TRUSTED COMPUTING TECHNOLOGIES AND THEIR USE IN THE PROVISION OF HIGH ASSURANCE SDR PLATFORMS

Eimear Gallery (Royal Holloway, University of London, Egham, Surrey, TW20 0EX, UK; e.m.gallery@rhul.ac.uk); and Chris Mitchell (Royal Holloway, University of London, Egham, Surrey, TW20 0EX, UK; c.mitchell@rhul.ac.uk).

ABSTRACT

This paper introduces the concept of trusted computing, and highlights the ways in which it may be leveraged to enable the provision of high assurance Software Defined Radio (SDR) platforms.

1. INTRODUCTION

A software defined radio is a communications device "whose operational modes and parameters can be changed or augmented, post manufacturing via software" [1]. This implies that the device can be reconfigured to communicate using multiple frequency bands and protocols, or upgraded in a low cost and efficient manner. While the concept of a reconfigurable air interface holds considerable promise, SDR will only be accepted if the security threats pertaining to the secure download and execution of reconfiguration software can be addressed.

In this paper the concept of trusted computing is initially explored. An overview of the trusted computing industry standard specifications is presented, in conjunction with a synopsis of the most recent developments in trusted computing technologies. Following this, we highlight the threats which may impact upon an SDR device, and analyze those threats which may be addressed through the deployment of trusted computing functionality.

2. TRUSTED COMPUTING

In the context of trusted computing, a platform is trusted if it "behaves in an expected manner for an intended purpose" [2]. This does not necessarily imply, however, that a Trusted Platform (TP) is a secure platform. For example, if an entity can determine that a platform is infected with a virus, whose effects are known, the platform can be trusted by that entity to behave in an expected but malicious manner [3].

In order to implement a platform of this nature, a trusted component, which is usually in the form of built-in hardware, is integrated into a computing platform [4]. This trusted component is then used to create a foundation of trust for software processes running on the platform [4].

It is said that "trusted platforms were so-called because they provide a technological implementation and interpretation of the factors that permit us, in everyday life, to trust others" [5], i.e.

- Either first hand experience of consistent behavior, or trust in someone who vouches for consistent behavior;
- Unambiguous identification; and
- Unhindered operation.

We examine this statement in relation to the 'trusted component' upon which a trusted platform is constructed, and the software processes running on the platform, for which it provides a 'foundation of trust'.

3. THE TRUSTED COMPUTING GROUP

The Trusted Computing Group (TCG)[1] is an industry forum which is developing standards for trusted computing platforms. Trusted computing, as currently defined by the TCG, is built upon four fundamental concepts: integrity measurement, authenticated boot, platform attestation, and sealing.

3.1. Integrity Measurement

An integrity measurement is defined in [6] as the cryptographic digest or hash of a platform component. For example, an integrity measurement of a program can be calculated by computing the cryptographic digest or hash of its instruction sequence, its initial state (i.e. the executable file) and its input.

3.2. Authenticated Boot

An authenticated boot process represents the process by which a platform's configuration or state is reliably measured, and the resulting measurement is reliably stored. During this process, the integrity of a pre-defined set of platform components is measured, as defined in section 3.1, in a particular order. These measurements are condensed to form a set of integrity metrics which can then be stored in a tamper-resistant log. Condensing enables an unbounded number of platform component measurements to be stored. If each measurement was stored separately it would be

[1] www.trustedcomputinggroup.org

difficult to decide on an upper bound on the size of memory required to store them [4]. A record of the platform components which have been measured is also stored on the platform.

3.3. Attestation

Attestation is the process by which a platform can reliably report evidence of its identity and its current state (i.e. the integrity metrics which have been stored to the tamper resistant log, and the record of the platform components which have been measured, as described in section 3.2).

3.4. Sealing

Sealing represents the process of associating data with a set of integrity metrics representing a particular platform configuration, and encrypting it. The data can only be decrypted and released when the state of platform is the same as that indicated by the integrity metrics sealed with the data.

