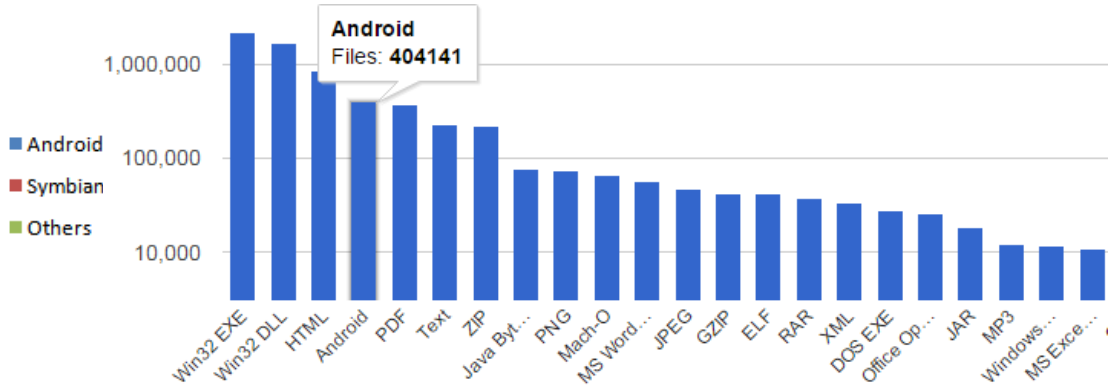


Figure: Submissions @ VirusTotal: June 18-24, 2016

App Distribution

- Apps available via **Google Play. 2 million unique apps at market and increasing [4]**.
  - Dozens of unmonitored third party Stores - Can we trust them?

Android Malware Statistics

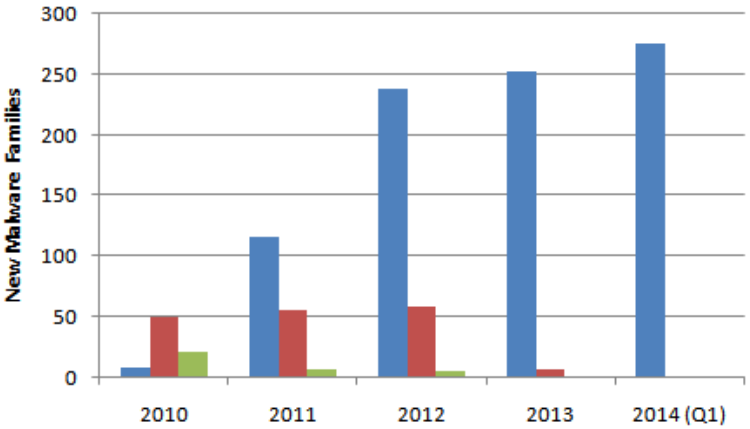


Figure: Source: F-Secure

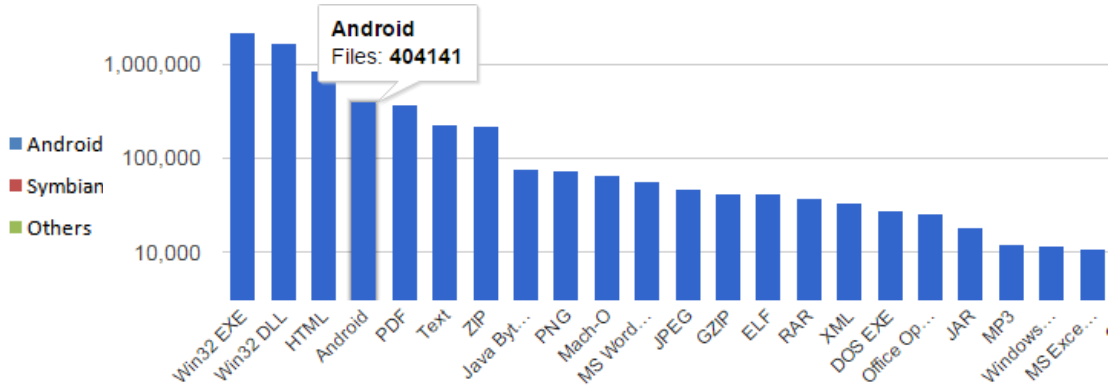


Figure: Submissions @ VirusTotal: June 18-24, 2016

App Distribution

- Apps available via **Google Play. 2 million unique apps at market and increasing [4]**.
  - Dozens of unmonitored third party Stores - Can we trust them?
- **Signatures still preferred by commercial anti-malware.**
  - Constrained resource availability.
  - Need a scalable and automatic approach to tackle increasing threats
  - **Large Scale, automated dynamic analysis is necessary**.

