WiSec 2011 Demo: RFReact—A Real-time Capable and Channel-aware Jamming Platform

Matthias Wilhelm[*] *wilhelm@cs.uni-kl.de*
Ivan Martinovic[*] *martinovic@cs.uni-kl.de*
Jens B. Schmitt[*] *jschmitt@cs.uni-kl.de*
Vincent Lenders[‡] *vincent.lenders@armasuisse.ch*
[*] Disco Labs, TU Kaiserslautern, Germany
[‡] Armasuisse, Thun, Switzerland

We present RFReact, a USRP2-based platform that enables selective and reactive RF jamming. We demonstrate that RFReact is both powerful and versatile with a jamming system that can demodulate the header of an IEEE 802.15.4 packet, decide whether to jam it based on its content, and carry out the decision all while the packet is still on the air.

I. Motivation

Recently, the physical layer of wireless communications received an increasing interest in the networking community in general, and in the wireless security community in particular. Especially the effects of interference (e.g., by co-existing networks or RF jamming) on performance and security, as well as the identification of possible countermeasures, are an active area of research. An experimental evaluation of RF interference is often required here, as channel models do not capture the behavior of wireless propagation accurately. To assist researchers in this important task, we provide RFReact to the community, a reliable and versatile platform for RF experimentation that can generate precise and even channel-aware interference patterns.[1]

As RFReact is implemented on top of the widely used software-defined radio (SDR) platform USRP2, it can analyze received RF signals in software and trigger transmissions of arbitrary waveforms in response. In this aspect, it is similar to GNU Radio-based systems where a host PC is used to analyze and generate signals. However, the delay from a signal's presence on the channel to the reaction by the host-based system is too large for time-sensitive applications like reactive jamming. With RFReact, we moved all critical components closer to the hardware to minimize these delays. Our experiments in recent work [1] show that the system reacts fast enough to detect and jam packets currently on the air, with a delay in the order of microseconds, instead of milliseconds as in the host-based architecture. We argue that such short reaction times are essential to perform repeatable and controllable real-world RF interference experiments.

The design of RFReact focuses on flexibility and extensibility, e.g., new RF analyzers may be added to detect a wider range of signals on the wireless medium. In this demo, we show the potential of our concept: RFReact is able to selectively jam packets based on their content, using a real-time 802.15.4 demodulator. This allows to restrict its jamming activity to ACKs only, to selected beacons or to data frames carrying a specific sender address.

II. Applications

We summarize some of our current work and new application ideas for RFReact.

Jamming strategies. The selectivity offered by RFReact enables the implementation and experimental evaluation of *smart jammers*. Such jammers focus on the destruction of high-value packets to maximize the impact of their attacks; and with real-time demodulation, this content-based jamming decision can be done while packets are still in transmission. Experiments using RFReact help to gauge the real-world performance of such highly selective attacks.

Jamming detection. With the shown practicability of smart jammers, the detection of reactive jamming is crucial to initiate countermeasures and to ensure reliable operation in the face of such threats. RFReact can make a contribution to the evaluation of jamming detection methods in realistic scenarios when considering powerful adversaries.

Jamming mitigation. The performance of anti-jamming techniques implemented in hardware can also be evaluated by experimentation. Our system offers transmission scheduling in the order of nanoseconds, such that the effect of jamming bursts to arbitrary parts of a signal can be quantified repeatably.

Active wireless adversaries. RFReact helps to evaluate injection or message manipulation attacks that require a tight time synchronization with a signal, e.g., to jam a packet and subsequently inject another one (of the attacker's choice) with minimal delay, or even to overshadow parts of transmissions

[1] For more information and access to the source code, please visit `http://disco.cs.uni-kl.de` or contact the main author. We welcome your feedback on the RFReact platform and any information regarding its successful application.