4. THE TRUSTED PLATFORM SUBSYSTEM

As described in section 2, in order to provide the services described above, a 'trusted component' must be integrated into a platform. This trusted component is comprised of three so-called 'roots of trust' – the Root of Trust for Measurement (RTM), the Root of Trust for Storage (RTS), and the Root of Trust for Reporting (RTR). A root of trust is defined as a component that must be unconditionally trusted for the platform to be trusted [2].

4.1. The RTM

The RTM is an engine capable of measuring at least one platform component, and hence providing an integrity measurement, as described in section 3.1. The RTM is typically implemented as the normal platform engine controlled by a particular instruction set (the so-called 'Core Root of Trust for Measurement' (CRTM)). On a PC, the CRTM may be contained within the BIOS or the BIOS Boot Block (BBB), and is executed by the platform when it is acting as the RTM. It is required by the TCG that the CRTM is protected against software attack: the CRTM must be immutable, as defined by the TCG, meaning that its replacement or modification must be under the control of the host platform manufacturer alone [7]. It is also preferably that the CRTM be physically tamper-evident [4].

4.2. The RTS and RTR

The RTS is a collection of capabilities which must be trusted if storage of data inside a platform is to be trusted [4]. The RTS is capable of maintaining an accurate summary of integrity measurements made by the RTM, i.e. condensing integrity measurements and storing the resulting integrity metrics, as described in section 3.2. The RTS also provides integrity and confidentiality protection to data and enables sealing. In conjunction with the RTM and RTS, an additional root of trust is necessary for the implementation of platform attestation, namely the RTR. The RTR is a collection of capabilities that must be trusted if reports of integrity metrics are to be trusted (platform attestation) [4].

The RTR and the RTS constitute the minimum functionality that should be provided by a Trusted Platform Module (TPM) [9-11]. A TPM is generally implemented as a chip which must be uniquely bound to a platform. In order to support RTS and RTR functionality, a TPM incorporates various functional components such as: I/O; non-volatile and volatile memory; a minimum of 16 Platform Configuration Registers (PCRs), which are used by the RTS to store the platform's integrity metrics; a random number generator; a hash engine; key generation capabilities; an asymmetric encryption and digital signature engine; and an execution engine. The TPM must be protected completely against software attack, i.e. the RTS and RTR (i.e. the TPM) must be immutable, which implies that the replacement or modification of RTS and RTR code must be under the control of the TPM manufacturer alone. The TPM is required to provide a limited degree of protection against physical attack (tamper-evidence) [4].

5. TP SUBSYSTEM FUNCTIONALITY

We now examine how the services described in section 3 are provided by the RTM, RTS and RTR.

5.1. The Authenticated Boot Process

An authenticated boot process enables the state of a platform to be measured and recorded so that it can be reported to a challenger of the platform, as described in section 3.3. A simplified authenticated boot process may proceed as follows, where we assume that the CRTM is part of the BBB. The CRTM measures itself and the rest of the BIOS (i.e. the POST BIOS). The computed measurements are then passed to the RTS which condenses them and stores the resulting integrity metric to the first of the 16 PCRs (PCR-0). Control is then passed to the POST BIOS which measures the host platform configuration, the option ROM code and configuration, and the Operating System (OS) loader. The computed measurements are passed to the RTS, which condenses them and stores the resulting integrity metrics to PCRs 1-5. Control is then passed to the OS loader which measures the OS. This process of measuring, condensing, storing, and handing-off, continues until the platform's configuration has been measured and stored. The

exact measurement process is dependent on the platform; for example, the TCG specifications detail authenticated boot processes for a platform which has a 32-bit PC architecture BIOS, [7] and for an Extensible Firmware Interface platform [8].

5.2. The TPM Protected Storage Functionality

The TPM protected storage functionality, which incorporates its sealing capability, was designed so that an unbounded number of secrets/data could be confidentiality and integrity protected on a TP. Asymmetric cryptography is used to confidentiality-protect data.

Protected storage also provides implicit integrity protection of data objects. Data can be associated with a string of 20 bytes of authorization data before it is encrypted. If data decryption is requested, the authorization data must be submitted to the TPM. The submitted data is compared to the authorization data in the decrypted string, and the decrypted data object is only released if the values match. If the encrypted object has been tampered with, after decryption the authorization data will most likely have been corrupted (because of the method of encryption employed) and access will not be granted even to an entity which has submitted the correct authorization data. Functionality to control how data is used on its release, or to protect data from deletion, is not provided.

