FCC TAC Cybersecurity Working Group
Software Configurable Radios Subcommittee

SCR White Paper
December 2016

Table of Contents

Foreword 4

Acronyms 5

1 Executive summary 6

2 Purpose and Scope 8

2.1 Purpose: 8

2.2 Scope: 8

2.3 Background: 9

2.4 Limitations Of This Study 9

3 Stack and Reference System Architectures 10

3.1 Programmable Firmware 11

3.2 Firmware and ASIC Support 11

3.3 Support Device Specific Functions 11

3.4 Other Functions 12

4 Target (At-Risk) Elements 13

4.1 RF Parameters 13

4.1.1 Operating frequencies (band and bandwidth) 13

4.1.2 Output power 13

4.1.3 Modulation and media access types 13

4.1.4 Smart antenna programming 13

4.1.5 Spectrum sharing algorithms and decision-making processes 13

4.2 Configuration controls 14

5 Attack Surfaces 14

6 Protection Categories or Methods 17

6.1 About Software Security 17

6.1.1 Software Security Methods and Open Source Software 17

6.2 Software Security Methods 17

6.2.1 Methods Review: Image Blocking 17

6.2.2 Methods Review: Partitioning 18

6.2.3 Methods Review: Direct Image Management 18

6.2.4 Methods Review: Digital Code Signing 19

6.2.5 Methods Review: Virtualization 20

6.3 Other Ways to Protect Target Elements 20

6.3.1 Methods Review: 3rd Party Firmware Creators Voluntarily Removing Target Element Modification Controls 20

6.3.2 Methods review: Devices Include Warning 3rd Party Firmware Users 21
6.3.3 Methods review: Supporting self-enforcement 21
7 Impact of Target Element Protection 21
7.1 Tradeoffs with regard to 3rd Party / Open Source Firmware and Software Security 21
7.2 Cost 22
7.3 Design lead time 23
7.4 Manufacturing process changes 23
7.5 3rd party compliance issues 23
7.6 Finding devices that need to be patched 23
7.7 Changing the balance of manufacturer and user control 24
8 Equipment Categories 24
8.1 Professional / Enterprise Systems 24
8.2 Consumer / Entry-Level Systems 25
9 Critical Issues 25
9.1 Potential Weaknesses in Specific Methods 25
9.1.1 Image blocking 25
9.1.2 Partitioning 25
9.1.3 Direct image management 25
9.1.4 Code signing 25
9.1.5 Virtualization 26
9.2 Geolocation 26
9.3 User-configured Domain 26
9.4 Migration of Devices 27
10 Other Considerations 28
10.1 Existing product flexibility 28
10.2 Emulating Mobile OS Upgrade Strategies 28
10.3 Authorizations 29
11 Review of Prior TAC Work 29
Appendix A: Supplemental Information 31
11.1 DFS In Action 31
11.2 Potential Legal Implications of Digital Code Signing Under Open Source Licenses 32

Foreword

This white paper is the product of a subgroup of the FCC's Technological Advisory Council.[1] The Cyber Security – Software Configurable Radio group was assembled in May 2016 to consider the issues discussed in this white paper. Members of the group include:

Alex Salvarani – Nokia
Amit Ganjoo – ANRA Technologies
Brian K. Daly – AT&T
Bruce Oberlies – Motorola Solutions
Christoffer Jerkeby – Ericsson
Dan Torbet – Arris
David Gurney – Motorola Solutions
David Kay – Netgear
Eric Schultz – prpl Foundation
George Popovich – Motorola Solutions
Martin C. Dolly – AT&T
Mike Bergman – CTA (leader)
Mike Geller – Cisco
Paul Steinberg – Motorola Solutions
Pierre de Vries - Silicon Flatirons
Richard Green – Liberty Global
Russ Gyurek – Cisco
Ahmed Lahjouji- FCC
Edna Prado – FCC
Rashmi Doshi – FCC

[1] https://www.fcc.gov/general/technological-advisory-council

Acronyms

ASP	Application Specific Processor
CDMA	Code Division Multiple Access
CPE	Customer Premises Equipment
CRDA	Central Regulatory Domain Agent
CSA	Cloud Security Alliance
CSMA/CA	Carrier Sense Multiple Access/Collision Avoidance
CTA	Consumer Technology Association™
DFS	Dynamic Frequency Selection
DHS	Department of Homeland Security
DSP	Digital Signal Processing
FCC	Federal Communications Commission
FPGA	Field-Programmable Gate Array
FTC	Federal Trade Commission
GPL	General Public License (note also Limited GPL, LGPL; and Affero GPL, AGPL)
IEEE	Institute of Electrical and Electronics Engineers
IoT	Internet of Things
JTAG	Joint Test Action Group (interface)
LAN	Local Area Network
LTE	Long-Term Evolution
NIST	National Institute of Standards and Technology
OFDM	Orthogonal Frequency Division Multiplexing
OFDMA	Orthogonal Frequency Division Multiple Access
OOBE	Out-of-Band Emissions
OWASP	Open Web Application Security Project
PGP	Pretty Good Privacy
QPSK	Quadrature Phase Shift Keying
QAM	Quadrature Amplitude Modulation
RF	Radio Frequency
SoC	System on a Chip
SPI	Serial Port Interface
TAC	Technological Advisory Council
TDMA	Time Division Multiple Access
TEE	Trusted Execution Environment
TPM	Trusted Platform Module
uP	Microprocessor
WLAN	Wireless Local Area Network

1 Executive summary

In 2016, the FCC Technological Advisory Council Cybersecurity Work Group was asked to investigate the following proposition:

How to strike the appropriate balance between embedding frequency security mechanisms into Software Defined Radios while allowing innovation and the flexible addition of features.

A subgroup, TAC Cyber Security – Software Configurable Radios (CS – SCR), was formed and pursued the question over the ensuing months.

Software Configurable Radios (SCRs) — a more general case of Software Defined Radios — have operating parameters controlled by internal programming. Should such a device be hacked, the radio may be moved to an operating frequency or power level other than that for which the device was certified. This situation has evolved over time, and the FCC has taken various steps to protect services from the consequences of this operation.

This white paper focuses on responding to this request. However, the scope of the problem can be rather broad and general, given the flexibility of modern wireless chipsets. This paper discusses multiple aspects of this challenge, but the most common example used is that of 5 GHz Wi-Fi routers supporting Open Source Software (OSS). The intent is to consider the broader generic challenge in search of broadly applicable solutions, balancing concerns from various stakeholders.

Devices are certified for operation in the U.S. based on certain critical RF parameters such as transmit power and occupied frequency (the "target elements" discussed in Section 4). Manufacturers put significant effort into meeting these requirements. At the same time, a robust user community for open source software seeks to modify these devices, seeking improved performance and enhanced security; and doing wireless research.

Given such a device, the challenge is to protect these target elements while protecting the needs of the stakeholders.

More generally, there is a tension between embedding security mechanisms such as those described in this paper, and improving the cybersecurity and functionality of systems taken as a whole. In fact, using some of the methods described in this paper could block 3rd-party attempts to address vulnerabilities and thus eliminate a path to improved overall security.

Recommendations:

During this process, the group reviewed the current FCC guidance in KDB 594280, "Software Security Requirements for U-NII Devices" (November 2015). This document includes important questions about the software security and configuration controls. However, feedback from FCC staff indicates that the corresponding information provided by manufacturers is not helpful and that the certification process is still insufficient to prevent interference issues due to unauthorized modifications. The effort of this group is broader than U-NII devices, but we found this example to be useful in discussing process.

