An Architecture for Secure Software Defined Radio

Chunxiao Li
Department of EE
Princeton University
chunxiao@princeton.edu

Anand Raghunathan
School of ECE
Purdue University
raghunathan@purdue.edu

Niraj K Jha
Department of EE
Princeton University
jha@princeton.edu

***Abstract*—Software defined radio (SDR) is a rapidly evolving technology which implements some functional modules of a radio system in software executing on a programmable processor. SDR provides a flexible mechanism to reconfigure the radio, enabling networked devices to easily adapt to user preferences and the operating environment. However, the very mechanisms that provide the ability to reconfigure the radio through software also give rise to serious security concerns such as unauthorized modification of the software, leading to radio malfunction and interference with other users' communications. Both the SDR device and the network need to be protected from such malicious radio reconfiguration.**

In this paper, we propose a new architecture to protect SDR devices from malicious reconfiguration. The proposed architecture is based on robust separation of the radio operation environment and user application environment through the use of virtualization. A secure radio middleware layer is used to intercept all attempts to reconfigure the radio, and a security policy monitor checks the target configuration against security policies that represent the interests of various parties. Therefore, secure reconfiguration can be ensured in the radio operation environment even if the operating system in the user application environment is compromised. We have prototyped the proposed secure SDR architecture using VMware and the GNU Radio toolkit, and demonstrate that the overheads incurred by the architecture are small and tolerable. Therefore, we believe that the proposed solution could be applied to address SDR security concerns in a wide range of both general-purpose and embedded computing systems.

I. Introduction

Software defined radio (SDR) is a promising technology in radio communications that offers a flexible mechanism to implement radio functionality, such as signal generation, coding, and modulation, in software. This gives rise to the possibility of adapting the radio to users' preferences and the operating environment (*e.g.*, cognitive radios), and of supporting multiple standards without requiring separate hardware for each standard. However, the ability to reconfigure the radio through software also gives rise to new security concerns.

In its report on SDR, FCC states that it is critical "to ensure that software changes cannot be made to a radio that will cause it to operate with parameters outside of those that were approved in order to prevent interference to authorized radio services" [1]. For example, malicious software could modify the RF parameters, like frequency and power, such that the radio functions out of the regional regulations and interferes with other users' normal communication through jamming. Due to the presence of software vulnerabilities in all operating systems (OSs), any reconfiguration mechanism that is embedded within or layered on top of the OS is as insecure as the OS itself. Thus, it is very important to develop technologies that ensure secure and reliable reconfiguration of SDRs.

Past research in SDR security has followed a few general directions. The first direction is to explore threat analysis and security requirements for SDR. For example, the threats and objectives of SDR security are discussed in [2], [3], [4], [5]. A thorough list of SDR security requirements for hardware, platform integrity, downloading, key management, and operational guidance is presented in [6]. A threat analysis of GNU software radio is performed in [7]. The second direction targets an essential issue in SDR security: secure downloading of new radio functionality. A framework for establishing secure download using a tamper-proof hardware module is proposed in [8], [9]. In [10], [11], it is shown how the download of radio configuration files by a protocol called LSSL can securely connect a server or base station and SDR devices. An FPGA-based SDR that uses different layouts and configuration information of FPGA as the key for downloading and execution is discussed in [12]. The third direction focuses on spectrum management and policy enforcement of SDR. A solution consisting of three distinct modules to monitor, validate and implement system configurations to ensure policy compliance is proposed in [13]. A new spectrum management architecture for a flexible SDR based on an automatic calibration and certification unit, built-in GPS receiver, and radio security module is described in [14].

All of the proposals described above rely on a security module to secure the downloading and installation of software or RF configuration parameters. The security module is based on either software or tamper-proof hardware. However, these solutions have their own intrinsic drawbacks. Solutions that are based on hardware cannot adapt the security policy to the changing environment and cannot be easily updated, *i.e.*, they are not easy to reconfigure (contrary to the objectives of SDR). On the other hand, software implementations of the security module are vulnerable to malicious modification. Even though some solutions include an integrity checking mechanism, this mechanism may itself be modified/blocked by a malicious operating environment (such as a compromised OS), just like a virus-infected OS would block or bypass virus-scanning

Acknowledgment: This work was supported by NSF under Grant no. CNS-0720110.

