# Consolidated Review of

# *RILAnalyzer: A Comprehensive 3G Monitor on Your Phone*

## 1. Strengths:

Paper is very well written and easy to follow.

The tool provides RNC state information for a device without relying on captures from cellular infrastructure or external energy measurements. The tool does not require root access or modifications to the phone's firmware. This paper introduces an open-source and much-needed tool to analyze cellular networks for many researchers who do not have access to the internal components of cellular networks. RILAnalyzer pretty clearly advances the state of the art for collecting control- and data-plane information for analyzing wireless application interactions. It appears to be a very powerful and useful tool, and will be released to the community. Tool is open-source and available for user and improvement.

The authors ascertain the RNC state machine for four different cellular networks and correlate RNC state machine transitions with specific applications without complex analyses or inference algorithms

The authors discovered interesting inefficiencies.

## 2. Weaknesses

The tool only works for mobile chipsets whose special commands for extracting state information are known (currently only one chipset). It would be very interesting to see the results collected by the tool from a diverse set of popular mobile handsets including iPhones.

The small number of users in the study makes me question the accuracy of the frequency of RNC state machine transitions induced by specific applications.

The paper contains many unexplained details, especially for non-experts in this area. The case studies especially contain references to details that not explained well or at all, making the paper pretty hard to digest (especially RNC state machine discussion).

Insufficient detail describing validation of the tool (ironic, considering so much detail in other parts of the paper).

Lack of strong motivation and sell; what recommendations does this tool actually provide for telcos?

The tool doesn't seem to offer much novelty in terms of engineering/design; it extends existing work a bit and probes with the right (hidden) parameters.

## 3. Comments

First off, the paper is very clearly written and easy to follow. A nice story is presented, revolving around the tool, and there are some useful insights obtained from the smart use of graphics and figures. The tool seems solid and easy to run by many individuals.

I was immediately intrigued why the backdoor access that is available to RNC state machines on some chipsets, and I immediately went to try it on my own Samsung Galaxy S II. Although your tool only works for a limited set of devices, I think it is still important to let the research community know about this valuable, albeit hidden, source of data.

It was not clear how you were improving on existing approaches until after I read about your tools design in the beginning of Section 3. Section 2 talks about limitations of other approaches, but makes no forward reference as to how you are proposing to overcome these limitations or what specific problem you are targeting.

The front half of the paper, particularly the introduction, does not motivate or sell the tool very well. Specifically, the reader does not get a clear understanding of exactly how this tool is going to help solve, or provide recommendation for, the well-known problems of cellular networks. It would be more satisfying if there were a handful of major recommendations to telcos, derived from the problems found by the tool, right up front. The introduction does not clearly present the problem being addressed or provide useful forward references into how your tool addresses the problem nor does it talk about the measurements you gathered. The first two paragraphs of the introduction do not add any value to the paper The third paragraph of the introduction talks about correlating information between different layers, but you really only talk about two pieces of information---RNC state and packets sent by applications.

There's a lot more information that other applications have gathered (signal strength, location, CPU usage, energy usage, etc.), but you don't gather any of this. If you only think network information is important, then you should be more clear about why you focus on this.

In the first paragraph of Section 3 you say that you don't rely on collecting data from the OS, but you do not clearly explain why such an approach is limited or problematic. I would have liked to see support for more chipsets, or at least something that would convince me that other chipsets could be amenable to being supported in your tool. You say that Qualcomm provides a licensed monitoring tool; is it possible to reverse engineer this? Why or why not? Also, what other common chipsets exist that your tool should target?

When verifying the accuracy of data packets reported by RILAnalyzer, this paper uses a small set of experiments with lightweight traffic load. It is desirable to perform some stress tests here to see whether RILAnalysis is capable of accurately capturing packets during heavy traffic load. The current version of RILAnalyzer supports a single chipset.

What is the road map of supporting other chipsets and mobile handsets? I believe many researchers would like to deploy such tools on both Android phones and iPhones across different cellular networks in North America and Asia.

During the idle experiments, RILAnalyzer consumes an average of 22MB of physical memory, while it consumes only 42MB of memory during the stress test. What is the reason of such a high memory usage during the idle condition? This paper uses a small-scale user study to evaluate the developed tool. Given the popularity of Android-based mobile handsets, this paper could expand the scale of the deployment.

Table 1 (and the entire paper) uses the term U-plane, i.e., user plane, while Table 2 uses the term "data plane". For consistency, Table 2 should also use the term "user plane".