The TPM protected storage functionality incorporates an asymmetric key generation capability. This capability enables the generation of key pairs, where the private keys from these pairs can only be used on the TPM on which they were generated, and/or can only be used if the TPM host platform is in a specified state. These private keys are never exposed outside the TPM in the clear. The TPM enables the encryption of keys or data external to the TPM so that they can only be decrypted on a particular TPM; it also enables the encryption of keys or data external to the TPM so that they can only be decrypted by a particular TPM when the TPM host platform is in a particular state. Finally, sealing is provided, i.e. the association of data with a particular platform configuration (i.e. a set of integrity metrics) and its encryption by a particular TPM. The sealed data can only be decrypted by the same TPM and will only be released if the TPM host platform is in the specified state.

5.2. Platform Attestation

Platform attestation enables a TPM to reliably report information about its identity and the current state of the TPM host platform. Each TPM is associated with a unique asymmetric key pair called an endorsement key pair, and a set of credentials. A trusted platform management entity (which is generally the TPM manufacturer) attests to the fact that the TPM is indeed genuine by digitally signing an endorsement credential, which binds the public endorsement key to a TPM description. Conformance credentials may be issued by laboratories: these attest that a particular type of TPM, associated components such as a CRTM, the connection of a CRTM to a motherboard, and the connection of a TPM to a motherboard, conform to TCG specifications. A platform entity (usually the platform manufacturer) offers assurance in the form of a platform credential that a particular platform is an instantiation of a TP. In order to create a platform credential, a platform entity must examine the endorsement credential of the TPM, the conformance credentials relevant to the TP, and the platform to be certified.

Since a TPM can be uniquely identified by its endorsement key pair, this key pair is not routinely used by a platform, ensuring that the activities of a TP cannot be tracked. Instead, an arbitrary number of pseudonyms in the form of attestation identity key (AIK) pairs can be generated by a TPM and associated with a TP. Privacy-Certification Authorities (P-CAs) enable attestation identity public keys to be associated with TPs through the generation of AIK credentials. Once a platform has requested an AIK credential from a specified P-CA, the P-CA verifies all the TP credentials, as described above, to ensure that the TP is genuine, and then creates (signs) an AIK credential which binds the public AIK to a generic description of the TP. The private AIK is used by the TPM during platform attestation.

Platform attestation is a process by which a platform signs a nonce (sent by a challenger of the platform) in conjunction with integrity metrics reflecting the current state of the platform, using one of its private AIKs. This signed bundle is returned to the challenger with the record of the platform components which are reflected in the integrity metrics, together with the appropriate AIK credential. The challenger then uses this information to determine whether it is:

1. Safe to trust the TP from which the statement has originated;
2. Safe to trust (part of) the software environment running on the platform.

6. TRUST

We now evaluate the factors which make it safe to trust a TP and (part of) the software environment running on the platform. It is safe for the challenger to trust a TP on validation of two elements.

- An AIK credential, in which a trusted entity vouches for the consistent behavior of the TP, i.e. that the CRTM and TPM comply with TCG specifications. If the CRTM and TPM comply with the TCG specifications, this also implies that both the CRTM and TPM are immutable and tamper-

evident and can therefore be trusted to operate unhindered.
- The signature of the TPM generated using its private AIK, which serves to unambiguously identify a TP.

It is only safe to trust (part of) the software environment of the platform which has been attested to after examining two elements.

- The signed integrity metrics reported by the TP, which enable the challenger to verify (part of) the platform's software environment.
- The expected integrity measurements of each platform component, which can be extracted from the component's validation certificate. A validation certificate gives the expected integrity measurement of a component if it is behaving as intended. These measurements are then condensed and compared to the integrity metrics attested to by the TP and received by the challenger. The reported identity of (part of) the TP's software environment can thence be validated. After validation, (part of) the TP's software environment can be unambiguously identified.

In order to ensure software can operate unhindered, the definition of what constitutes trusted computing functionality, as defined by the TCG, must be revised and extended to incorporate concepts such as software isolation or protected software execution.