978-3-9810801-5-5/DATE09 © 2009 EDAA

software.

In this paper, a new architecture for secure SDR is proposed, based on two key concepts. The first concept is to separate the application environment and the radio operation environment such that the compromise of one will not affect the other. Second, all the SDR reconfiguration parameters produced by the application environment are checked against security policies before they take effect in the radio environment. Even if the application environment (OS kernel in most situations) is tampered with and becomes malicious, it cannot infect the radio environment and thus the RF characteristics can be ensured to be in compliance with the desired policies.

The rest of this paper is organized as follows. Section II motivates the need for a new architecture for SDR security. Section III introduces the details of the proposed architecture. Section IV presents the implementation of the architecture using VMWare and the GNU Radio toolkit, experimental results evaluating the incurred overheads, and a conceptual security analysis of the proposed architecture. Section V concludes the paper.

II. Motivation

In this section, we examine security issues in dynamic spectrum sharing to illustrate why it is important to secure the reconfiguration of SDR, and discuss limitations of the methods proposed in past research. Then, we discuss the basic ideas behind the proposed architecture and show how these drawbacks can be overcome.

The radio spectrum is a limited resource that needs to be shared among a perpetually increasing range of wireless communication services. It is generally agreed that a more dynamic policy of spectrum sharing is needed. In [15], secondary use of spectrum is discussed for dynamic spectrum sharing. It is defined as "a temporal use (by a secondary user) of existing licensed spectrum currently occupied by the incumbent (primary user)." The primary user may have paid for the spectrum, and will allow secondary use only without interference or within a tolerable quality of service (QoS). The security issue that emerges here is that a greedy secondary user may maliciously modify his SDR and try to occupy disproportionate resources, interfering with the primary user. Even worse, a malicious user may try jamming the entire channel to interfere with normal communications.

Previous work on secure spectrum management [13], [14] has proposed the use of a policy module to define the regional constraints of RF parameters and a monitor module to check whether the SDR is in a legal state, *i.e.*, parameters are within legal constraints. These modules are implemented in either software or hardware. If these modules are implemented in software, the same problem as before emerges: what if the OS that runs these software modules is tampered with, and makes the monitor module pass all legal or illegal parameters? If these modules are implemented in hardware, some attributes are not easy to check: the hardware monitor can analyze the frequency and power, but it is difficult to check properties such as the cryptographic strength (whether the data have been encrypted as required), *etc*. Moreover, implementing a hardwired policy module goes against the basic principle of SDR to provide maximum flexibility.

To solve the problem in SDR based spectrum sharing, the proposed architecture separates the application environment and radio environment, so that the application environment, which is easy to tamper with through malicious programs or vulnerabilities, cannot interfere with the radio environment, which is security-critical in implementing and protecting RF parameters. Secondly, in the tamper-proof radio environment, a security policy monitor is implemented that checks for compliance when reconfiguration is needed for the radio applications. In the above scenario of dynamic spectrum sharing, when the secondary user has received the allocated spectrum, the security policy monitor forms the corresponding policy. Before SDR reconfiguration occurs, it is checked against the new policy. If the parameters do not comply with the security policy, the monitor blocks the reconfiguration from taking effect.

III. Secure SDR architecture

In this section, we first provide an overview of the proposed architecture, and then introduce its components in detail.

A. Overview

The software in an SDR can be classified into several components, based on guidelines provided by the SDR Forum.

- The **radio operation environment (ROE)** consists of the core components that are fundamental to the operation of the radio platform and support its software configurability, such as the radio device drivers, middleware, and some OS services.
- **Radio applications (RA)** control the behavior of the radio by implementing the air interface and modulation and communication protocols. In SDR security, one of the goals is to ensure that the reconfiguration of RA is authorized.
- **Service provider applications (SPA)** include messaging services, video services, voice services, *etc*.
- **User applications (UA)** consist of all end applications installed on the system in which the SDR is embedded, including games, word processing, browsers, *etc.*
- The **user application environment (UAE)** is not in the categories defined by the SDR Forum, because it is regarded by default to be the same as the ROE: both RA and UA are operated in the same environment. In our architecture, however, we separate these two environments and define UAE to be the environment (OS) where UA are executed.