We recommend revising this type of guidance to be more specific about disclosures manufacturers should make about the methods they employ, for example by reference to the methods analysis provided in this white paper.

We found that digital code signing is the method preferred by industry when it is necessary to control images loaded onto devices. However, digital code signing does not strike the desired balance, as all parties other than the manufacturer are unable to modify code on the device.

We also found that the number of stakeholder groups is higher than expected. While this effort had participation from manufacturers and the open source community, there are also humanitarian and rescue groups, radio amateurs, third-party firmware creators, user rights activists, security officials, makers, researchers, and consumers to consider.

Therefore, this group recommends that the FCC encourage the formation of a multi-stakeholder forum to find a way in which manufacturers can strike the appropriate balance between embedding security mechanisms into software configurable radios and their ecosystem to ensure compliance with FCC service rules, while allowing innovation and the flexible addition of features, and fostering cybersecurity overall. One trade association has agreed to consider hosting such a forum and some members of the 2016 subgroup are available to continue into 2017 to provide continuity and complete this effort.

2 Purpose and Scope

2.1 Purpose:

The purpose of this document is to provide industry feedback to the United States Federal Communications Commission (FCC) on handling of software configurable radios by analyzing the technical space and characterizing the issues and the mitigation methods. The document will define design parameters that need to be considered both by the FCC regulations and then by manufacturers. The goals are to encourage innovation in new products, while avoiding harmful interference.

2.2 Scope:

The scope is limited in this document to devices that intentionally transmit RF energy for any purpose at any frequency, which can be tuned or altered by software programming. Devices that transmit energy and their transmission frequencies and levels are controlled by mechanical means such as antennas, variable components such as capacitors, resistors or inductors, and devices that are purposely changed by mechanical means are not covered in the scope.

The term "Software Defined Radio" (SDR) is defined specifically in 47 C.F.R. § 2.1 with certain limitations. Specifically, "*only radios in which the software is designed or expected to be modified by a party other than the manufacturer*"[2] is part of the criteria that define SDRs under § 2.1. The scope of this effort must include devices where it software reconfiguration of RF parameters by third parties is technically feasible but not intended by the manufacturer. Therefore this scope is in relation to Software Configurable Radios.

The scope is also limited to seeking to prevent interference-causing modifications made by those physically in possession of the device under consideration, from the time it leaves the manufacturer to the time of operation. While it is also meaningful to consider modifications made over the internet or over a local intranet, or by spoofing the local RF environment (e.g. faking GPS signals), the focus of the effort was on an existing situation, that of modifications made by equipment owners leading to real interference. However, it is believed that the software image protections discussed in this paper may also help protect against internet or intranet based attacks. There are also other groups considering the more general IoT problem, that of devices being attacked over the internet. This group's task is more specific to devices-in-hand.

[2] U.S. Government Publishing Office, Electronic Code of Federal Regulations at http://www.ecfr.gov/cgi-bin/text-idx?SID=691dc7599174d1e8e1f1f38d531e2fc5&mc=true&node=se47.1.2_11&rgn=div8.

2.3 Background:

Technology used in 2016 by silicon ASIC (Application-Specific integrated circuit) chip vendors has yielded versatile designs, including development of products that can maintain a single hardware definition but have a broad range of applications. Technology advances have led to innovative products that fill the needs of US consumers, including products for wireless data exchanges and telephony. This, however, has created a dilemma for FCC regulators, as it raises the possibility of intentional and inadvertent modifications being made in software that would cause products to produce unwanted interference. The question posed is, can SCRs be regulated so that innovation can continue, while preventing harmful interference? Also, what is the reasonable expectation of manufacturers, retailers and end users?

It is acknowledged that it is impossible to prevent extreme cases where a "hacker" has intimate design knowledge and maliciously alters the design of a device. Nevertheless, it should be possible to prevent typical users, or even somewhat advanced users, from using modified products in a manner inconsistent with FCC regulations.

The scope of the SCR problem can be rather broad and general, given the flexibility of modern wireless chipsets. For example, one can imagine a wide variety of radio-related hazards to aviation alone, of which unauthorized modification of SCRs is just one:

1. 5 GHz Wi-Fi routers supporting open source software (OSS) are reprogrammed to disable DFS, degrading TDWR
2. Imported, non-US U-NII base stations with DFS disabled disrupting terminal Doppler weather radar (TDWR)
3. Operator of U-NII base stations disables DFS (e.g. using the manufacturer's admin password), degrading TDWR
4. DFS signatures U-NII devices comply with will work for DoD radars, but not TDWR, degrading TDWR by not disabling all harmful operation
5. Spoofing ADS-B using off-the-shelf SDRs, disrupting air traffic control
6. Spoofing GPS using off-the-shelf SDRs, disrupting aircraft navigation

This paper will discuss multiple aspects of this challenge, but the most common example used will be 5 GHz Wi-Fi routers supporting OSS. The intent is to consider the broader generic challenge in search of broadly applicable solutions. However, this approach does not take into account the likelihood and consequences of various specific hazards.

2.4 Limitations Of This Study

This paper primarily considers the effect of SCR target element protection methods on unlicensed operation of devices used for wireless networking (Wi-Fi). SCRs are used for other purposes, including by amateur radio operators, emergency and rescue personnel, government agencies, the military as well for other uses such as Bluetooth and 802.14.5 low-power networking/ZigBee. Decisions on target parameter protection methods, whether from manufacturers or regulators, may differ upon consideration of different use-case for different devices. Further research and exploration is needed to more fully consider these use-cases.

This group did not explore hardware-based protection mechanisms. Hardware-based protection mechanisms are used in ham radios and include preventing unwanted use-cases from the keypad and use of jumpers or diodes inside the radio which, when present, prevent use in ways outside of regulatory limits. Further study is needed to fully consider hardware-based

protection mechanisms and their relative advantages and disadvantages to other target element protection methods.

3 Stack and Reference System Architectures

To consider SCRs in depth, we first consider generalized architectures for these systems.

Figure 1 below shows the typical OSI 7 layer communications networking system model, with color coding to indicate the most vulnerable layers and functions. Layers or functions that control RF physical layer parameters, such as operating frequency, transmitter spectral characteristics (e.g., spectral shape/bandwidth/OOBE, power level, etc.), or channel access capabilities (e.g., DFS sensing capability) are the most vulnerable to misconfiguration.

Functions shown in red are the most vulnerable, followed by orange and yellow functions. Higher layer functions such as the data and network layers (e.g., media access control (MAC) and link layer control (LLC)) may also need to be secured, to the extent that they control channel access or band selection (or similar functions). Higher layer functions (e.g., application and presentation layers) may require security mechanisms for controlling RF parameters by those in possession of the device if configuration interfaces are accessible at those layers.