7. ISOLATED EXECUTION ENVIRONMENTS

Isolation enables the unhindered execution of software [5]. In addition to the services provided by the TCG and described above, an isolated execution environment should provide the following services to hosted software [6].

- Protection from external interference.
- Observation of the computations and data of a program running within an isolated environment only via controlled inter-process communication.
- Secure communication between programs running in independent execution environments.
- A trusted channel between an input/output device and a program running in an isolated environment.

In order to provide software isolation and the services described, an isolation layer or virtual machine monitor can be deployed on the platform. Such a mechanism is not defined by the TCG. An isolation layer may be implemented using a number of approaches, as described below. Many of these approaches, however, have associated difficulties with respect to assurance, device support, legacy OS compatibility and performance.

7.1. OS-Hosted Virtual Machine Monitor

In the case of an OS-hosted virtual machine monitor, such as VMWare workstation, all guest OSs executing in VMs utilize the host OS device drivers. While this implies that every guest can utilize drivers developed for the host machine, it also means that the isolation layer essentially incorporates the VMM and the host OS, making assurance problematic [6, 13].

7.2. Standalone VMM

In a standalone VMM, such as Terra [14], all devices are virtualized or emulated by the VMM. This means that the VMM must contain a virtual device driver for every supported device. As the set of devices in consumer systems is often large, and as many virtual drivers are complex, the size of the VMM quickly grows at the cost of assurance.

A standalone VMM exposes the original hardware interface to its guests. While this implies that legacy OSs can be supported, it also means that the VMM size is increased due to the complexity involved in virtualizing the x86 CPU instruction set [6].

7.3. Para-Virtualization

Isolation layers using para-virtualization techniques, such as XEN [15], have been designed for efficiency, and try to alleviate the complexity introduced when devices are virtualized. Two common approaches used in order to para-virtualize I/O are as follows [13].

In the first case, an I/O-type-specific API for each device is integrated into the VMM, in conjunction with the device drivers [13]. This approach requires a guest OS to incorporate para-virtualized drivers which enable communication with the VMM APIs rather than the hardware device interfaces. While this gives performance gains over full virtualization, the guest OS must be modified to communicate with the I/O-type-specific APIs.

Alternatively, a service OS, which incorporates the VMM APIs and the device drivers, may execute in parallel to guest OSs, which are modified to incorporate para-virtualized drivers [13]. To enable this approach, devices are exported to the service OS. While this approach means that device drivers do not have to be implemented within the isolation layer, the isolation layer may become open to attack from a guest in control of a direct memory access device which is, by default, given unrestricted access to the full physical address space of the machine.

7.4. An Isolation Layer with Hardware Support

The isolation layer described as part of the NGSCB [6, 16] was designed to take advantage of CPU and chipset extensions in the next generation of hardware. Such extensions are being provided, for example, by Intel's

LaGrande [3]. The isolation kernel has been designed to execute in a CPU mode more privileged than the existing ring 0, effectively in ring -1, which will be introduced in forthcoming versions of the x86 processors. This enables the isolation layer to operate in ring -1 and all guest OSs to execute in ring 0. Thus, complexity problems which arise when virtualizing the x86 instruction set are avoided [6].

The original hardware interface is exposed to one guest OS [6]. However, rather than necessitating the virtualization of all devices, as a VMM does, devices are exported to guest OSs which contain drivers for the devices they choose to support. Guest operating systems may then efficiently operate directly on the chosen device. This does, however, leave the problem of uncontrolled Direct Memory Access (DMA) devices, which by default have access to all physical memory. In order to prevent DMA devices circumventing virtual memory-based protections provided by the isolation layer, it is necessary for the chipset manufacturers to provide chipset extensions. This enables a DMA policy to be set by the isolation layer which indicates, given the state of the system, if a particular subject (DMA device) has access (read or write) to a specified resource (physical address), [6]. The DMA policy is then read and enforced by hardware, for example the memory controller or bus bridges.

Hardware extensions required in order to facilitate the implementation of the NGSCB isolation layer have been provided as part of Intel's LaGrande [3] and AMD's Presidio initiatives. Both enable the efficient and secure implementation of an isolation layer, as described by Microsoft, through the implementation of CPU and chipset extensions. Both also support the establishment of trusted channels between the input and output devices and programs running within an isolated environment.