In section 2, end of paragraph 4, a comment is made about the inaccuracy of using reverse DNS lookups for identifying applications, and that you "prove" in section 4 this inaccuracy. Are you referring to the fact that some apps rely on other infrastructures (e.g., Google) for some services (like push notifications), thus rendering reverse DNS approaches unable to identify the app correctly? The connection to the statement in section 2 isn't clearly made in section 4, and that would be helpful for overall coherency. Why no explicit comparison of packet timestamps (Section 3.1, validation)? Inaccuracy is mentioned, but it would have been nice to actually see the differences. While the inaccuracies did not affect the specific case studies described in this paper, it seems like they could have detrimental impact on other kinds of experiments using RILAnalyzer output. For comparing RIL state change info (Section 3.1, validation), does using the cell testbed have an impact on the rate or nature of state changes you'd see versus what you'd see in a live environment? Some additional details of the testbed experiments would have been helpful to understand the validation approach better.

However, there are some questionable measurements presented by the authors, such as using 47% of the CPU under heavy load (as an aside, it may also have been useful to translate that to battery life lost). This may be a tool limitation (the authors specifically mention logging by iptables), which could be optimized by the open source community. The other limiting factor is the specific hardware required; although, as a proof-of-concept, this is fine. There are a few somewhat unspecific recommendations scattered throughout the paper, such as "These observations suggest that to reduce the energy and network overheads of mobile traffic, it is essential to control downlink traffic…" The user study was done with real phones, over real networks, using real apps, leading strong credence to the results. Many of the results are interesting, such as the RNC state transitions and the impact that CDNs and other backend infrastructure can have. The validation in 3.1 has no data/graphs associated with it; however, it was thorough of the authors to perform this type of validation. While the tool seems interesting, I question the overall contribution to the research community. Discovering the correct codes to enter into the dialer seemed to be the most tricky part of the whole system, and in fact that seems to be the only step necessary to obtain the control-plane data. This seems like a fantastic and useful open-source project and engineering effort, but a questionable research contribution. Overall, it seems like a nice tool that can be expanded on, and I hope it continues to provide useful information.

## 4. Summary from PC Discussion

The PC discussion focused on how useful this tool was likely to be, both now and longer term. We discussed which current cell phones could run this tool and concluded it was certainly useful now and decided that was sufficient to accept.

Is root access needed to use your tool? The PC wondered about how this would be used? If root access is needed, this may limit how useful the tool is. The author should clarify this. The PC wanted the authors to provide more details on which phone models this tool applies to, and whether those are popular models.

## 5. Authors' Response

The tool, as described in the text, is intended as a detailed cross-layer debugging/analysis/monitoring tool on the handset, rather than a tool for long-term analysis. Many issues arise in the latter case that need to deal with users' privacy and the need to build an scalable online logging capability. We would like to explore this in the future.

At the moment, the tool only works for the XGold chipsets, as described in the paper. Extending the tool to other handsets requires time to identify the chipset-specific commands and we would like to have support from the open source community to extend the capabilities of this tool. We hope that this paper will motivate open APIs to access this information, nowadays restricted, directly on mobile handsets for research, development and network deployment purposes.

Regarding the content of the paper itself, we have modified the introduction to better describe the content of the paper, as well as a new section that includes some basic knowledge about cellular networks. We also described better the limitations of our tool as it only works for rooted Android handsets with XGold chipsets (only tested for Samsung Galaxy S2 and S3 devices). Furthermore, Section 4 already describes the different memory and CPU overheads, caused by the need to poll all the information from the radio chipset and the OS. In particular, the reason why Rilanalyzer consumes 22 MB of physical memory even at low traffic loads is due to the need to have multiple components monitoring different aspects simultaneously. Each one of them takes some space. Furthermore, the traces are collected, batched and saved in memory for post-processing, being written to the SDCARD once the OS has enough resources to do it. Although a high CPU load is not desirable due to the associated energy costs, this is a limitation imposed by platform/chipset support.

We also explained in S2 and S6 the limitations associated to Reverse DNS techniques on identifying the process that has generated a given packet. The main reason is that many mobile apps use 3$^{rd}$ party online services such as Google's Push Notification, analytics, advertisement and even content delivery networks. Many applications do not have a monolithic backend, making the identification of an app's traffic difficult.

Furthermore, the inaccuracy on the incoming timestamps is a NFLOG limitation (Kernel). This has been already explained in the paper. The same timestamps are also reported by *tcpdump*. If reviewer's point is related to the inaccuracy of the RNC transition and the 1 second sampling rate, we explain also that it is not necessarily a limitation and that it can be only improved with newer APIs.

The settings for RNC promotions and demotions, as well as the RNC state machine, are configurable and we have observed that in different settings in different networks (Figure 4). The testbed allowed us to have a controlled environment to validate what the handset reported.