Extractocol: Automatic Extraction of Application-level Protocol Behaviors for Android Applications

Hyunwoo Choi[†*], Jeongmin Kim[†*], Hyunwook Hong[†], Yongdae Kim[†]
Jonghyup Lee[‡], and Dongsu Han[†]

KAIST[†], Gachon University[‡]

CCS Concepts

•**Networks** → **Application layer protocols;** •**Security and privacy** → *Software reverse engineering;*

Keywords

Android; protocol behaviors; static analysis

1. INTRODUCTION

Android app is an important class of today's Internet applications that generate roughly 40-50% of mobile Web and app traffic. More than 1.4 million Android apps are offered through Google's open market, and tens of thousands of new apps are added every month. However, very little information is known about their application protocol behaviors because they predominantly use proprietary protocols on top of HTTP [3, 6]. The problem is further exacerbated by the popular use of common data representation, such as JSON and XML, due to the popularity of REST-ful web services [5]. As a result, application protocols appear similar to each other, and analyzing them requires an in-depth characterization of each individual application. Despite this, the state of the art remains that even the problem of finger-printing the traffic they generate is extremely challenging [3, 6], let alone a full protocol analysis.

Our vision is to be able to automatically reconstruct the client's protocol state machine and the message format from the program binary. Being able to analyze network protocols provides not only intrinsic values, but also enables many new applications, such as protocol testing [4], flow classification, protocol normalization, and automatic proxying. To this end, we design a system, called Extractocol, that takes only the application binary as input and uses static program analysis. We specifically focus on automatically extracting the request-response formats and signatures for apps that use the popular HTTP protocol as transport. It extracts parts of application code (i.e., program slices) that either generate HTTP(S) requests or parse response messages and applies a careful semantic analysis to extract message formats and signatures from the target program. In particular, it output regular expression (regex) signatures for each request (including URI, query string, request method, header, and body) and the corresponding response. It also partially reconstructs client state by tracking dependancies between messages.

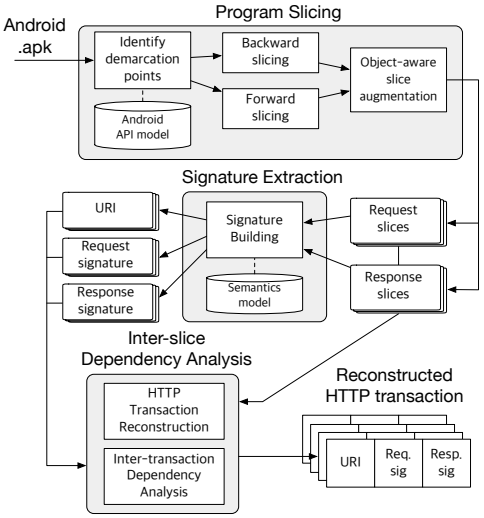

Figure 1: Design overview of Extractocol

In summary, this paper makes two key contributions:

- **Novel approach to Android application protocol analysis:** We present the first comprehensive protocol analysis framework for Android applications that is capable of extracting protocol behaviors, formats, and signatures, given an APK.
- **Working system prototype and its evaluation:** Our evaluation on 12 open-source demonstrates that Extractocol provides a rich and comprehensive characterization.

2. DESIGN

Figure 1 illustrates three main components of Extractocol: program slicing, signature extraction, and inter-slice dependency analysis.

Program slicing: We extend FlowDroid [2], a static taint analysis tool, to reconstruct the data dependencies arising from network I/O. We use this information to create request and response slices. A typical program contains many instructions other than protocol processing. Thus, Extractocol pre-processes the APK to extract code slices only related to protocol processing. The goal of this step is to output program slices that generate HTTP requests and process responses.

*These authors contributed equally to this work.

SIGCOMM '15 August 17-21, 2015, London, United Kingdom

ACM ISBN 978-1-4503-3542-3/15/08.

DOI: http://dx.doi.org/10.1145/2785956.2790003

#	Request URI	Request Body
1	GET (http://www.reddit.com/api/info.json?)	-
2	GET (http://www.radioreddit.com/)(status.json)	-
3	POST (https://ssl.reddit.com/api/login)	(user=).*(&passwd=) (&api_type=json)
4	POST (http://www.reddit.com/api/)(unsave|save)	(id=).*(&uh=)(.*)
5	POST (http://www.reddit.com/api/vote)	(id=).*(&dir=).*(&uh=)(.*)
6	GET (.*)	(.*)

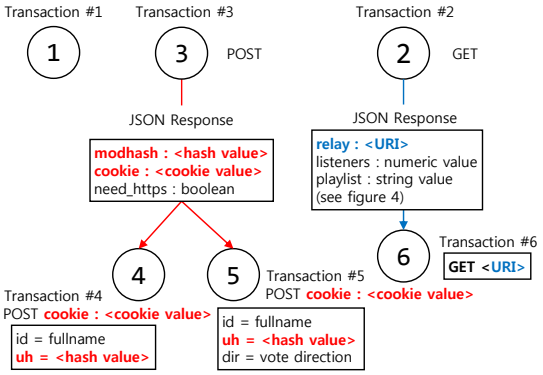

Figure 2: Reconstructed HTTP transactions and their dependency graph for Radio reddit

Signature extraction: The second phase takes the request/response slices as input and generates formats and signatures for each. To extract signatures, Extractocol performs analysis using semantic models for a set of commonly used Android and Java APIs for protocol processing, such as `java.lang.String` and `java.net`. It then outputs signatures for request URIs and request/response bodies as well as the request method and additional HTTP headers used.