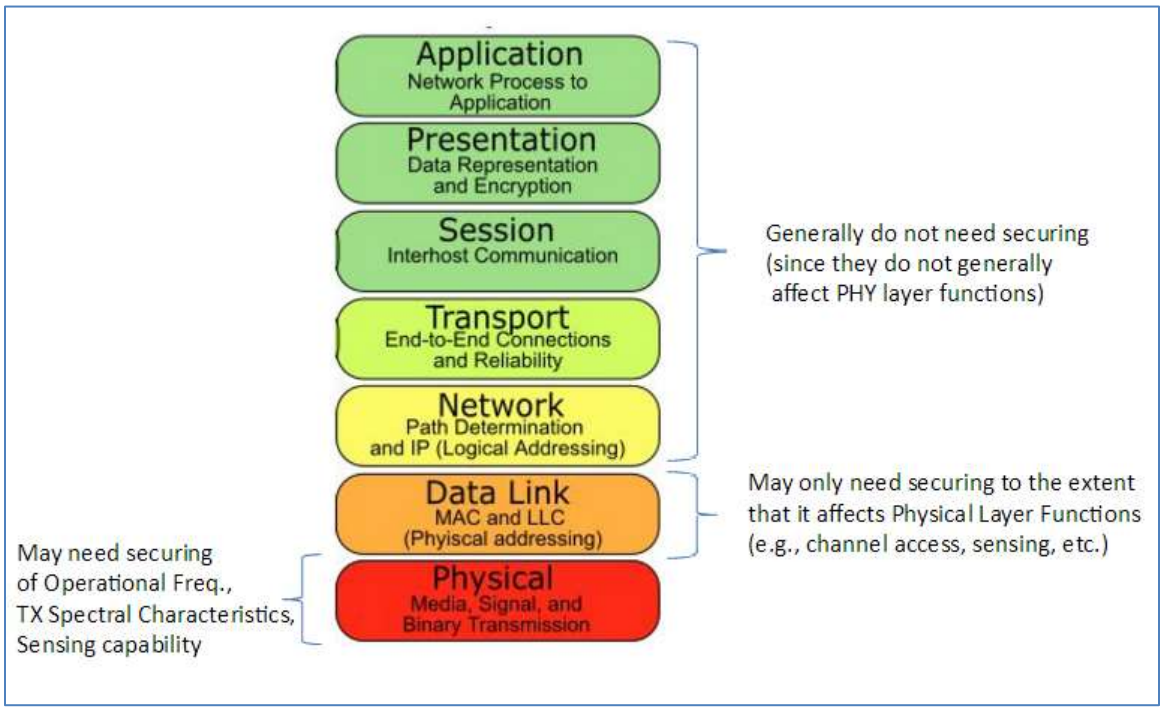

Figure 1: OSI Communications and Vulnerability Model

Though the control means of these functions are highly implementation dependent and varied, they may often be controlled by firmware and SPI/USB/JTAG communications interfaces or

other programmable hardware (e.g., FPGAs). Figure 2 below depicts a reference system architecture which can be used to describe where these functions may reside. The architecture elements that are considered in this document are described below.

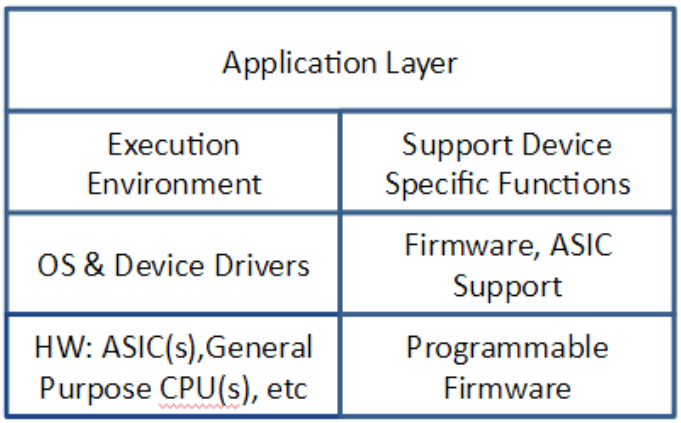

Figure 2: Reference System Architecture

3.1 Programmable Firmware

Operating parameters in this architectural element include radio signal processing functions that include modulation, filters, as well as spectrum sensing algorithms that determine transmitter spectral characteristics and channel access availability. These functions may be implemented through immutable software implementations or software downloadable implementations, or combinations of both. Typical target devices for programmable firmware include FPGAs, DSPs, ASPs, and general purpose CPUs. Some functions may reside in dedicated hardware.

3.2 Firmware and ASIC Support

This architectural element includes software that controls the RF physical layer parameters, such as operating frequency, transmitter spectral characteristics, transmit power and channel access capabilities (if any). Higher layer functions such as the data and network layers (e.g., MAC and LLC) may also be found here. These functions may be implemented through immutable software implementations and/ or software downloadable implementations. Typical target devices are general purpose CPUs. Note that Moore's Law has increased the processing power of modern CPUs enabling them to perform both Programmable Firmware and Support Functions in a single CPU for some radio protocols. Multi-core CPUs also enable these types of functions.

3.3 Support Device Specific Functions

This architectural element includes general functions that support peripheral interfaces to a radio. Examples include A/V interfaces (e.g., audio, camera), GPS, SPI, USB, JTAG, flash memory,

etc. These functions may be used in certain cases as a method to change the firmware that impacts the transmitter and channel access capabilities. For example, some devices use JTAG ports as a debug and test interface. As such, those interfaces that allow access to target RF parameters would need to be secured (see below).

3.4 Other Functions

Other categories, such as applications, OS & Device Drivers, and fixed ASIC/hardware functions generally don't directly impact altering target RF operating parameters. One exception to this case would be where an OS or driver could be altered and inserted into a software environment, assuming that such an OS or driver could alter a target element. In general, approaches to protect devices can be utilized to combat against such vulnerabilities and are discussed in more depth in section 6, "Protection Categories or Methods".

4 Target (At-Risk) Elements

This section identifies and describes the at-risk technical (operating) parameters for a SCR in the context of interference prevention. While there may be good reason to seek to protect other elements of the SCR, this study is only concerned with those parameters that, if modified, present an increased risk or occurrence of harmful interference in the U.S.

Note that the security problem considered here is limited to that of interference protection—the issues and concerns related to a modified device causing interference, not the more general case of a modified device causing a generic problem (for this reason, use cases like devices being compromised to become part of a botnet are out-of-scope).

4.1 RF Parameters

The operating parameters considered in this section are those potentially under software control in an SCR and that affect the transmitted waveform in ways that pertain to interference potential, either in the certified band of operation or in other bands.

4.1.1 Operating frequencies (band and bandwidth)

This category includes all the ways to specify and control the intended span or spans of transmitted frequencies such as center frequency, occupied band or bandwidth, lower band edge, upper band edge, etc. Software controls may include programmable filter bank or frequency synthesizer controls.

4.1.2 Output power

This category includes all the ways to specify and control the intended in-band output power, such as peak, average, and channel power. Software controls may include variable gain or switchable output controls.

4.1.3 Modulation and media access types

This category includes amplitude and phase modulation techniques, usually grouped in methods such as QPSK, n-QAM and OFDM. It also covers link layer multiple access control techniques such as CSMA/CA, TDMA, CDMA and OFDMA. Software programming in FPGA, DSP or SoC usually controls these functions.

4.1.4 Smart antenna programming

This category includes several related areas, including aiming a directional antenna, modifying the beam of a phased-array antenna, and modifying the time slicing of an antenna system.