Outline

MimeoDroid

Key Features

- An automated for offline dynamic analysis via machine learning
  - Includes multiple detection techniques for dynamic analysis
  - Mimics the real Android smartphone modifying the **Goldfish** emulator

MimeoDroid

Key Features

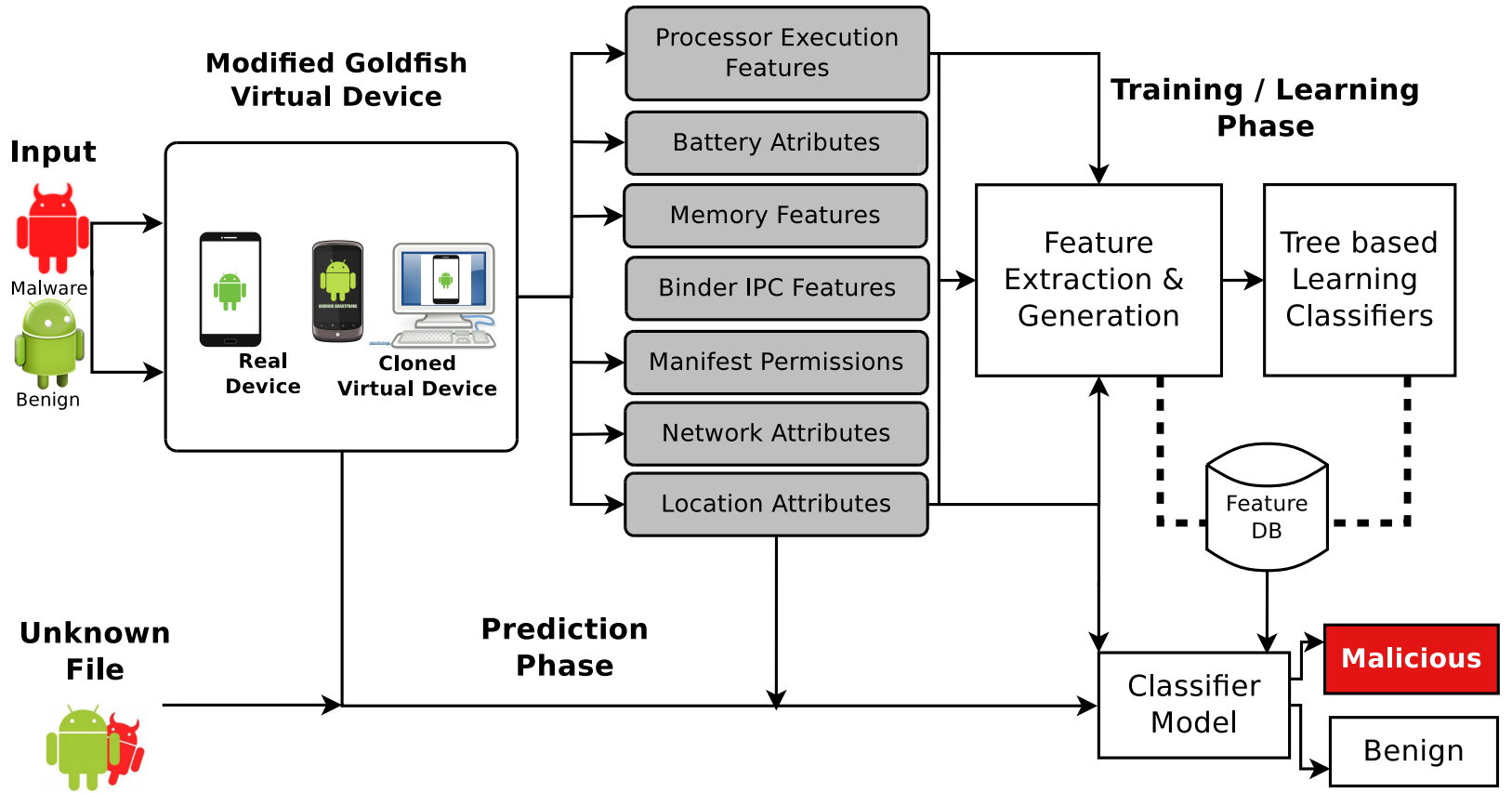
- An automated for offline dynamic analysis via machine learning
  - Includes multiple detection techniques for dynamic analysis
  - Mimics the real Android smartphone modifying the **Goldfish** emulator