Referring to Fig. 1(a), the component that needs to be protected is RA, because it may otherwise be reconfigured with illegal parameters (frequency, power, modulation, *etc.*). The untrusted component is UA, because the unauthorized and malicious software that is downloaded can tamper with the OS, and then alter the parameters of RA or bypass security mechanisms that are based on the OS. Several methods exist to

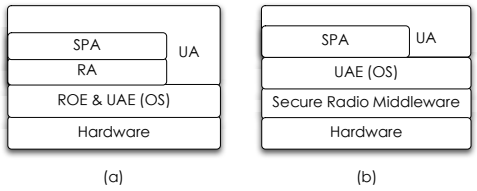

Fig. 1. Traditional and proposed secure architecture of SDR

preserve the integrity of RA. However, they are all vulnerable if the OS kernel itself is tampered with, which can be caused by malicious UA that exploit an OS vulnerability.

The proposed secure SDR architecture is shown in Fig. 1(b). We define a separate layer called *secure radio middleware* (SRM), which includes the most security-critical components, namely RA and ROE. This layer is implemented underneath the UAE (OS), and is, therefore, immune to compromises of the UA and UAE. It contains checking mechanisms that intercept all reconfigurations of the radio and ensure their compliance with specified security policies.

B. Secure Radio Middleware

The architecture of the SRM and its relationship to the OS and hardware are shown in Fig. 2.

*1) **Bypass**:* "Bypass" is the part of SRM where non-critical operations from the OS to hardware go through directly, *e.g.*, LCD, speaker device drivers.

*2) **Memory management unit (MMU)**:* The MMU is important in SRM because it controls the behavior of the OS in memory read and write. In the new architecture based on the SRM, the MMU performs access control and breaks the whole memory space into three parts:

- **Memory where OS has no permission to read or write.** It is used for storage of private keys and secrets that UA should never know.
- **Memory where OS has permission to read-only.** It is used for some modules like the V-HW and security policy monitor discussed below. These modules need to be protected from alteration by the OS.
- **Memory where OS has full permission.** The OS can use this area for its normal use.

*3) **Virtualized hardware (V-HW)**:* V-HW is a critical part of the SRM. All the RA are implemented as V-HW.

Why is V-HW called hardware? In the proposed SDR architecture, the integrity of RA modules can be protected like hardware because they are execute-only to the OS. Even the kernel in the OS does not have the right to write into this segment of memory. Thus, the OS cannot write into V-HW, but only has a control interface to it – just like an ASIC in cell phones nowadays.

And why is V-HW called virtualized? Because it is essentially still software in the SRM. It still has all the nice characteristics of software in the SDR: it can be downloaded through networks, installed under supervision, and altered in different environments. The key point here is that only the SRM, which is more secure than the UAE (OS), can perform these functions.

*4) **Security policy monitor**:* We can ensure the integrity of V-HW through the above mechanism. However, UA and SPA will pass to V-HW the parameters they would like to implement, such as frequency, power level, modulation method, cryptographic algorithms, key size, *etc.* Hence, a security policy monitor becomes necessary, which tries to decide a normal value or range for the radio parameters and compare them to the ones the OS passes to V-HW. If a violation is found, suitable recovery mechanisms are triggerred.

Now two questions arise: (1) what is contained in the security policy, *i.e.*, what characteristic of the transmitted data should be checked by the monitor? and (2) how is the normal value or range decided?

To the first question, the answer depends on the specific RA. The following considerations are often important:

- Which process do the data come from?
- Whether the data should be encrypted – which "cryptographic V-HW" they need to pass through?
- Whether the data should be digitally signed – is non-repudiation required in the communication scenario?
- At which frequency band are they transmitted – what is the parameter in the "modulation V-HW"?
- At what power are they transmitted – what is the parameter in the "power V-HW"?