4.1.5 Spectrum sharing algorithms and decision-making processes

Spectrum sharing devices may be required to check a database, like a white space device ("WSD"). For example, in the IEEE 802.11af standard, "Spectrum sharing is conducted through the regulation of unlicensed WSDs by a geolocation database."[3]

[3] Adriana B. Flores et al, *"IEEE 802.11af: A Standard for TV White Space Spectrum Sharing"*, IEEE Communications Magazine, October 2013

Along the lines of a geolocation database, the Linux Wi-Fi regulatory database toolkit (wireless-regdb) is a set of tools and data available for integration into devices. The database includes digitally signed information about wireless regulatory limits worldwide. When a device detects a regulatory domain change (for example by observing an access point with country information), the kernel can request regulatory information to update the drivers. Permitted power, channels, and DFS compliance are all part of this database. New submissions are cryptographically signed and regularly updated.[4]

Another method used is Dynamic Frequency Selection (DFS). In DFS, the access point (AP) automatically selects a channel based on a listen-before-transmit mechanism (i.e., the level of observable interference in the channel).

Changing any parameters in these algorithms, such as thresholds while listening; frequency or time of listening windows; location or contents of databases; or actions taken before or after executing the algorithm must be secured against.

This challenge is both a hardware issue and a software issue. For example, DFS could, in theory, be implemented in hardware.

4.2 Configuration controls

The following are examples of configuration controls:

- Master vs. client controls (as per 47 C.F.R § 15.202)
- Regional controls (keeping to U.S. compliant behavior vs. the RF compliance requirements of other regions of the world)
- Module controls
- Operational mode: Ad-hoc/p2p/mesh (for 802.11); bridge/mesh; master/client; and p2p/p2mp (for access points)
- Antenna configuration
- Out-Of-Band Emissions (OOBE) (when software-configurable)

In each case, these are controls available without modifying the hardware or software. Proper authorization and authentication controls are required if these configuration controls can modify Target Elements (such as by changing transmit power by changing to a different pre-programmed configuration that matches a different regulatory environment).

5 Attack Surfaces

The term of art in cyber research for the point of entry for modifying the device outside its authorized operation is “attack surface” (and more generally, any unauthorized cyber operation on any kind of system must go in via an attack surface).

In the case of SCRs, not all “attacks” are malicious. In some cases, the user simply wants more performance and may be ignorant of the rules. It is also important to note that not all modifications are against the rules; such changes are not attacks and instead may be means to

[4] https://wireless.wiki.kernel.org/en/developers/regulatory

change or improve performance of the device. Our focus is on the modifications that may change the Target Elements as described in the prior section.

Because it is commonly understood in the industry and literature, we will use the term of art “attack surfaces” to be clear from an engineering point of view without judgement on the motivation of the operator.

The table below highlights some possible radio/RF parameter attack surfaces and the associated risk level for tampering/modification associated with each interface.

Attack Surfaces (for Target Element modification)	**Process**	**Relative Ease of Attack**	**Comments:**
JTAG port	Open case, identify port, attach interface and instrumentation, modify binary	Moderately Difficult	Requires case intrusion, specific design knowledge, JTAG equipment and experience.
SPI port	Open case, identify port, attach interface and instrumentation, identify method to affect code, modify binary	Moderately Difficult	Requires board intrusion and specific device programming knowledge
Internal (SoC/uP) bus	Open case/chip, identify bus, attach interface and instrumentation, identify method to affect code, modify binary	Highly Difficult	Requires chip intrusion
Software/Firmware Configuration	Use designed-in method to update code	Either Easy, or Highly Difficult	Highly difficult if secured (e.g., partitioned, digitally signed/authenticated code, etc.) Easy if not secured.
Internet-based hacking	Use over-the-wire debug interface to access device	n/a	Out-of-scope for this paper[5]

[5] These attack surfaces are not considered in depth here, as it is a general case of the security of IoT devices. However, it can be noted that good design practice for a connected device includes disabling standard internet based interfaces except those absolutely necessary and under solid authentication/authorization control.

6 Protection Categories or Methods

This section considers potential software approaches for protecting the target elements discussed in the above sections. Some limitations are discussed in Sections 7 and 9.

6.1 About Software Security

The purpose of software security is to ensure the authenticity and integrity of the device software (or firmware) image and stored data. "Encryption" is often discussed in this context; here we focus on authenticity and integrity and treat encryption as a design choice (albeit a common one).

In addition to the above requirement, any controls that the manufacturer provides to alter target elements outside the regulatory limits for the U.S. (such as to allow a band or transmit power for another country or regulatory regime) must be difficult to circumvent. Debug interfaces must also be considered..

However, there is a business requirement that authorized third parties must be able to make modifications. Sometimes this requirement includes a need to support privilege tiers or levels of rights.

Finally, it is a good best practice to maintain event logs suitable for forensic analysis.

6.1.1 Software Security Methods and Open Source Software

Restrictions on using modified code can cause severe damage to open source software and its community around it. One of the principles behind open source software is the freedom to modify the software on a device you own. All mentioned software security methods severely limit that freedom. Additionally, a diverse community of individuals is required to develop the highest quality, safest and most innovative open source software. Any restrictions that prevent dedicated users from improving their software will potentially endanger innovation as well as the people who use it. Technical restrictions on modifying software inherently imply preventing at least some open source developers from innovating, experimenting and improving software controlled radios.

6.2 Software Security Methods

Consistent terminology will aid in verifying the protection of a device. Here we discuss several available methods. Note that none of these methods would necessarily fully protect the target elements and in some cases one method may need to be combined with other methods, if used.

6.2.1 Methods Review: Image Blocking

Image Blocking refers to designed-in means to prevent any change of the image by other than the manufacturer. Because the manufacturer requires the ability to update the image post-sale, there will generally be a secure method in the device that the manufacturer can use but unauthorized third parties cannot (or should not be able to).

An assertion of Image Blocking does not, therefore, convey information about how the blocking is performed.

An example of Image Blocking is seen in the public statement by TP-Link in August 2016. At that time, TP-Link stated that the company made changes to its router configurations, resulting in the inability to flash open-source, third-party firmware to the devices. However, their statement also indicated that there was an opportunity for third-party software/firmware developers to "demonstrate how their proposed designs will not allow access to the frequency or power level protocols" in the TP-Link devices.[6]

6.2.2 Methods Review: Partitioning

For a Software [Configured] Radio, such as a WLAN device, the software can be divided into partitions or physical code blocks. Some partitions can be readily accessible and others can be built-in with the manufacturer driver and cannot be accessed.

For example, methods to control the transmit power from a given radio can be included in the wireless driver firmware during manufacturing. During FCC approval testing, a power table is would be obtained from the certification Lab, and the levels given to the chipset manufacturer who then manufactures a permanently "burned" flash memory with corresponding wireless driver specifically for a given model.

In addition, some implementations could employ software partitions that can be permanently secured by the flash memory. Examples of this include frequency selection and DFS timing. In these cases, the software "driver" accepts input from an API interface in the OS so that a lesser percentage of the transmit power can be used.

Software Partitioning can also be used for managing operational parameters that are configured ot obtained at run-time. Examples of this type of configuration data include IP protocol settings, user authentication, Quality of Service configuration, port forwarding, and other applications to meet user's needs. These partitions may or may not be accessible to the end user, but can be defined to open source developers to enhance the user experience. These features are not considered essential for regulatory compliance and access is granted based on marketing decisions for the product.

6.2.3 Methods Review: Direct Image Management

In this case, a central authority directly manages firmware. This may be the case when the hardware in question is owned by a service provider, such as a DSL or cable modem. However, the model is valid on a technical basis assuming the parties involved reach an appropriate business agreement.

However, if this is the only method for loading firmware onto devices post-sale, the goals and assumptions of permitting third-party / open-source firmware and fostering innovation are not met. Additionally, depending on the open-source license of the software included from the central authority, this may preclude open source software under certain open source licenses from being included on the firmware.