8. SOFTWARE DEFINED RADIO

Software defined radio is an important innovation for the communications industry, providing many advantages over a wireless networking infrastructure and terminals that are implemented completely in hardware. Cost reductions may result from the deployment of a generic hardware platform which can be customized using software [17]. The value of terminals is increased as public/private sector radio system sharing becomes possible and as terminals can be upgraded to comply with evolving communications standards. In conjunction with this, SDR enables operation and maintenance cost reductions, as bug fixes may be completed by software download rather than terminal recall.

Re-configurable radios can also be adapted to meet user and/or operator preferences. A terminal can also be reconfigured to efficiently cope with changing network conditions such as utilization, interference or radio channel quality, thereby offering an enhanced user experience [18]. Efficient roaming is also enabled, as air interface and frequency bands can be reconfigured as required.

While there are many advantages associated with the introduction of SDR terminals, if SDR is to be accepted the security threats introduced by reconfigurable terminals must be analyzed, and measures taken to mitigate these threats.

9. SDR THREAT ANALYSIS

The threats which impact upon a reconfigurable SDR device may be categorized as follows:

- Those which impinge on the security of the downloaded reconfiguration software; and
- Those which impinge on host security.

The fundamental threat to the security of the downloaded reconfiguration software is:

- Unauthorized reading of software while in transit between the software provider and the end host, or while in storage or executing on the end host.

This threat may result in an infringement of the intellectual property rights associated with the software. It may also result in unauthorized access to and execution of software.

Fundamental threats to end host security include:

- Malicious or accidental modification or removal of security-critical software while in storage or executing on the end host.
- The download of inappropriate reconfiguration software which does not meet the capability requirements of the SDR device.
- Malicious or accidental modification, addition or removal of downloaded software in development, in transit or while in storage or executing on the end host.

These threats may result in:

- An inoperable device. For example, if a device uses software modulation, an improper change of the modulation format can render the individual device inoperable [19].
- Violation of Radio Frequency (RF) spectrum rights. This may, for example, result in RF interference. If a device can be programmed to transmit on a frequency for which it is not authorized, signals from other nodes, which are authorized to use this frequency, may be jammed [19]. Alternatively, spurious emissions resulting from unauthorized radio spectrum use could violate user safety [20].
- Increased output power. If a device, for example, operates at maximum power, its performance may be increased at the expense of other users in the communications network [19]. This in turn may force other users to use increased power. In this way, the device battery life is severely shortened. If the radiated power is too high, user safety may also be put at risk [20].

- Compromise of user applications and/or data by malicious software.

The threats listed above, and the possible impacts of their exploitation, include those defined by the SDRF security working group in [21].

10. ADDRESSING THREATS TO SDR USING TC

We now investigate some of ways in which trusted computing functionality may be used in order to address the threats outlined above or, failing that, to limit the level to which a threat may be exploited.

10.1. Protecting the Reconfiguration Software

TC mechanisms may be used to confidentiality-protect the reconfiguration software in transit between the software provider and the end host, while in storage or executing on the end host, and to ensure that only the intended recipient device can access the software.

In [22], we describe a software download protocol which leveraged trusted computing functionality, or, more specifically, sealed storage, platform attestation, and isolation techniques. This protocol is now summarized.

Before the required reconfiguration software can be downloaded to a TP, the TPM is used to generate an asymmetric key pair. This key pair is bound to a set of integrity metrics such that the private key can only be utilized by the TPM on which it was generated when the TPM host platform is in the specified state. The public key from this pair, and the integrity metrics with which its private key are associated, are then certified by the TPM using a TP AIK, described in section 5.2, so that the state to which the private key is bound can be shown to the software provider. The certified public key and the corresponding AIK credential are then sent to the software provider.

On receipt of the certified key and the AIK credential, the software provider verifies the TP's AIK credential and the signature of the TPM on the public key and the associated integrity metrics. If these two elements can be verified, and if the software provider considers the platform software state to which the key is bound to be trustworthy, the provider computes a MAC on and encrypts the reconfiguration software, encrypts the symmetric MACing and encryption keys using the public key received from the TP, signs the encrypted symmetric keys using his private signature key, and transmits this data to the TP. The symmetric keys received by the TP, and therefore the reconfiguration software, can only be accessed when the TP is in the state deemed trustworthy by the software provider.