To answer the second question, various security policies could come from several parties, and the security monitor should enforce the requirements from all the parties. These parties include:

- Independent GPS which gives the location information, thus the local regulatory requirements.
- Security policy received from the base station. The base station may coordinate with all the SDR devices in the region, optimize the performance globally and send out requirements for each device through security policy packets, like frequency and power.
- Security policy received from the primary user. In the scenario that we discussed in Section II related to dynamic spectrum sharing, the secondary user needs to cooperate with these policy packets and comply with the frequency band and power defined in these packets.
- Security policy from other authorities. At some special places, like customs checkpoints at the airport, no communication is allowed. Thus, security policy packets can be distributed to all SDR devices to prohibit any data transmission.
- Applications which need a specific security requirement, *e.g.*, whether the communication in this application is confidential and what cryptographic methods are needed.

IV. Implementation and Results

This section describes our implementation of the proposed architecture, evaluates the performance penalties it incurs, and performs a security analysis of the architecture under various attacks.

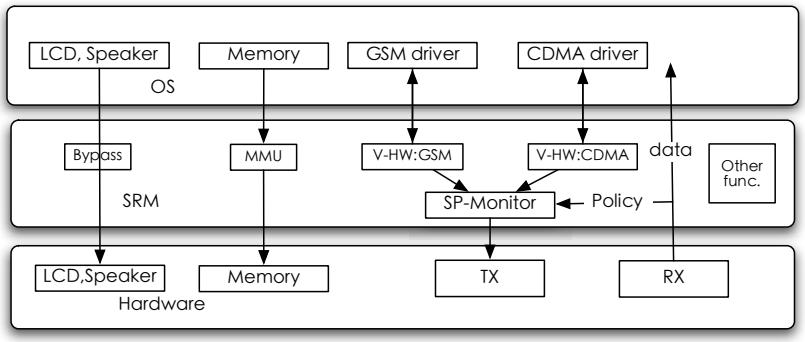

Fig. 2. Architecture of the SRM

A. Implementation Infrastructure

We implemented the proposed architecture using the VMWare Server [16] virtualization framework and the GNU Radio toolkit [17], which we describe briefly.

VMware Server is a free virtualization product for Windows and Linux servers. It creates a virtualization layer above the host OS in which it is installed. This thin virtualizaion layer separates the computation resources in the host machine into several partitions, each of which is called a virtual machine, and each virtual machine is isolated from its host and other virtual machines.

GNU Radio is a free software development toolkit that provides runtime and processing blocks for signal processing to implement software radios using readily-available, low-cost external RF hardware and commodity processors.

B. Implementation Overview

In our implementation, shown in Fig. 3, we realize the SRM in the host OS and the VMware Server layer, and the UAE is realized using a virtual machine. The V-HW and security policy monitor are implemented in the host OS just on top of the hardware, and the Bypass and MMU parts are already present in VMware.

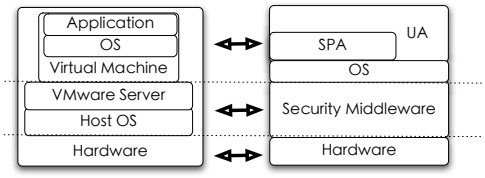

Fig. 3. Proposed architecture and implementation using VMWare

The purpose of our implementation of the proposed architecture is to show that (1) conceptually the architecture can be used in an SDR device, and (2) the overhead of the new SRM inserted into the proposed architecture is acceptable. For objective (1), the implementation architecture is virtually the same as the proposed conceptual architecture. They both have isolated environments for RA and UA, and the security policy monitor and V-HW are also the same. For objective (2), the architecture we implemented obviously has more overhead than the proposed one because a complete host OS obviously has more overhead than a thin SRM layer. However, as shown in this section, the overhead of our implemented architecture is already tolerable, thus achieving objective (2).

In the implementation, the UAE and SRM are both instances of Linux OS. The communication between UAE and SRM (virtual machine and host OS) is passed through a virtual Ethernet interface provided in VMware Server. The data structure passed over this interface includes the following parts:

- Process id: Process id is the identification of which process the data packet is from.
- V-HW id: V-HW id represents the RA module that must be used.
- Parameters: The parameters are the reconfiguration information passed to V-HW and are specific to the V-HW.
- Data: Data are also passed to V-HW for transmission. Either the data or the memory addresses that store the data can be passed here. In our implementation, the data themselves are transmitted from the virtual machine to the host OS, which may result in more overhead, but also leads to greater safety.

C. Illustrative Security Policy

For the purpose of illustration, we take FRS (Family Radio Service) [18] and GMRS (General Mobile Radio Service) [19] as examples. In our implementation architecture, V-HW is an FRS/GMRS module.

