Apposcopy: Automated Detection of Android Malware (Invited Talk)*

Yu Feng, Isil Dillig
University of Texas at Austin, USA
{yufeng, isil}@cs.utexas.edu

Saswat Anand, Alex Aiken
Stanford University, USA
{saswat, aiken}@cs.stanford.edu

ABSTRACT

We present Apposcopy, a new semantics-based approach for detecting Android malware that steal private information. Apposcopy incorporates (i) a high-level language for specifying malware signatures and (ii) a static analysis for deciding if a given application matches a given signature. We have evaluated Apposcopy on a corpus of real-world Android applications and show that it can effectively pinpoint malicious applications that belong to certain malware families.

Categories and Subject Descriptors

D.4.6 [**Software Engineering**]: Security and Protection

General Terms

Security, Verification

Keywords

Android, Inter-component Call Graph, Taint Analysis

1. INTRODUCTION

As the most popular mobile operating system, the Android platform is a growing target for mobile malware. Today, many of the malicious applications that afflict Android users exploit the private information stored in a user's smartphone. According to a recent report [3], nearly half of Android malware are multi-functional Trojans that steal personal data stored on the user's phone.

In response to the rapid dissemination of Android malware, there is a real need for tools that can automatically detect malicious applications that steal private user information. Two prevalent approaches for detecting such Android malware are *taint analyzers* and *signature-based detectors*:

Taint analyses, such as [5], are capable of exposing applications that leak private user information. Unfortunately, since many benign apps also need access to sensitive data to perform their advertised functionality, taint analyses cannot automatically distinguish benign apps from malware, and a security auditor must invest significant effort to determine if a given information flow constitutes malicious behavior.

Signature-based malware detectors, including commercial virus scanners, classify a program as malware if it contains a sequence of instructions that is matched by a regular expression. As shown in a recent study, malware detectors that are based on syntactic low-level signatures can be easily circumvented using simple program obfuscations. Hence, these malware signatures must be frequently updated as new variants of the same malware family emerge.

In this paper [1], we present Apposcopy, a new semantics-based approach for detecting Android malware that steal private user information. Drawing insights from the respective advantages of pattern-based malware detectors and taint analyzers, Apposcopy incorporates (i) a high-level specification language for describing semantic characteristics of Android malware families, and (ii) a powerful static analysis for deciding if a given application matches the signature of a malware family. The semantic, high-level nature of the signature specification language allows analysts to specify key characteristics of malware families without relying on the occurrence of specific instruction or byte sequences, making Apposcopy more resistant to low-level code transformations.

The specification language provided by Apposcopy allows specifying two types of semantic properties —control-flow and data-flow—of Android applications. An example of a control-flow property is that the malware contains a broadcast receiver which launches a service upon the completion of some system event. An example of a data flow property is that the malware reads some private data of the device and sends it over the Internet.

To match the signatures specified in this language, Apposcopy's static analysis relies on two key ingredients. First, we construct a new high-level representation of Android applications called the *inter-component callgraph (ICCG)*, which is used to decide whether an Android application matches the control flow properties specified in the signature. Second, Apposcopy incorporates a static taint analysis which is used for deciding whether a given application matches a specified data-flow property.

We have evaluated Apposcopy on a corpus of real-world Android applications and show that it can effectively and reliably pinpoint malicious applications.

2. OUR APPROACH BY EXAMPLE

We illustrate Apposcopy's basic approach using a simplified version of the GoldDream malware family.

*This work was sponsored by the Air Force Research Laboratory, under agreement number FA8750-12-2-0020

DeMobile'14, November 17, 2014, Hong Kong, China
ACM 978-1-4503-3225-5/14/11
http://dx.doi.org/10.1145/2661694.2661697

[1] This is the abbreviated version of our paper in FSE2014. [6]

```
1. GDEvent(SMS_RECEIVED).
2. GDEvent(NEW_OUTGOING_CALL).
3. GoldDream :-  receiver(r),
4.               icc(SYSTEM, r, e, _), GDEvent(e),
5.               service(s), icc*(r, s),
6.               flow(s, DeviceId, s, Internet),
7.               flow(s, SubscriberId, s, Internet).
```

Figure 1: GoldDream signature (simplified)

2.1 GoldDream Signature in Apposcopy

To detect a sample of GoldDream malware, an analyst first writes a signature of this malware family in our Datalog-based language. In this case, the behavior of GoldDream is captured by the specification in Figure 1. Here, lines 1-2 introduce a new user-defined predicate `GDEvent(x)` which describes the events that the `GoldDream` malware listens for. In particular, `GDEvent(x)` evaluates to true when `x` is either `SMS_RECEIVED` or `NEW_OUTGOING_CALL` but to false otherwise.

