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

Hyunwoo Choi1†, 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 fingerprinting 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

[Diagram of Extractocol system]

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.
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.

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-2014M3C1A7030648).

6. REFERENCES