The software provider may require that the integrity metrics to which the private key is bound, represent an isolated execution environment executing on a specified isolation layer, which is turn is supported by a TP which incorporates hardware extensions that enable efficient and secure isolation, as described in section 7. In this way, the confidentiality of the reconfiguration software can be protected in transit between the software provider and the TP, in storage, and while executing on the device. The software provider is also assured that only a specified TP in a particular state can access the software.

Alternatively, if a more traditional mechanism such as SSL/TLS is used in order to provide secure download of the reconfiguration software, TC functionality can be used in order to harden the SSL/TLS implementation. In this case, prior to the completion of any SSL/TLS protocol, the TPM is used in order to generate the client-side (the SDR device) asymmetric key pair for authentication, which is bound to a set of integrity metrics such that the private key can only be utilized by the TPM on which it was generated when the TPM host platform is in the required state. This key is then certified using a TP AIK. Evidence that this SSL/TLS key pair has been generated on, and certified by, a TPM is then provided by a Certification Authority (CA) in an extension of the SDR device's X.509 SSL/TLS certificate.

During an SSL/TLS protocol run between a software provider and the SDR device, the information provided in the extension of the SDR device's X.509 SSL/TLS public key certificate enables a software provider to trust that the SDR device's private SSL/TLS key is held within a TPM, and that the key can only be used when the platform is in a particular state. As above, the software provider may require that the integrity metrics to which the private key is bound, represent an isolated execution environment into which the software will be downloaded and executed. This hardened implementation of SSL/TLS gives the software provider some assurance that the SDR device's SSL/TLS private key is stored securely and cannot been stolen. Evidence of the device's ability to provide an isolated execution environment for the downloaded software can also be demonstrated. This process is described in [23]. The protocol described in [22] has the advantage that less processing is required on the potentially resource-constrained SDR device in order to complete the download.

10.2. Protecting the Host

10.2.1. Security-Critical Software

While the integrity of security-critical software while in storage cannot be ensured using TC functionality, TC mechanisms may be utilized to help detect its malicious or accidental modification or removal.

A secure boot process can be used to ensure that a set of security-critical platform components boot into the required state. Secure boot is not currently enabled by the TCG TPM main specifications. However, much work on secure boot has been conducted independently of the TCG, including by

Tygar and Yee [24], Clark [25], Arbaugh, Farber and Smith [26] and Itoi and Arbaugh [27]. Each of these papers describe a similar process, in which the integrity of a pre-defined set of system components is measured, as described in section 3.1, and these measurements then compared against a set of expected measurements which must be securely stored and accessed by the platform during the boot process. If, at any stage during the boot process, the removal or modification of a platform component is detected, the boot process is aborted. While a secure boot process is not specified in the TPM specification set, the TCG mobile phone working group has recently released a specification for a Mobile TPM which enables a secure boot process [28].

TC functionality also enables the isolation of security-critical software in a secure execution environment so that it cannot be observed or modified when executing by software executing in parallel insecure execution environment.

10.2.2. Reconfiguration (Radio) Software

A capability exchange must be completed by the network and the SDR prior to software download to ensure that the appropriate software entities and parameter sets are selected for a particular SDR device, [29]. The use of platform attestation, as described in section 3.2, could be used to ensure that the reports sent by the device are accurate.

TC cannot prevent denial of service attacks resulting from the removal/deletion of the downloaded radio software, either in development, or in transit between the software provider and the host or in storage on the host. While TC functionality cannot prevent the malicious or accidental modification or addition of downloaded software in development or in transit, in the advent of malicious or buggy software being downloaded to and executed on a device, there are a number of ways in which TC can lessen the impact of the exploitation of this threat.

If the downloaded software is isolated in its own execution environment, as described in section 7, then any malicious behavior can be controlled and its effects limited.

If sealed storage is utilized by the end user to protect their private data (e.g., credit card numbers), then the impact of malicious software may be lessened, as it cannot gain access to security sensitive data which has been protected.