FRS is a license-free walkie-talkie system FM (frequency modulation) UHF (ultra high frequency) radio service, while GMRS is a licensed land-mobile FM UHF radio service. Recent hybrid FRS/GMRS consumer radios have 22 channels in all [20].

Suppose process 1 has a license for GMRS, but process 2 does not and only has the right to operate on FRS. The security policy monitor would check the UAE parameters sent to V-HW and see whether the frequency and power are within the regulations for each process: process 1 can transmit data from Channels 1 to 7 and 15-22 with power less than 5W; process 2 can transmit data from Channels 1 to 14 with power less than 0.5W.

D. Results

In this section, we demonstrate that the overhead incurred by the secure SDR architecture is small and tolerable. We use a

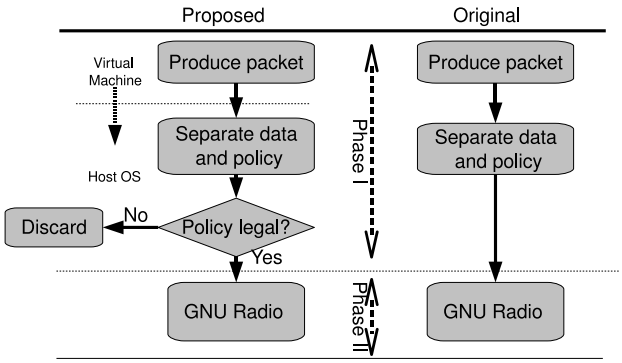

Fig. 4. Performance measurement setup

PC equipped with a 3GHz dual-core Intel CPU and 4GB RAM running Ubuntu Linux 7.10. VMware Server 1.06 is used for virtualization and the virtual machine also runs Ubuntu Linux. GNU Radio version 3.0.3 is used for the V-HW.

Fig. 4 shows the processing flow in the proposed and original architectures. In the proposed architecture, packets containing data and control parameters (frequency, power, *etc.*) are produced in the virtual machine and transmitted to the host OS, which emulates the SRM. After the packet is analyzed, the parameters are checked against our security policy and then transmitted to the GNU Radio for processing. In the original architecture, producing and processing of packets are performed in the same OS layer and no policy checking is implemented.

We measure the processing time for 10000 packets to be produced, passed through different parts, and then processed by GNU Radio. The data in each packet are produced randomly. Reconfiguration occurs once per packet. Thus, the packet size determines the reconfiguration overhead. The larger the packet size is, the longer this set of SDR configuration parameters lasts.

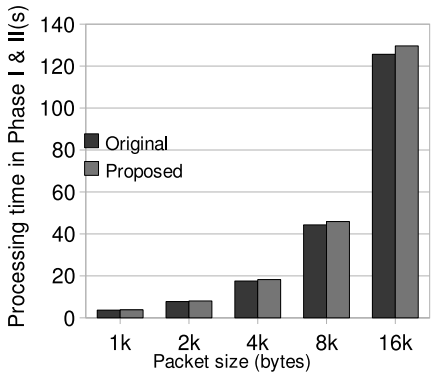

Fig. 5. Processing time for processing 10000 packets

In Figs. 5 and 6, the processing time and overhead for processing 10000 packets are measured for both the proposed and original architectures. We can see that for a packet size of 1kB through 16kB, the overhead is reduced from 4.0% to 3.2%. As the packet size increases, reconfiguration occurs less often and the overhead becomes smaller. It is worth pointing out that the GNU Radio processing part is a very simple FM

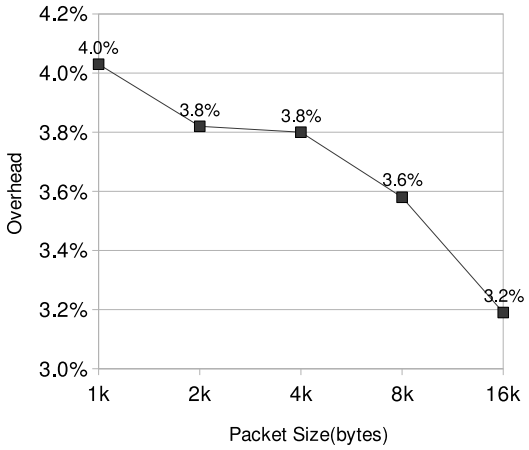

Fig. 6. Overhead for processing 10000 packets

module. If a more complex protocol is used, such as GSM or CDMA, every packet needs more time in Phase II (which requires the same processing time in both the proposed and original architectures). Thus, the overhead would be smaller.