---

[6] TP-Link, "TP-Link Statement and FAQ for Open Source Firmware", http://www.tp-link.us/faq-1058.html, 8/5/2016

6.2.4 Methods Review: Digital Code Signing

There are two ways digital code signing can be helpful.

- By design, a device can detect non-manufacturer firmware and warn users that by using such an image, they are responsible for potential interference issues. This approach assumes a user interface.
- By design, the manufacturer can block outright attempts to use an image that is not signed by the manufacturer or other authorized developer. This approach would be typical of a managed ecosystem, for example a DOCSIS router supplied by a cable company.

There are several ways in which digital code singing can be implemented typically based on either use of a centralized Certificate Authority (CA) or the use of a non-centralized web of trust methodology.

A good summary of how digital code signing techniques with a centralized CA can be applied to firmware images can be found by reviewing the advice to developers which is available from the CA Security Council[7]:

> *In order to sign the code, the publisher needs to generate a private-public key pair and submit the public key to a CA along with a request to issue a code signing certificate. The CA verifies the identity of the publisher and authenticates the publisher's digitally signed certificate request. If this vetting and key-verification process is successful, the CA bundles the identity of the publisher with the public key and signs the bundle, creating the code signing certificate.*
>
> *Armed with the code signing certificate, the publisher is ready to sign the code. When the code is signed, several pieces of information are added to the original file holding the executable code. This bundled information is used by the recipient's user agent to authenticate the publisher and check for code-tampering.*
>
> *The entire sequence for bundling the digitally signed code takes place as follows:*
>
> - *A hash of the code is produced*
> - *Public-key algorithms are inefficient for signing large objects, so the code is passed through a hashing algorithm, creating a fixed-length digest of the file*
> - *The hash is a cryptographically unique representation of the file*
> - *The hash is only reproducible using the unaltered file and the hashing algorithm that was used to create the hash*
> - *The hash is signed using the publisher's private key*
> - *The hash is passed through a signing algorithm using the publisher's private key as an input*

[7] Bruce Morton, CA Security, "Code Signing", October 2013, https://casecurity.org/author/bmorton/

    - *Information about the publisher and the CA is drawn from the code signing certificate and incorporated into the signature*
- *The original code, signature and code signing certificate are bundled together*
    - *The code signing certificate key is added to the bundle (as the public key is required to authenticate the code when it is verified)*

Code signing is required to install code on many platforms because it provides assurances of authenticity and origin. When signing code you have to make a few decisions to protect your deployed software. The most important decision when establishing end-user trust is that the signed code is backed by a code signing certificate issued by a trusted Certification Authority. Self-signed certificates should only be used for testing, not for production releases.

The second most important step is to timestamp your code. In the event of a compromised key, your time-stamp may ensure that your code is protected even if your certificate needs to be revoked.

As far as alternatives to using a centralized CA, code signing without centralized certificate authorities is also commonly used by software authors, software distributions, and firmware using, for example, a PGP based infrastructure.

6.2.5 Methods Review: Virtualization

There has been some discussion of separating code in distinct virtual machines. A paper presented at the OpenWRT Summit in October 2016 proposed running OpenWrt in its own virtual machine on a MIPS chip.[8] The initial prototype of this solution highlighted a method of separating the wireless radio stack into a separate virtual machine from the main OpenWrt system. The initial prototype provides no mechanism for preventing modification to the VM containing the wireless radio stack. While this prototype shows potential, the method is new and not widely adopted, nor do all chipset platforms support the right kind of virtualization.

6.3 Other Ways to Protect Target Elements

This section discusses additional methods that are not strictly speaking software security. While these methods are not generally "secure", they can help reduce the scale of the problem and are worth considering.

6.3.1 Methods Review: 3rd Party Firmware Creators Voluntarily Removing Target Element Modification Controls

While open source provides the ability of users to modify and improve software, most users tend to download binary of the open source and rarely, if ever, modify it. Open source 3rd party software firmware users are no different in this respect. This puts open source 3rd party firmware creators ("creators") in a position to prevent inappropriate usage. Creators may be encouraged to not release binaries which provide UI elements and APIs for modifying Target Elements outside of FCC regulations. While a user could still modify the source and rebuild the

[8] Michael Hohmuth, Kernkonzept, "FCC compliance with open source: Running OpenWrt in a VM on top of the L4Re microhypervisor", October 2016, http://openwrtsummit.org/

code in a way which allows Target Element modification, this would require a significant amount of time, knowledge and dedication that the average user would not have, thus reducing the scale of the problem.

6.3.2 Methods review: Devices Include Warning 3rd Party Firmware Users

The user interfaces in most routers have a mechanism for uploading and installing a new firmware. In the past, most routers would not notify the user if they were uploading a new firmware which didn't come from the manufacturer. In order to discourage uneducated users from installing 3rd party firmware, manufacturers could add a feature to verify whether an uploaded firmware came from manufacturer. If it did not, the UI should provide a warning telling the user the potential risks of interference from 3rd party firmware and then confirm that the user still wants to replace the firmware. Such a warning would provide an educational opportunity and discourage uneducated users from using 3rd party firmware.

6.3.3 Methods review: Supporting self-enforcement

Self-enforcement of FCC regulations by ham radio operators has been the norm for decades. Through self-enforcement, ham radio operators have protected the spectrum and made the FCC's enforcement work significantly easier. Given the large overlap and compatible ethos between the ham radio and open source software community, a similar mechanism may work for reducing interference from all SCRs. A community of dedicated open source developers could be recruited to look out for interference and track down perpetrators. This effort could be supported through software for smart phones, routers, computers to voluntarily notify the FCC of cases of interference. Along with the location of the report, this would provide the FCC with data to address interference.

Another potential partner for self-enforcement is through industry. As of 2013, the Wireless Internet Service Provider Association has voluntarily created a system for collaborating with federal authorities and assisting WISPA members in obeying the law as it relates to TWDR[9]. Additional free-market collaborations may provide additional data for regulators while reducing interference.

7 Impact of Target Element Protection

7.1 Tradeoffs with regard to 3rd Party / Open Source Firmware and Software Security

Open source software provides great benefits to both industry and the community. Most Internet of Things devices are based upon a distribution of Linux. OpenWrt is a community maintained Linux distribution for embedded devices that has been in existence for over ten years. OpenWrt is part of a default base support package provided by some of the largest chipmakers for integration into wireless devices. Firmware from industry often consists primarily of software created by the open source community. Having fewer devices to experiment and improve will likely harm these open source communities thereby harming the quality of software available to industry and, ultimately, consumers.

[9] UNII Device Interference Advisor (UDIA), http://udia.spectrumbridge.com/udia/home.aspx

Open source, modifiable software, including at the lowest levels of the driver and firmware, can be seen as a contributor to innovation. The Make-Wifi-Fast project used fully open source wireless and unrestricted radio drivers (ath9k) to eliminate a buffering problem, reducing latency by a factor of 10; and to implement airtime fairness to increase average transmission speed by over 400% in some scenarios[10]. Other research using open source firmware and drivers addresses interference by reducing transmission power on wireless stations and clients when the system senses it would have no effect on user performance.[11] Ad-hoc Mode Wi-Fi, used primarily for mesh networking, was not well supported by vendors early on in the history of Wi-Fi. It was improved over time by open source developers collaborating and competing in events like Battlemesh held annually in Europe.[12]

Finally, the ability to install and modify customized software, including in areas that deal with target elements, allows innovation and experimentation in ways that may be unanticipated by the manufacturer or not seen by manufacturers as cost-effective. The Make-Wifi-Fast initiative engaged in unique and beneficial research that required access to the MAC layer and lower-level driver software. Other research using open source firmware and drivers addresses interference by reducing transmission power on wireless stations and clients when the system senses it would have no effect on user performance.[13] Ham radio operators install Broadband-HamNet to use routers and devices on ham bands to assist in disaster recovery and preparation.[14] As far as academic research goes, over 8,200 research papers reference a particular brand of wireless radios with open source drivers[15], many of these projects involving modifying software to study speed, interference or even ways to use the Wi-Fi signal to measure the environment.