On reconnection to a commercial network, a trusted SDR device could be required to attest to its state so that a decision can be made as to whether the device should be authorized to access the network. Building upon basic attestation, as described in section 3.2, the specifications of the TCG Trusted Network Connect work group describe a process which may be completed in order to ensure that devices connecting to a network are in a trustworthy state. This 3-stage process involves:

- Assessment (is the platform in a trustworthy state?);
- Isolation (of the device if its state is not considered trustworthy by the network); and
- Remediation (where the state of the device can be updated/modified as required) [30].

Process isolation may be utilized in order to ensure that the downloaded radio software can execute unhindered.

11. CONCLUSIONS

In this paper we have introduced the concept of trusted computing. Following a brief overview of the threats introduced by the deployment of re-configurable terminals we highlighted some of ways in which TC functionality may be used in order to mitigate, or lessen the impact of, these threats. While trusted computing is not a panacea to the plethora of security threats pertaining to the secure download and execution of reconfiguration software, it enables us to address a significant number of them, either through threat mitigation or through threat impact reduction.

12. REFERENCES

[1] Software Defined Radio Forum (SDRF), Overview and Definition of Software Download for RF Reconfiguration, SDRF Archived Approved Document, DL-DFN Document SDRF-02-A-0002-V.0.0, 27th August 2002.

[2] Trusted Computing Group (TCG), "TCG Specification Architecture Overview", TCG Specification Version 1.2, The Trusted Computing Group, Portland, Oregon, USA, April 2003.

[3] D. Grawrock, The Intel Safer Computing Initiative Building Blocks for Trusted Computing, Intel Press, Oregon, USA, 2006.

[4] B. Balacheff, L. Chen, S. Pearson, D. Plaquin, and G. Proudler, Trusted Computing Platforms: TCPA Technology in Context, Prentice Hall, Upper Saddle River, New Jersey, USA, 2003.

[5] G.J. Proudler, "Concepts of Trusted Computing", In C.J. Mitchell, editor, Trusted Computing, IEE Professional Applications of Computing Series 6, chapter 2, pages 11-28, The Institute of Electrical Engineers (IEE), London, UK, 2005.

[6] M. Peinado, P. England and Y. Chen, "An Overview of NGSCB", In C.J. Mitchell, editor, Trusted Computing, IEE Professional Applications of Computing Series 6, chapter 4, pages 115-144, The Institute of Electrical Engineers (IEE), London, UK, 2005.

[7] Trusted Computing Group (TCG), "TCG PC Client Specific Implementation Specification for Conventional BIOS - for TPM Family 1.2; Level 2", TCG Specification Version 1.2 Final Revision 1.00, The Trusted Computing Group, Portland, Oregon, USA, July 2005.

[8] Trusted Computing Group (TCG), "TCG EFI Platform - for TPM Family 1.1 or 1.2", TCG Specification Version 1.2 Final Revision 1.00, The Trusted Computing Group, Portland, Oregon, USA, June 2006.

[9] Trusted Computing Group (TCG), TPM Main, Part 1 Design Principles, TCG Specification Version 1.2 Revision 94, The Trusted Computing Group, Portland, Oregon, USA, March 2006.