Using this predicate, lines 3-7 describe the signature of the GoldDream malware family. The signature uses three kinds of predicates provided by Apposcopy:

Component type predicates, such as `receiver(r)` and `service(s)`, specify that `r` and `s` are `BroadcastReceiver` and `Service` components in the Android framework.

Control-flow predicates, such as `icc`, which describes inter-component communication. `icc(SYSTEM, r, e, _)` expresses that the Android system invokes component `r` when system event `e` happens, and `icc*(r, s)` means that component `r` transitively invokes component `s`.

Data-flow predicates, such as `flow(x, so, y, si)`, express that the application contains a flow from source `so` in component `x` to a sink `si` in component `y`. Hence, lines 6-7 state that component `s` sends the device and subscriber id of the phone over the Internet.

Therefore, according to the signature in Figure 1, an application A belongs to the GoldDream malware family if (i) A contains a broadcast receiver that listens for system events `SMS_RECEIVED` or `NEW_OUTGOING_CALL` (lines 3, 4), and (ii) this broadcast receiver starts a service which then leaks the device id and subscriber id over the Internet (lines 5-7).

2.2 GoldDream Malware Detection

Given an Android application A and malware signature S, Apposcopy performs static analysis to decide if app A matches signature S. Apposcopy's static analysis has two important ingredients: (i) *construction of the ICCG*, which allows determining the truth values of control-flow predicates used in the signature, and (ii) *static taint analysis*, which is used to decide the truth values of data-flow predicates. In particular, the ICCG is a graph where nodes are Android components and edges denote inter-component call relations. On the other hand, Apposcopy's taint analysis identifies which designated sources(i.e., sensitive data) can flow to which designated sinks(e.g., Internet, SMS message, etc.). Since Apposcopy's static analyses are both sound and sufficiently precise, it can detect whether an application matches the signature from Figure 1 with very few false positives.

3. EVALUATION

To evaluate the effectiveness and accuracy of Apposcopy, we performed three sets of experiments, including evaluation on (i) known malware from the Android Malware Genome project [1], (ii) Google Play apps, (iii) obfuscated malware.

Table 1: Evaluation of Apposcopy on malware from the Android Malware Genome project.

Malware Family	#Samples	FN	FP	Accuracy
DroidKungFu	444	15	0	96.6%
AnserverBot	184	2	0	98.9%
BaseBridge	121	75	0	38.0%
Geinimi	68	2	2	97.1%
DroidDreamLight	46	0	0	100.0%
GoldDream	46	1	0	97.8%
Pjapps	43	7	0	83.7%
ADRD	22	0	0	100.0%
jSMSHider	16	0	0	100.0%
DroidDream	14	1	0	92.9%
Bgserv	9	0	0	100.0%
BeanBot	8	0	0	100.0%
GingerMaster	4	0	0	100.0%
CoinPirate	1	0	0	100.0%
DroidCoupon	1	0	0	100.0%
Total	1027	103	2	90.0%

(i) The data from Table 1 shows Apposcopy is able to detect Android malware with high accuracy of overall 90.0%. More specifically, the false negative rate is around 10.0% and the false positive ratio is less than 0.2%.

(ii) To verify that our high-level signatures also differentiate benign applications from real malware, we evaluated Apposcopy on 11,215 apps from Google Play and Apposcopy reported 16 of them as malware. We confirmed that those 16 apps are malware through VirusTotal [4].

(iii) In the third experiment, we obfuscated existing malware using the ProGuard [2] tool and compared the detection rate of Apposcopy with other commercial anti-virus tools on obfuscated versions of known malware. The result shows that Apposcopy's detection rate is 100.0% while the detection rates of other tools range from 14.3% to 57.1%.

4. CONCLUSION

We presented Apposcopy, a static analysis approach for detecting malware in Android apps. Malware that belong to one family share a common set of characteristic behaviors, which an auditor can encode through Apposcopy's Datalog-based malware specification language. Apposcopy performs deep static analysis to extract data-flow and control-flow properties of Android apps and uses these results to identify whether a given app belongs to a known malware family. Our experiments indicate that Apposcopy can detect malware with high accuracy and that its signatures are resilient to many kinds of program transformations.

5. REFERENCES

[1] Android malware genome project. `http://www.malgenomeproject.org/`.

[2] ProGuard. `http://proguard.sourceforge.net/`.

[3] Q2 IT evolution threat report. `http://tinyurl.com/lcg3ojb`.

[4] VirusTotal. `https://www.virustotal.com/en/`.

[5] W. Enck, P. Gilbert, B. gon Chun, L. P. Cox, J. Jung, P. McDaniel, and A. Sheth. TaintDroid: An information-flow tracking system for realtime privacy monitoring on smartphones. In *OSDI*, pages 393–407, 2010.

[6] Y. Feng, S. Anand, I. Dillig, and A. Aiken. Apposcopy: Semantics-based detection of android malware through static analysis. In *SIGSOFT FSE*, 2014.