In the results, the overhead of the new architecture comes mainly from two parts. The first part is the communication overhead because we use a virtual Ethernet interface provided by VMware Server for communication from the virtual machine to the host OS. In a more collaborative implementation, we can reduce this overhead even further by designing a more efficient communication mechanism. The second part of overhead comes from security policy check for the reconfiguration parameters, which is unavoidable in a secure SDR architecture. Obviously, in the new architecture, security check is needed only when reconfiguration occurs, and the target is only a set of reconfiguration parameters, which we believe will lead to a smaller overhead than previous secure SDR architectures.

E. Limitations of the Implementation

The implementation presented in this paper is a simple and conceptual one that illustrates the separation of UAE and ROE and the basic mechanism to check reconfiguration parameters. There are some limitations of this implementation, which may serve as the basis for future work.

Firstly, the implementation is based on a PC testbench with generously available resources. However, SDR is mainly meant for more resource-constrained platforms like mobile phones. In our conceptual implementation, the concept of separation of UAE and ROE is realized by virtualization technology. Some alternative light-weight solutions for embedded systems may be Microkernel [21] and multiple independent levels of security [22].

Secondly, the implementation uses two FM radio applications, in which the security policy is not very complicated and the data rate is not very high. A future research direction may be use of more elaborate security policies in complex communication scenarios and algorithm optimization to reduce the overhead of the security policy monitor.

Thirdly, the new architecture and implementation mainly focus on protecting the reconfiguration parameters of V-

HW under malicious UAE (OS) during SDR runtime. Other security mechanisms, like encryption and decryption services, information integrity, access control and secure radio software downloading, are still needed. In the SDR software communications architecture (SCA) [23], all these features are discussed, which our implementation does not include.

F. Security Analysis

The main goal of the new SDR architecture is that, even if the UAE (OS) is tampered with, we can still ensure that the SDR device is in compliance with the security policies. Therefore, for this analysis, we work under the assumption that the OS is already compromised and it can change all the programs and data in the UAE, SPA and UA. The analysis is based on a trusting computing base of all hardware and SRM. We do not consider physical attacks targeted at hardware, nor do we consider a wrongly installed SRM, which should be protected by other security mechanisms, as discussed in Section IV-E. In order to break SDR compliance, the malicious OS could try the following:

1) ***Directly change the hardware parameters****:* The malicious OS could try to modify the device driver between V-HW and real hardware. In the new architecture, there is no interface from the UAE (OS) to the real radio hardware – the SRM separates them to prevent the malicious OS from directly changing the hardware configuration.

2) ***Change the V-HW, MMU or security policy monitor code****:* The malicious OS could try to modify the code of the V-HW, MMU or security policy monitor and then break its compliance. However, these parts are located in the memory where the UAE has read-only access.

3) ***Change the data structure passed to V-HW****:* This is the only interface from the UAE that can be used to "control" the SRM. In the structure of data packets described in Section III-B4, several parts may be maliciously modified:

- *Change the V-HW id or parameters:* The OS could do this to change the transmission format by using a different V-HW module, or to present a V-HW module with illegal parameters or payload data. In such a case, the security policy monitor can check against the installed security policies and discard the data.
- *Change data:* The OS could change the application data to leak secrets from the SDR device, like keys. This is prevented by the architecture because all the secrets are protected and key management is done in SRM memory which cannot be read from or written to by the UAE.
- *Change the process id:* The OS could change the process id to make one process gain the communication rights of another process. In the proposed architecture, SRM can be enhanced to tell which process OS is currently executing through a technique known as introspection [24].

V. Conclusion

In this paper, a new architecture to protect SDR devices from malicious reconfiguration is proposed. In this architecture, an SRM layer is used to separate the most vulnerable environment (UAE) from the most security-critical environment (ROE). The SRM contains a security policy monitor that checks all the RA reconfigurations against the specified security policies. This achieves the goal that even if the OS is compromised in the UAE, secure reconfiguration can still be ensured in the ROE. The architecture is implemented using VMware and GNU Radio. The overhead incurred by the new architecture is shown to be small.

