GNU Based Security in Software Defined Radio

Dr. H. B. Bhadka
C. U. Shah College of Master of Computer Application, Wadhwan. Gujarat.

ABSTRACT:
Various new technologies are explored for radio communication toward the 21st century. Among them the technology of "software defined radio" attracts large attention. Software Defined Radio (SDR) technology implements some of the functional modules of a radio system in software enabling highly flexible handsets. SDR devices may be reconfigured dynamically via the download of new software modules. Malicious or malfunctioning downloaded software present serious security risks to SDR devices and networks in which they operate. Together with the use of software downloading, future terminals will become a platform to support the deployment of yet unspecified services and applications.
Keywords: *software Radio, communication system, mobile communication standards*

1. INTRODUCTION

For the Third Generation Mobile Communication System represented by the mobile phone and the wireless LAN, there are several international specifications but currently, there is no compatibility among those standards.
SDR is a rapidly evolving technology that is receiving enormous recognition and generating widespread interest in the telecommunication industry. SDR technology facilitates implementation of some of the functional modules in a radio system such as modulation/ demodulation, signal generation, coding and link-layer protocols in software. The hardware of SDR is shown below:

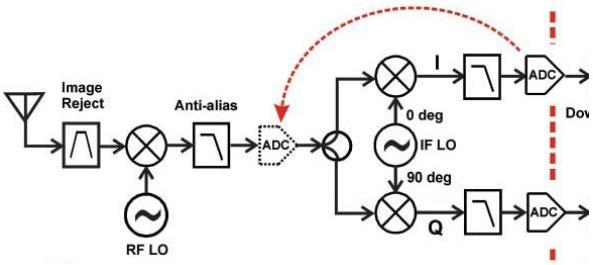

Fig. 1 Simplified Radio hardware Block Diagram of SDR

This helps in building reconfigurable software radio systems where dynamic selection of parameters for each of the above-mentioned functional modules is possible.
Today software radio is viewed more as a technology to enable the reconfiguration of terminals at all stages of design, production and in the field. This paper introduces the concepts of software radio, security and reconfiguration. The potential use of these concepts to provide a terminal with reconfigurable protocol stacks is examined as is the impact of software downloading on 3G and possible future 4G standards. The security concern also affects all ways [1-2, 10].

2. GNU SOFTWARE RADIO

The need for software-defined radios is underlined and the most important notions used for such reconfigurable transceivers are thoroughly defined. The role of standards in radio development is emphasized and the usage of transmission mode parameters in the construction of software-defined radios is described. The software communications architecture is introduced as an example for a framework that allows an object-oriented development of software-defined radios. The GNU Software Radio is a code base of free software that performs digital signal processing using a personal computer and freely available RF receiver front-end designs. Since the system utilizes open-source software and a standard PC under Linux, the GNU Radio receiver is an ideal platform for learning and experimenting with SDR concepts.
In this paper, we will base our experiments on GNU Radio software [4]. GNU Radio is organized around the principle of constructing a graph that describes the data own in the radio system, and then handing that graph o_ to the runtime system for execution. The vertices of the graph are signal processing blocks. The edges are the connections between the blocks. Signal processing blocks have a set of typed streaming input ports and output ports [5]. A dynamic scheduler is used. There are primitive signal processing blocks and hierarchical signal processing blocks. Hierarchical blocks are aggregates of primitive and hierarchical blocks. The primitive signal processing blocks are implemented in C++. All graph construction, policy decisions and non-performance critical operations are performed in Python. All of the underlying runtime system is manipulatable from Python. Python is an interpreted, interactive, object-oriented programming language. It is often compared to Tcl, Perl, Scheme or Java [6].

3. SECURITY CONSIDERATIONS [7-9, 11-12]

Successful deployment of software radios will depend on the confidence that the required security mechanisms are in place to prevent misuse of the system, inappropriate RF emissions, inadequate accounting for services, and compromise of information integrity. To date there are no accepted standards governing what constitutes acceptable software radio security. SDR Forum classifies the collection of software for SDR as the following:

- Radio Operating Environment
- Radio Applications
- Service Provider Applications
- User Applications

Each class of software has own security considerations, but they all share the attribute that downloaded software has the potential of bypassing the installed security functions. Specific threat areas encompass a broad spectrum including network stoppage, unintentional interference, theft of services, and exposure of private and confidential user information. SDR threats may result from the deliberate overt or covert actions of third parties such as viruses, or through human error such as software bugs. The point of attack can be either the network infrastructure or the end user terminal.

Its includes the following parties: manufacturers of the terminal hardware and software; government authorities relevant for SDR and users of SDR terminals. The underlying ideas for system development include employment of four different cryptographic techniques and tamper-resistant hardware. The cryptographic techniques employed are a secret key encryption technique, a public key encryption technique, a technique for cryptographic hashing, and a technique for digital signature.

SDR Forum notes that in all cases, because we are dealing with software radios, the threats can only be realized through the loading, installation and activation of software. Another critical threat arises from downloading certified but buggy or malicious code. In this case, it's detrimental to be able to detect such faulty software and thwart its actions. Thus the primary threat mitigation for SDR security is to prevent the loading, installation and instantiation of unauthorized or unproven software, and detect authorized but faulty software that's already been installed.

4. CONCLUSIONS

In the future, terminals will become "future proof" (within the limitations of the terminal's hardware) by having the capability to download new air interfaces, and operate on new communications standards, using lower layer reconfiguration technology implemented with software radio technology. The main goals of the security system are verification of the declared identity of the source that produces the software to be downloaded, control and verification of integrity of the downloaded data, disabling of the ability to run unauthorized software on the software-defined terminal, and secrecy of the transmitted data to prevent problems such as loss of intellectual property contained in the software.

5. REFERENCES

1. *GNU Software Radio project. http://www.gnu.org/software/gnuradio/.*
2. *Libsafe, project, AvayaLabs. http://www.research.avayalabs.com/project/libsafe/.*
3. *NCASSR Software-Defined Radio research. http://www.ncassr.org/projects/sdr.html.*
4. *Python. http://www.python.org.*
5. *StackGuard, project, IMMUNIX. http://www.cse.ogi.edu/DISC/projects/immunix/StackGuard/.*
6. *Vanu Software Radio. http://www.vanu.com/index.html.*
7. *SDRF wireless security. SDR Forum Input Document, SDRF-04-I-0023, April 2004.*
8. *R. Falk, J. F. Esfahani, and M. Dillinger. Reconfigurable radio terminals - threats and security objectives. SDR Forum Input Document, SDRF-02-I-0056, November 2002.*
9. *L. B. Michael, M. J. Mihaljevic, S. Haruyama, and R. Kohno. Security issues for software defined radio: Design of a secure download system. IEICE Trans. Commun., January 1999.*
10. *Report on issues and activity in the area of security for software defined radio. SDR Forum Approved Document, SDRF-02-A-0003, 2002.*
11. *A structure for software defined radio security. SDR Forum Input Document, SDRF-03-I-0010, May 2003.*
12. *S. Grover. Buffer overflow attacks and their countermeasures. LINUX Journal, March 2003.*