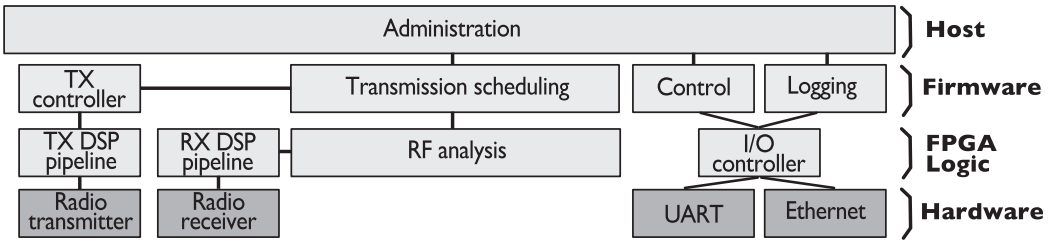

Figure 1: RFReact's design, showing the functional components and their allocation to the USRP2's resources.

in order to manipulate the content of packets without affecting their timing. Along this line, RFReact can help to understand the capabilities of sophisticated attackers, e.g., when considering a wireless Dolev-Yao adversary with full channel control.

Controlled interference. Related to jamming, RFReact can generate arbitrary interference patterns with fine-grained control, without the requirement to change the networked system under test. For example, a scenario with 70 % loss of ACK frames is easily defined and generated at a central point with RFReact, without the need to alter hardware drivers of participating devices in the experiment to implement the experiment's specifications.

The USRP2 is compatible to a large number of daughterboards, making the whole spectrum from DC to 5.8 GHz reachable for experimentation with RFReact.

III. RFReact's Design

We implemented RFReact by extending the default system that comes with the USRP2. The system provides sending and receiving functionalities of RF samples, and communication facilities to the host. It is divided into four tiers: *(i)* the hardware tier with converter chips that sample the RF spectrum, *(ii)* an FPGA, *(iii)* a soft-microcontroller implemented in FPGA logic that is used to control the device via firmware, and *(iv)* a host computer that interconnects via Ethernet to receive and process the samples, as well as to configure the device settings.

RFReact's design uses several of these tiers on the USRP2 for its components (also refer to Fig. 1):

RF analysis: The RF analyzers detect features of interest on the medium. They are implemented in FPGA logic as they require high performance and deterministic timing. Thus, the samples are processed where they are generated, cutting down the latency to a few microseconds. An example of this is our IEEE 802.15.4 demodulator. Several of such analyzer modules may be implemented in parallel to enable signal analysis along several features.

Transmission scheduling. Transmissions can be scheduled in firmware based on global time or on the channel state delivered by the RF analyzers. Its program runs on the microcontroller, combining a short reaction time and easy development, as the firmware code is written in C. In this module the output of the RF analyzers is processed.

TX controller. The shape of TX waveforms can be chosen arbitrarily: they are either precomputed on the host PC, or generated in firmware or in FPGA logic, depending on the desired complexity of the signal and the tolerable delay to initiate sending.

Administration. The control and administration functionality resides on a host PC, which can communicate with the USRP2 either via serial connection or UDP over Ethernet. With this mechanism, the configuration of the scheduling logic can be changed, new waveforms inserted, and statistics gathered during the runtime of the system.

III.A. Extending RFReact

In our design, we emphasized the ability to adapt and extend RFReact for various applications. *RF analyzers* can be added to search for new features of interest on the wireless channel. As input, such an analyzer receives complex samples from the radio front-end to apply its signal processing on. As output, interrupts are used to interact with the firmware for event notification, and a bus interface is used to exchange data between the analyzer and the rest of the system.

The *transmission scheduling* module is part of the microcontroller's interrupt handler, and is able to request information on the detected event via the bus interface. This information is then used to make scheduling decisions, specifying the behavior of RFReact.

Defining *transmit waveforms* by the system is a matter of specifying a sequence of complex samples that represent the waveform. In this way, complex waveforms such as modulated bytes can be transmitted, e.g., to communicate with the wireless network.

References

[1] M. Wilhelm, I. Martinovic, J. B. Schmitt, and V. Lenders. Reactive jamming in wireless networks: How realistic is the threat? In *Proc. of ACM WiSec '11*, pages 47–52. ACM, June 2011.