[10] Trusted Computing Group (TCG), TPM Main, Part 2 TPM Data Structures, TCG Specification Version 1.2 Revision 94, The Trusted Computing Group, Portland, Oregon, USA, March 2006.
[11] Trusted Computing Group (TCG), TPM Main, Part 3 Commands, TCG Specification Version 1.2 Revision 94, The Trusted Computing Group, Portland, Oregon, USA, March 2006.
[12] National Institute of Standards and Technology (NIST), Security Hash Standard, Federal Information Processing Standards Publication FIPS PUB 180-2, The National Institute of Standards and Technology, August 2000.
[13] D. Abramson, J. Jackson, S. Muthrasanallur, G. Neiger, G. Regnier, R. Sankaran, I. Schionas, R. Uhlig, B. Vembu and J. Wiegert, "Intel Virtualization Technology for Directed I/O", Intel Technology Journal, 10(3): pages 179-192, 10th August 2006.
[14] T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum and D. Boneh, "Terra: A Virtual Machine Based Platform for Trusted Computing", In Proceedings of the 19th Symposium on Operating Systems Principles (SOSP 2003), pages 193-206, Bolton Landing, New York, USA, 19-22 October 2003, ACM Press, New York.
[15] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt and A. Warfield, "Xen and the Art of Virtualization", In Proceedings of the 19th ACM Symposium on Operating System Principals (SOSP 2003), pages 164-177, Bolton Landing, New York, USA, 19-22 October 2003, ACM Press, New York.
[16] M. Peinado, Y. Chen, P. England, and J. Manferdelli, "NGSCB: A Trusted Open System", H. Wang, J. Pieprzyk, and V. Varadharajan, editors, In Proceedings of 9th Australasian Conference on Information Security and Privacy, ACISP 2004, volume 3108 of Lecture Notes in Computer Science (LNCS), pages 86-97, Springer-Verlag, Berlin-Heidelberg, Germany, July 2004.
[17] S.M. Blust, Presentation on "Perspective on Software Defined Radio Focusing on Reconfiguration and Radio Software Download", Cingular Wireless, 8th September 2003.
[18] R. Falk, R.K. Atukula and U. Lucking, "Secure Communications in Future Mobile Communication Systems", In Proceedings of the 14th IST Mobile and Wireless Communication Summit, Dresden, Germany, 19-23 June 2005.
[19] R.L. Hill, S. Myagmar and R. Campbell, "Threat Analysis of GNU Software Radio", In Proceedings of the World Wireless Congress (WWC 2005), Palo Alto, California, USA, 24-27 May 2005.
[20] Software Defined Radio Forum (SDRF), Security Considerations for Operational Software Defined Radio Devices in a Commercial Wireless Domain, SDRF Working Document, DL-SIN Document SDRR-04-A-0010-V.0.00, 27th October 2004.
[21] Software Defined Radio Forum (SDRF), Security Functional Requirements for a Software Reconfigurable Communications Device, SDRF Working Document, SDRF-06-W-0002-V0.15, 3rd October 2006.
[22] E. Gallery and A. Tomlinson, "Protection of Downloadable Software on SDR Devices", In Proceedings of the 4th Software Defined Radio Forum Technical Conference (SDR 2005), Orange County, California, USA, 14-18 November 2005, Software Defined Radio Forum (SDRF).
[23] Trusted Computing Group (TCG), Subject Key Attestation Evidence Extension, TCG Specification Version 1.0 Revision 7, The Trusted Computing Group, Portland, Oregon, USA, 16th June 2005.
[24] J. Tygar and B. Yee, "Dyad: A System for Using Physically Secure Co-processors", Technical report CMU-CS-91-140R, Carnigie Mellon University, May 1991.
[25] P.C. Clark and L.J. Hoffman, "BITS: A Smartcard Protected Operating System", Communications of the ACM, 37(11): pages 66-94, November 1994.
[26] W.A. Arbaugh, D.J. Farber and J.M. Smith, "A Secure and Reliable Bootstrap Architecture", In Proceedings of the 1997 IEEE Symposium on Security and Privacy, pages 65-71, Oakland, California, USA, 4-7 May 1997, IEEE Computer Society, IEEE Computer Society Press.
[27] N. Itoi, W.A. Arbaugh, S.J. Pollack, and D.M. Reeves. "Personal Secure Booting", In Proceedings of the 6th Australasian Conference on Information Security and Privacy, volume 2119 of Lecture Notes in Computer Science (LNCS), pages 130-141, Springer-Verlag, London, UK, July 2001.
[28] Trusted Computing Group (TCG), TCG Mobile Trusted Module Specification, TCG Specification Version 0.9 Revision 1, The Trusted Computing Group, Portland, Oregon, USA, 12th September 2006.
[29] Software Defined Radio Forum (SDRF), Requirements for Radio Software Download for RF Reconfiguration, Technical Report SDRF-02-A-0007-V0.0, SDR Forum, November 2002.
[30] Trusted Computing Group (TCG), Trusted Network Connect Architecture for Interoperability, TCG Specification Version 1.1 Revision 2, The Trusted Computing Group, Portland, Oregon, USA, 1st May 2006.