Acknowledgment

We are grateful to Najwa Aaraj for technical discussions.

References

[1] SDR Forum, "Report on issues and activity in the area of security for software defined radio, SDRF-02-A-0003," Tech. Rep., 2002.

[2] ——, "SDR system security, SDRF-02-A-0006," Tech. Rep., 2002.

[3] ——, "A structure for software defined radio security, SDRF-03-I-0010," Tech. Rep., 2003.

[4] ——, "SDR wireless security, SDRF-04-I-0023," Tech. Rep., 2004.

[5] R. Falk, J. F. Esfahani, and M. Dillinger, "Reconfigurable radio terminals - threats and security objectives, SDRF-02-I-0056," Tech. Rep., 2002.

[6] SDR Forum, "SDR security requirements, SDRF-04-W-0006," Tech. Rep., 2005.

[7] R. Hill, S. Myagmar, and R. Campbell, "Threat analysis of GNU software radio," in *Proc. World Wireless Congress*, May 2005.

[8] L. Michael, M. Mihaljevic, S. Haruyama, and R. Kohno, "A framework for secure download for software-defined radio," *IEEE Communications Magazine*, vol. 40, no. 7, pp. 88–96, July 2002.

[9] ——, "A proposal of architectural elements for implementing secure software download service in software defined radio," in *Proc. IEEE Int. Symp. Personal, Indoor and Mobile Radio Communications*, vol. 1, Sept. 2002, pp. 442–446.

[10] A. Brawerman, D. Blough, and B. Bing, "Securing the download of radio configuration files for software defined radio devices," in *Proc. Int. Wkshp. Mobility Management & Wireless Access Protocols*, Sept. 2004, pp. 98–105.

[11] A. Brawerman and J. Copeland, "An anti-cloning framework for software defined radio mobile devices," in *Proc. IEEE Int. Conf. Communications*, vol. 5, May 2005.

[12] H. Uchikawa, K. Umebayashi, and R. Kohn, "Secure download system based on software defined radio composed of FPGAs," in *Proc. IEEE Int. Symp. Personal, Indoor and Mobile Radio Communications*, vol. 1, Sept. 2002, pp. 437–441.

[13] P. Flanigan, V. Welch, and M. Pant, "Dynamic policy enforcement for software defined radio," in *Proc. Software Defined Radio Technical Conf. and Product Exposition*, Nov. 2005.

[14] K. Sakaguchi, C. Fung Lam, T. Doan, M. Togooch, J. Takada, and K. Araki, "ACU and RSM based radio spectrum management for realization of flexible software defined radio world," *IEICE Trans. Communications E Series B*, vol. 86, no. 12, pp. 3417–3424, Dec. 2003.

[15] A. Tonmukayakul and M. Weiss, "Secondary use of radio spectrum: A feasibility analysis," in *Proc. Telecommunications Policy Research Conf.*, Oct. 2004.

[16] "VMware Server, http://www.vmware.com/products/server/."

[17] "GNU Radio, http://www.gnu.org/software/gnuradio/."

[18] "Family Radio Service, http://wireless.fcc.gov/services/index.htm?job=service_home&id=family."

[19] "General Mobile Radio Service, http://wireless.fcc.gov/services/index.htm ?job=service_home&id=general_mobile."

[20] "Frequency Table of FRS and GMRS, http://en.wikipedia.org/wiki/Family_Radio_Service."

[21] "Virtualization for Embedded Systems, http://wiki.ok-labs.com."

[22] "Multiple Independent Levels of Security, http://www.ois.com/Products/MILS-Technical-Primer.html."

[23] M. Kurdziel, J. Beane, and J. Fitton, "An SCA security supplement compliant radio architecture," in *Proc. IEEE Military Communications Conf.*, vol. 4, Oct. 2005, pp. 2244–2250.

[24] S. T. Jones, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, "Antfarm: Tracking processes in a virtual machine environment," in *Proc. USENIX Annual Technical Conf.*, June 2006, pp. 1–14.