Challenges

- Analysis techniques must be lightweight
- Automation requires good exploration techniques
- Manage multiple clones for analysis
- Force environment-aware malware reveal hidden behavior.
- Detect *anti-analysis* malware.

MimeoDroid: Architecture

MimeoDroid



Attributes for classification

Features Selected

- Processor details [cpuUser; cpuIdle; cpuOther]
- Binder IPC [BinderReply; BinderAcquire; Release; ActiveNodes]
- Location [Location changes; No. of nearby Devices]
- Memory [Active; InActive; Mapped; FreePages; DirtyPages;]
- Network [TotalTXPacket; TotalRXPacket; TotalBytes;]
- Manifest details [Androidmanifest.xml requested Permissions]
- Battery attributes [Voltage; Temperature; BatteryLevel;]

Mimic Android device

- Clone the virtualized environment like a real Android device
- Set Phone identifiers and properties
- Data from sensors like GPS
- Add contacts, SMS, files to the **Goldfish**
- Clone static properties to Real Android device

Static Features that Clone Goldfish

How Goldfish mimics Real Android device

Property Value	Default AVD	Real Android Device
IMEI	000000000000000	911315462535214
IMSI	310260000000000	925117254763458
Phone Number	15555525554	91777863242
Serial Number	98101430121181100000	54215E52C54525851254
Network	Android	Tmobile
ro.build.id	ICS MR0	IMM76I
ro.build.display.id	sdk-eng 4.0.2 ICS MR0229537 testkeys	TBW592226 8572 V000225
ro.build.version.incremental	229537	TBW592226 8572 V000225
ro.build.type	eng	user
ro.build.user	android-build	ccadmin
ro.build.host	vpbs2.mtv.corp.google.com	BUILD14
ro.product.model	sdk	msm7627a
ro.product.brand	generic	qcom
ro.product.name	sdk	msm7627a
ro.product.device	generic	msm7627a
ro.product.board		7x27
ro.board.platform		msm7627a
ro.build.product	generic	msm7627a
ro.build.description	sdk-eng 4.0.2 ICS MR0 229537 testkeys	msm7627a-user 4.0.4 IMM76ITBW592226 8572 V000225 testkeys
ro.build.fingerprint	generic/sdk/generic:4.0.2/ICS MRO/229537:eng/test-keys	qcom/msm7627a/msm7627a:4.0.4/IMM76I/ TBW592226 8572 V000225: user/test-keys
net.bt.name	Android	Airtel

Malware Evades Goldfish

Evading the Goldfish

```
public static String a(Context arg4) {
        int v0_1;
        String v3 = "000000000000000";   Default Emulator IMEI
        if(b.c == null) {
            if(arg4.checkCallingOrSelfPermission("android.permission.READ_PHONE_STATE") != -1) {
                Object v0 = arg4.getSystemService("phone");
                if(v0 != null) {
                    b.c = ((TelephonyManager)v0).getDeviceId();   Get Emulator IMEI
                    if(b.c == null) {
                        b.c = Settings$Secure.getString(arg4.getContentResolver(), "android_id");
                    }

                    if(b.c == null) {
                        b.c = v3;
                    }

                    if(v3.equals(b.c)) {   Check for default IMEI
                        v0_1 = 1;
                    }
                    else {
                        v0_1 = 0;
                    }
                    .
                    .
                    .

public static boolean b(Context arg4) {
        boolean v0_1;
        int v0;
        int v1 = -1;
        if(b.d == v1) {
            b.a(arg4);
        }

        if(b.d == v1) {
            if("sdk".equalsIgnoreCase(Build.MODEL)) {   Check for default Build model
                v0 = 1;
            }
            else {
                v0 = 0;
            }
```