To restrict the modification of a device without regard for these beneficial applications is to risk losing these benefits.

7.2 Cost

Moving from a non-secure platform to a signed-code platform will obviously impact cost. Manufacturers are forced to pay close attention to the cost of goods. It is not always well understood by the public how much a small cost increase—say, 1% of the retail price—will affect sales and profits for a manufacturer.

Small cost increases have a significant effect on profitability for two reasons. First, consumer electronics are sold at discrete price points. A device sold at retail for $99.95 with a (for example) $1 cost increase cannot be sold at $100.95; instead it may go to $109.95 or $119.95.

---

[10] Speeding up WiFi (whilst saving packets), https://youtu.be/fFFpo_2xlfU?list=PL3bvPCw5QCLJ0xR1oXui6M7QBh6UElgvn

[11] https://github.com/thuehn/Minstrel-Blues

[12] Battlemesh HomePage, http://battlemesh.org/

[13] https://github.com/thuehn/Minstrel-Blues

[14] http://www.arrl.org/news/broadband-hamnet-wins-international-association-of-emergency-managers-awards

[15] https://scholar.google.com/scholar?hl=en&q=atheros+802.11&btnG=&as_sdt=1%2C5&as_sdtp

The consumer does not treat such price increases favorably, so there is a strong incentive for a manufacturer to keep costs down.

The second reason is that a $1 increase in cost must be considered in the context of the net profit. Assuming about 5.5% net profit for consumer and office electronics[16] and assuming the manufacturer sold the device to the retailer for no more than 70% of the retail price, an increase in the manufacturing cost by an amount equal to 1% of the retail price will eliminate over 25% of the profitability of the product.

These are only examples to show how a small increase in cost has a large impact on price and profit (again, for example $1 out of $99.95).

For this reason, digitally signing code is far more common in higher-end platforms than in lower-end platforms (see also section 8, "Equipment Categories").

There are other long term costs not related to the manufacturing cost of the SCR device. There is a long term lifecycle - notably: support, bug fixes, security and regulatory updates, and ultimately - disposal, which may far outweigh the manufacturing cost of the device, and need to be covered during the manufacturing, pricing, and regulatory approval process. Decisions made in the mechanism for securing the target elements, both by manufacturers and regulators, will affect the long term cost of a device. Some examples of this long term cost effect are in sections 7.7 and 7.8.

7.3 Design lead time

Different product categories are on different time-to-market calendars, driven by the specific niche markets themselves. Some products are on annual update cycles, some are on shorter cycles. Prior to that point is the time available for development. Increased development time can have a significant impact on time-to-market. However, after the initial design change, product increments (adding features to an existing model to create the next-year's model) have less design lead time impact.

7.4 Manufacturing process changes

Depending on the security method chosen, the manufacturing process may require the addition of late-stage programming of devices prior to packaging or other changes. Like design lead time, subsequent model years may be less impacted.

7.5 3rd party compliance issues

For products where the image is protected, a 3rd-party repair center will need training and equipment.

7.6 Finding devices that need to be patched

Consumers' response rate on so-called "bounce-back" cards or warranty registration cards is typically in the 3-5% range in the consumer technology industry. Generally, unless there is a reason otherwise, manufacturers are not usually aware of who has the products that were

[16] Prof. A. Damodaran, Stern School of Business at New York University, "*Margins by Sector (US) (January 2016)*", http://pages.stern.nyu.edu/~adamodar/New_Home_Page/datafile/margin.html

manufactured and sold. Consumers and installer may also opt-out of updates and turn off auto-update mechanisms.

7.7 Changing the balance of manufacturer and user control

Most image protection methods would give manufacturers greater control over whether and how a device is updated. Depending on the particulars of the image protection used, it may not be possible for a third-party, whether responsible industry participants, open source developers or others, to create a fix for security holes in the protected section without collaboration from the manufacturer.

For example, a manufacturer could choose to not update devices even though the device is still in use by owners. Android phone manufacturers do not necessarily provide software updates, including for security updates, to customers in a timely fashion. A recent study of this issue found that nearly 80% of Android phones were running a version of Android with known security vulnerabilities. [17]

In other situations, a manufacturer could choose to only create updates that also include installation of what to the owner is an unwanted or unsafe feature or removal of a desired feature. Or a manufacturer may no longer be able to provide updates, such as when a manufacturer goes out of business.

8 Equipment Categories

Like most other systems topics, there is no "one-size-fits-all" technical solution that is ideal for all systems. This section discusses some ways of categorizing low-end vs. high-end systems.

One such categorization is related to professional/enterprise systems vs. consumer/entry-level systems. Since the risk of tampering is often directly related to the equipment control or access (and the size of the user base) of a device, such categorizations can be important.

8.1 Professional / Enterprise Systems

As in other areas of FCC regulations, professional use of equipment can be important, and may have different requirements than consumer-grade equipment. One such example can be found in the area of FCC RF exposure limits, where professional/trained users are allowed to tolerate higher RF exposure (for restricted use classes of equipment). Similar restrictions might be applied in terms of target RF parameter vulnerability when control of such equipment can be assumed to be within the hands of a limited set of (accountable) trained professionals. In this higher-end category, control and configuration of devices would have to be limited to a small, trained base of users, inherently reducing the risk of tampering. This may apply to professional or enterprise-based radio systems or other systems where control of hardware is other restricted to professional operators (e.g., in cable or cellular systems). In addition, the tracking of interference or other RF parameter mis-configuration issues is similarly improved for this group of equipment, since the number of responsible parties is limited.

[17] http://androidvulnerabilities.org/#vulnerabilities

8.2 Consumer / Entry-Level Systems

In this lower-end device category, the equipment user base may be much larger and unrestricted in nature. Due to these characteristics, such equipment may be more vulnerable to user alterations of RF parameters. In addition, it may be more challenging to identify the operators of such equipment, since the user base is much more diverse and uncontrolled. Several effective method have been described above (in Section 6) to limit the vulnerability of such systems to alterations.

9 Critical Issues

All protection methods have drawbacks and vulnerabilities. This section discusses issues that may arise when considering these potential solutions.

9.1 Potential Weaknesses in Specific Methods

Here we consider methods listed in Section 6.2 and highlight potential weaknesses to these approaches.

9.1.1 Image blocking

If the image is blocked to prevent any change post-sale, then important downstream fixes (for example, to patch bugs or security holes) cannot be made. If image changes by the manufacturer are, then the method used to provide manufacturer access can be compromised, for example if lock-down keys are released by accident or theft.

9.1.2 Partitioning

Partitioning assumes that all the vulnerable parameters, and only the vulnerable parameters, are sectioned into unchangeable code blocks. However, it may transpire that accessible parameters (or some combination of them with inaccessible parameters) can be changed in a way that leads to harmful interference. Conversely, valuable functionality changes could be compromised if a parameter is incorrectly assigned to a protected partition.