Inter-slice dependency analysis: Finally, Extractocol reconstructs a complete transaction by pairing a request URI with its corresponding response. It also infers the relationship between HTTP transactions. In particular, it infers which part of request URI or body is potentially derived from prior responses. The key idea is to identify inter-slice relationships between the request and response slices. For this, Extractocol performs novel inter-slice data flow analysis and addresses a number of issues in handling subtle, but complex inter-slice dependencies that arise due to code reuse.

3. EVALUATION

We use 12 apps with network permissions from an open source app repository (F-Droid) [1]. We obtain the ground truth by carefully inspecting the **source code**. We also collect traffic traces of all HTTP(S) transactions using **manual UI-fuzzing**, which often requires manual interventions, such as signing up and logging in for services. Extractocol took 14.2 minutes to analyze all 12 apps with the optional features. Table 1 shows the ground truth number of HTTP messages by their request method and message type (e.g., request or response body) and the fraction of signatures identified by Extractocol. To demonstrate Extractocol's effectiveness in characterizing app's protocol behavior, we perform a case study on radio reddit. Radio reddit is an online music streaming client that allows users to choose radio stations and vote on or save songs using their reddit accounts. Figure 2 shows a complete analysis result with six transactions; five of them use HTTP, while login request uses HTTPS. It also shows a dependency graph identified by Extractocol. Extractocol identifies that login request (#3) includes three fields, and its response is a JSON object including "modhash" and "cookie" as keys. They use the "modhash" value in the "uh" field, and add the "cookie" value to their request headers.

4. CONCLUSION AND FUTURE WORK

This work presents Extractocol, a framework for analyzing HTTP(S)-based application protocol behaviors for Android applications. Unlike previous approaches, Extractocol only uses the application binary as input to reconstruct application-specific HTTP-based interactions using static program analysis. It combines static taint analysis and semantic analysis to provide a comprehensive characterization of application protocol behaviors. We believe Extractocol and its approach can serve as a basis for generic protocol analysis (other than HTTP) for Android applications. Finally, we plan to evaluate it on a large set of commercial applications.

App	Request			Response	#Pair
	GET	POST	Body	Body	
ADP	2 (100%)	1 (100%)	1 (100%)	1 (100%)	1 (100%)
AXV	2 (100%)	-	-	2 (100%)	2 (100%)
BLP	1 (100%)	-	-	1 (100%)	1 (100%)
DIW	1 (100%)	-	-	1 (100%)	1 (100%)
LNT	2 (100%)	-	-	1 (100%)	1 (100%)
QBT	3 (100%)	13 (100%)	13 (100%)	3 (100%)	3 (100%)
RRD	3 (100%)	3 (100%)	3 (100%)	4 (100%)	4 (100%)
RDN	3 (100%)	3 (100%)	-	6 (100%)	6 (100%)
TWT	-	11 (100%)	11 (100%)	8 (100%)	8 (100%)
TZM	2 (100%)	-	-	1 (100%)	1 (100%)
WLB	1 (100%)	-	-	1 (100%)	1 (100%)
WTN	2 (100%)	-	-	2 (100%)	2 (100%)

Table 1: Coverage: Ground truth # of request/response and % of identified signatures

5. ACKNOWLEDGMENTS

This research was supported in part by an Institute for Information communications Technology Promotion (IITP) grant funded by the Korean government (MSIP) (No. B0126-15-1078), and by Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (No. NRF-2014M3C4A7030648).

6. REFERENCES

[1] Free & open source app repository, https://f-droid.org.
[2] S. Arzt, S. Rasthofer, C. Fritz, E. Bodden, A. Bartel, J. Klein, Y. Le Traon, D. Octeau, and P. McDaniel. Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps. In *Proc. ACM SIGPLAN PLDI*, June 2014.
[3] S. Dai, A. Tongaonkar, X. Wang, A. Nucci, and D. Song. Networkprofiler: Towards automatic fingerprinting of android apps. In *Proc. IEEE INFOCOM*, 2013.
[4] N. K. R. M. T. M. R. G. Luis Pedrosa, Ari Fogel. Spa: A general framework for systematic protocol analysis. In *Proc. USENIX NSDI*, 2015.
[5] L. Richardson and S. Ruby. *Restful Web Services*. O'Reilly, first edition, 2007.
[6] A. Tongaonkar, R. Keralapura, and A. Nucci. Challenges in network application identification. In *Proc. USENIX LEET*, 2012.