MimeoDroid coerces the Malware

Goldfish Evaded

```
Sent data via URL:
http://d.wiyun.com/adv/d?t=1369606972363&a=38&r=0b452cf934676302&s=1_0_6&h=480
&w=320&o=Android+Emulator&v=2.3.4&b=generic&m=google_sdk&u=000000000000000
&n=3&f=0&l=en&c=31260&mm=
```

Emulator IMEI sent to remote server

```
Response:
{"none":"true"}
```

Response due to emulation detection

MimeoDroid coerces the Malware

Goldfish Evaded

```
Sent data via URL:
http://d.wiyun.com/adv/d?t=1369606972363&a=38&r=0b452cf934676302&s=1.0.6&h=480
&w=320&o=Android+Emulator&v=2.3.4&b=generic&m=google_sdk&u=000000000000000
&n=3&f=0&l=en&c=31260&mm=
```

Emulator IMEI sent to remote server

```
Response:
{"none":"true"}
```

Response due to emulation detection

Mimeo Coerces to reveal behavior

```
Sent data via URL:
http://d.wiyun.com/adv/d?t=1390382404479&a=38&r=0b452cf934676302&s=1.0.6&h=800
&w=480&o=Android&v=4.2.2&b=qcom&m=A9%2B&u=911315462535214&n=3&f=0&l=en
&c=92617&mm=*%2F*
```

New IMEI sent to remote server

```
Response:
{"p":"1121","q":"802","m":"application/x-app-store","a":"2","z":"http://d.wiyun.com/adv/r",
"ra":"bb765046-8347-11e3-b7b1-d4bed9ad6f66","i":"http://static.wiyun.com/material/
4e6f48a278c4ad2517c32dbe0b3894a1.jpg","c":"http://downloads.wiyun.com/wiad/
fruitbubblewiad1220.apk","t":"水果泡泡大作战：阿狸抗击魔法婆婆对水果王国的阴谋！","tc":"#CCCCCC",
"bc":"#000000"}
```

Response of remote server(emulation undetected)

Classification Results

Malware Classfication

Table: Classification Results

10 fold Cross-Validation					Testing			
ML Classifier	**Model Size**	**TP %**	**FP %**	**FN %**	**TP %**	**FP %**	**FN %**	**Time taken**
RF	882K	98.1	4.7	1.9	91.2	8.2	9.0	0.72 sec
J48	78K	97.4	5.9	2.6	89.8	9.3	10.2	0.41 sec
MLP	2.9M	97.9	4.97	2.1	90.4	7.3	9.6	0.87 sec
NB	18K	83.2	12.3	16.8	78.3	13.4	21.7	0.31 sec

Classification Results

Class-wise App Analysis

Table: Individual APK performance analysis

Performance of RF on individual APK files malware families							
Benign	**Detection %**	**Malware family**	**Detection %**	**Malware family**	**Detection %**	**Malware family**	**Detection %**
AndroidPlayer	100	AdSmS	100	FakeDoc	92	NandroBox	99.23
Android Reader	100	Agent	100	FakeFacebook	100	Nyleaker	100
Angry Birds	98.3	Airpush	100	FakeFlashPlayer	100	Penetho	100
BatteryMaster	94.8	Antammi	97	FakeInst	100	Pincer	100
ChatOn	100	Antares	96	FakeMart	100	Scavir	100
Cleanmaster	91	ArSpam	100	FakeRegSms	100	Seaweth	100
Facebook	100	BadNews	100	FakeTimer	100	Tetus	94.2
Dropbox	100	FakeAngry	98	FakeUpdates	100	Updtkiller	100
QR Scanner	100	FakeAV	100	Fatakr	98.6	Uracto	100
Skype	100	FakeBank	100	Fidall	96.7	Uten	92
TripAdvisor	100	FakeDaum	100	FinSpy	100	WalkinWat	100
WhatsApp	98.72	FakeDefender	100	Fjcon	100	Zeahache	100