9.1.3 Direct image management

This method assumes that a central authority directly manages firmware, presumably via some authentication and authorization mechanism. That mechanism can be compromised by accidental, intentional or criminal disclosure of the security credentials.

9.1.4 Code signing

In the outright blocking approach, the manufacturer prevents the use of an image that is not signed by the manufacturer or other authorized developer. This is compromised if a third party obtains supposedly secure keys with which it can sign images so that they appear to have been authorized. It may also be possible to deceive the software in the radio into accepting improperly signed code.

9.1.5 Virtualization

The security of code in different virtual machines relies on a secure implementation. If there are leaks between virtual machines, the solution can be circumvented. It may also be possible, depending on implementation, to modify the virtual machine running the wireless radio stack.

9.2 Geolocation

One key issue is that there is no generic solution for a device to securely identify its location and therefore its regulatory domain.

A 3GPP network's country-specific information can be used, assuming the device includes a 3GPP-capable modem, provisioning and has access to the network.

A device can also use IEEE 802.11d information from access points, especially multiple ones. A device with a global navigation satellite system (GNSS) generally knows with some confidence where it is located, except when the satellite signal is unavailable (blocked or jammed) or spoofed.

These approaches are discussed in the FCC's guidance document KDB 594280[18]:

> (1) Global Navigation Satellite System (GNSS) sensors in the device, or
>
> (2) Mobile Country Code (MCC), or MCC with a Mobile Network code (MNC), received from a CMRS carrier and received directly by a receiver on the device, or
>
> (3) Country information derived from multiple adjacent access points (for example using IEEE Std 802.11d provisions) may be permitted on case-by-case basis, or
>
> (4) Other suitable geo-location data based on IP addresses or other reliable source.

For SCRs with mobile broadband capability or satellite receivers, #1 and #2 are feasible and coverage is quite high in the U.S. #3 requires one or more Access Points in range.

Note that government encouraged geolocation may make some privacy conscious users leery.

No one of these solutions is entirely perfect, so this item remains a key issue.

9.3 User-configured Domain

Another key issue is user-configured country or domain.

If the user is expected to or allowed to set the country or operation as a part of the initial device configuration, there are opportunities for inadvertent or intentional misconfiguration.

However, in a global trade environment, it is difficult to envision reliable provisioning based on manufacturer factory settings.

---

[18] Federal Communications Commission, "Guidance on Software or Network Configuration of Non-SDR Devices to Ensure Compliance", August 14 2014, https://apps.fcc.gov/kdb/GetAttachment.html?id=5NjjaXsjV97%2BhlMWvZ1QRw%3D%3D&desc=594280%20D01%20Configuration%20Control%20v02r01&tracking_number=39498

9.4 Migration of Devices

A third key issue is that of devices being moved from regulatory regime to regulatory regime. Owners of these products tend to expect to be able to travel with their wireless devices. If owners purchase a device in the US and take it to another country, they may need to change the radio to the correct regulatory regime in order to operate. If the ability to change regulatory regime is restricted, US travelers may be forced to not operate their device in some countries or, more likely, travelers will operate in a manner which causes interference. This issue is closely related to the above issue in Section 9.3.

10 Other Considerations

The material in this section was originally prompted by certain questions posed by FCC staff during the study period.

10.1 Existing product flexibility

Why don't consumer RF devices (i.e., SCRs) have the flexibility to allow 3rd party software upgrades while maintaining compliance related capabilities?

The following are samples of responses from the manufacturing community:

- A primary reason is simply lack of a reason in the past for such flexibility. Software stacks for IEEE 802.x-compatible digital communications devices are typically designed monolithically, in that replacing one capability requires replacing everything else at the same time. (There are some examples of non-802.x stacks that are layered and can be modified.) This lack of flexibility is primarily due to the expectation that such flexibility would not be required. In other words, the product planning and architectural decisions did not see a market need or technical requirement for such flexibility in the past.
- Another reason is that some CPE products are integrated with cable modems and do not lend themselves well to a 3rd-party implementation as a standalone device might. For the lion's share of CPE devices sold at retail or via cable operators/ISPs, these devices are too integrated to allow for 3rd-party WRT-style software. There are software protections around secure boot, etc., within the DOCSIS specification, so it is quite difficult to add in a 3rd party piece of software.
- Another reason has to do with quality and service related issues. Changes to the operational software in some cases may void the warranty or service contract with the product.
- Some current devices are already protected by digital signing that protects the device from any kind of post-distribution modification.
- TPM (Trusted Platform Module) technology has been used to protect systems when there is a need.
- One respondent's company has experience with customers using high-focus beams as a "security" mechanism, but it isn't true security (as it only requires more proximity for successful 'sniffing').
- Note that features such as are considered here have cost implications and this point will always be part of the manufacturer's product planning.
- Ultimately—this flexibility is not in the product requirements from customers (either cable companies for CPE devices, or consumers). Priorities lately have been on reducing power (EnergyStar).

10.2 Emulating Mobile OS Upgrade Strategies

Is there a model similar to that of the mobile OS (Android, iOS, Windows) that could allow freedom for apps but protecting RF low level functions?

The following are samples of responses from the manufacturing community:

- Yes. For example, a TEE trusted application can be signed, and paired with other applications for other functions. There can be secure vs. non-secure contexts. The operator (carrier) locking in cellphones is an example.
- Another example is that of separate keys.
- Note that Apple recently claimed that they can store information securely (e.g. FBI controversy), although it was feasible to hack (e.g. Israeli company contracting to FBI).
- The downloadable Apps do not change the operating parameters of the radio. The frequency and transmit power remains the same. Someone would also need to show a real need for it.
- See also https://imgtec.com/blog/mips-processors/open-source-virtualization-better-security-wireless-routers.

10.3 Authorizations

Can only authorized users modify compliance related parameters and 3rd party users modify unrelated functions, and can authorization levels be reliably controlled?

Yes, in theory. There are a number of examples of level-based authorization schemes, such as in Unix/Windows/iOS. However, this is a possible feature that must be added to the rest of the solution to the issues with regard to software security. A manufacturer would need to establish a full solution for the single case of "only the manufacturer can modify the device" first before adding "3rd party authorization" as an option.

11 Review of Prior TAC Work

The 2015 FCC TAC efforts around IoT security revealed both challenges facing the rapidly expanding IoT industry, and state of the art security best practices. These takeaways are relevant to this year's SCR efforts since many SCRs can be informally classified as IoT devices.

In general, the IoT investigation revealed industry gaps around the lack of cybersecurity expertise by traditional device manufacturers, long development cycles resulting in long lived vulnerabilities, lack of physical security in certain deployments, and inadequate protection around firmware updates.

Several leading best practices efforts around IoT security emerged during the 2015 investigations. The main sources of discovered best practices came from the CTA, CSA, NIST, FTC, DHS, and OWASP. Some highlights from the best practices which are relevant to the SCR activities include:

- Applying a Secure Systems Engineering approach to architecting and deploying a new IoT System.
- Implementing layered security protections to defend IoT assets.
- Implementing data protection best practices to protect sensitive information.
- Limiting permissions for configuration and SW upgrades
- Making default settings more secure
- Preventing unauthenticated code for critical functions from executing
- Creating more granular permissions to control what device functions end users can modify

- Securing the firmware update process to discourage reverse engineering and to protect against execution of unauthorized code
- Providing means for timely updates of security patches
- Securing web interfaces and APIs (e.g. proper input validation)
- Where applicable, leveraging security analytics features as part of the device management strategy

For reference, the IoT security document in its entirety can be found on the FCC TAC 2015 website: https://transition.fcc.gov/oet/tac/tacdocs/reports/2015/FCC-TAC-Cyber-IoT-White-Paper-Rel1.1-2015.pdf.

Appendix A: Supplemental Information

This section captures additional material assembled during this project that did not specifically apply to the project goal or scope, but that may still be useful or informative.

11.1 DFS In Action

An example of DFS is presented in Figure 3 and Figure 4, showing a “before” and “after” view of DFS in operation to back off Wi-Fi transmission when radar signal energy arises. The Y axis is time and the X axis is frequency centering on 5.805 GHz Wi-Fi channel 161 on U-NII.

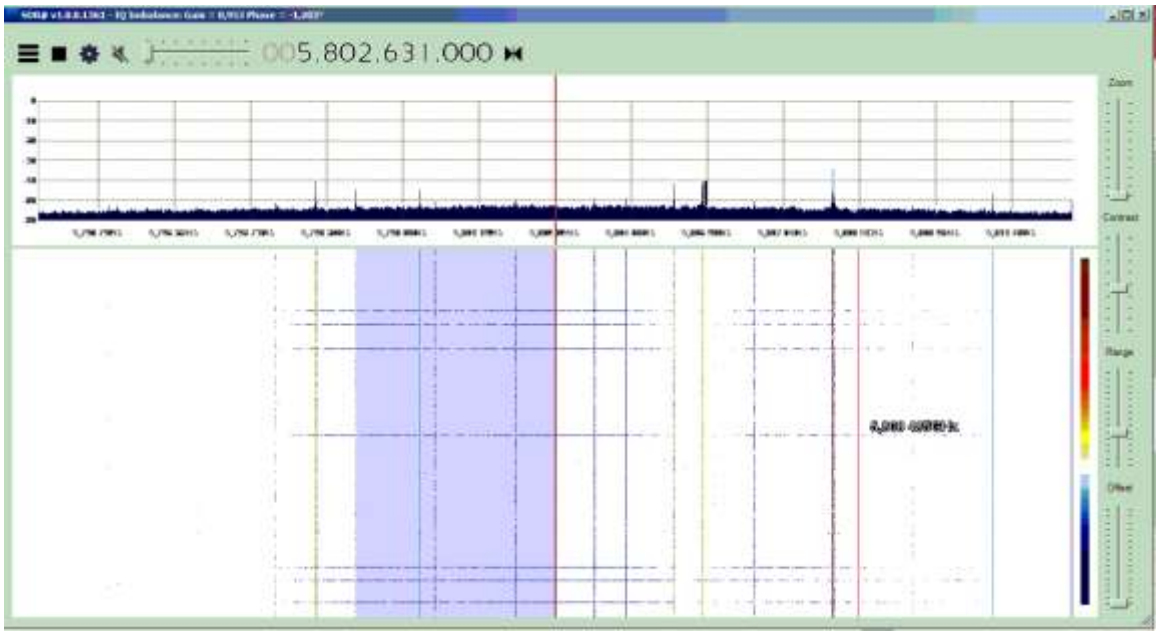

Figure 3: Spectrum without radar signal energy present.

In Figure 3, the radar device is off; this image shows a baseline “before” situation for the band. Pale blue horizontal lines show 40 MHz wide Wi-Fi transmissions on channel 161 (center frequency 5.805 GHz) that are characteristic of a wide band emission technology.

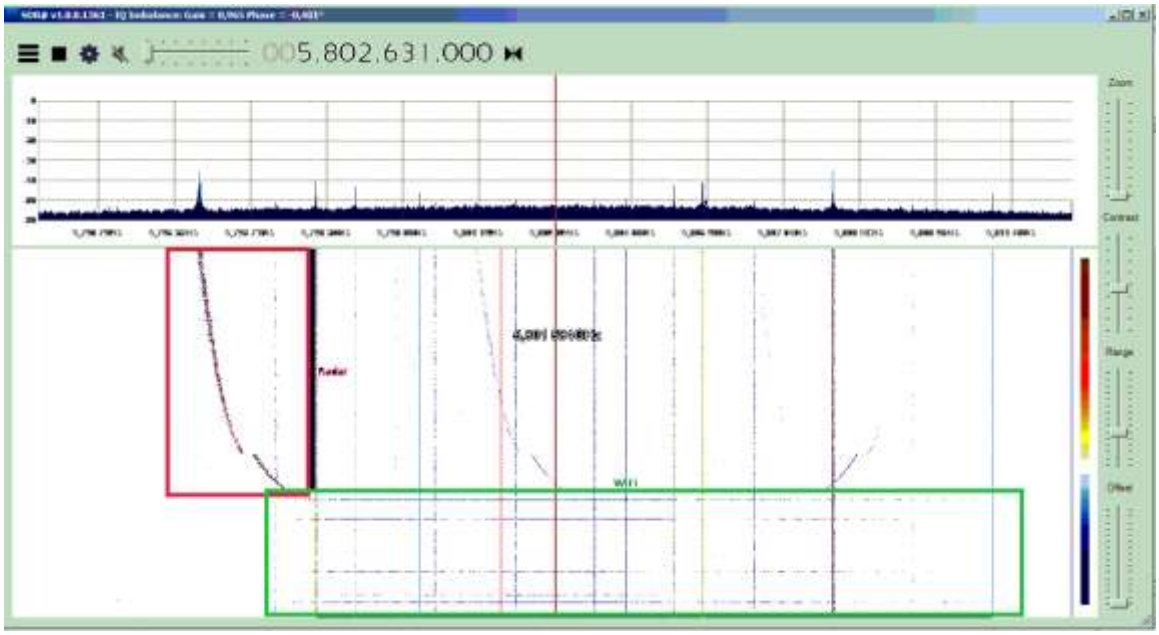

Figure 4: Spectrum with radar emissions present (red box) and Wi-Fi signal highlighted (green box).

In Figure 4, the radar device is on. The bending line in the red box is the radar emission and the horizontal lines are again the 40Mhz wide Wi-Fi transmissions. The green box highlights the much lower level of these transmissions. Notice that the Wi-Fi abruptly stops when radar is transmitting—this is the function of the DFS algorithm.

11.2 Potential Legal Implications of Digital Code Signing Under Open Source Licenses

The latest version (version 3) of the General Public License (GPL[19]) and its related licenses (LGPL[20] and AGPL[21]) may have impact for manufacturers with regard to this topic. Each of those licenses appears to require that the user be allowed to install an identical or modified version of software under those licenses. Importantly, there also appears to be a requirement that the modified object code be able to function despite being modified. Such a requirement would imply that most of the Methods discussed in section 6.2 would not be in compliance with the license.

---

[19] https://www.gnu.org/licenses/gpl-3.0.en.html

[20] https://www.gnu.org/licenses/lgpl-3.0.en.html

[21] https://www.gnu.org/licenses/agpl.html

The more common version 2 of the GPL (used by Linux, OpenWrt, and others) has a similar requirement that a user be provided with the instructions and source for rebuilding and installing the software. While GPLv3's requirements are more extensive, GPLv2 does require that redistributors include “scripts used to control compilation and installation of the executable”.[22]

Note that this is not a definitive legal reading, only a suggestion of a way to consider the matter.

[22] https://www.gnu.org/licenses/old-licenses/gpl-2.0.html