Comparison with state-of-the-art

Comparison with recent work

Property	AASandbox	Andromaly	Apps Playground	Droidbox	Andrubis	Proposed Approach
Transparent & Scalable	✔	✔				✔
Resource Consumption Analysis		✔				✔
API Hooking			✔	✔	✔	✔
Logcat Analysis						✔
System-call Analysis	✔					✔
Risk Prediction using Machine Learning	✔	✔			✔	✔
Anti *Anti-Analysis* Approach			✔	✔	✔	✔
Identifying Data Leakage			✔	✔	✔	✔
Identifying SMS/Call Misuse			✔			✔
Network Traffic Analysis				✔	✔	✔
File Operations Monitoring				✔	✔	✔
Mode of Operation	Offline	Offline	Offline	Offline	Web-based	Offline

Outline

Conclusions

Concluding Remarks

- Android malware problem is something we cannot ignore. And, given the future of IoTs, more to expect from malware community.
- Machine learning classfiers evaluate the dynamic analysis approach using large scale dynamic app validation.

Conclusions

Concluding Remarks

- Android malware problem is something we cannot ignore. And, given the future of IoTs, more to expect from malware community.
- Machine learning classfiers evaluate the dynamic analysis approach using large scale dynamic app validation.

Limitations

- New methods become available to detect Virtual device
- Unavailability of Sensors

Thank You

Questions!

Thank You

References I

Parvez Faruki, Shweta Bhandari, Vijay Laxmi, Manoj Gaur, and Mauro Conti.
Recent Advances in Computational Intelligence in Defense and Security, chapter DroidAnalyst: Synergic App Framework for Static and Dynamic App Analysis, pages 519–552.
Springer International Publishing, Cham, 2016.

Siegfried Rasthofer, Steven Arzt, Marc Miltenberger, and Eric Bodden.
Harvesting runtime data in android applications for identifying malware and enhancing code analysis.
Technical Report TUD-CS-2015-0031, EC SPRIDE, February 2015.

References II


Vaibhav Rastogi, Yan Chen, and Xuxian Jiang.
Droidchameleon: Evaluating Android anti-malware against Transformation attacks.
In *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security*, pages 329–334. ACM, 2013.


Statista.
App stores - statistics and facts — statista.
http://www.statista.com/topics/1729/app-stores/, Online; accessed June 20, 2016 2014.

References III

Guillermo Suarez-Tangil, Mauro Conti, Juan E. Tapiador, and Pedro Peris-Lopez.
Detecting targeted smartphone malware with behavior-triggering stochastic models.
In *In Proceedings of the European Symposium on Research in Computer Security, to appear*, ESORICS 2014. ESORICS, 2014.

Timothy Vidas and Nicolas Christin.
Evading android runtime analysis via sandbox detection.
In *Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security*, ASIA CCS '14, pages 447–458, New York, NY, USA, 2014. ACM.

References IV


Lukas Weichselbaum, Matthias Neugschwandtner, Martina Lindorfer, Yanick Fratantonio, Victor van der Veen, and Christian Platzer.
Andrubis: Android Malware Under The Magnifying Glass.
Technical Report TR-ISECLAB-0414-001, Vienna University of Technology, 2014.


Zhou Yajin and Jiang Xuxian.
Dissecting Android Malware: Characterization and Evolution.
In *Proceedings of the 33rd IEEE Symposium on Security and Privacy*, Oakland 2012. IEEE, 2012.


Min Zheng, Patrick P. C. Lee, and John C. S. Lui.
ADAM: An Automatic and Extensible Platform to Stress Test Android Anti-virus Systems.
In *DIMVA*, pages 82–101, 2012.