Crimeware and Trusted Computing

Shane Balfe, Eimear Gallery, Chris J. Mitchell and Kenneth G. Paterson
Information Security Group,
Royal Holloway, University of London
{s.balfe,eimear.gallery,c.mitchell,kenny.paterson}@rhul.ac.uk

October 1, 2007

Abstract

Trusted Computing technology has been put forward as a potentially revolutionary addition to the field of information security. In this chapter we examine how Trusted Computing may be used to defend against the ever-growing threat posed by crimeware. We also highlight a counterintuitive but important use of Trusted Computing, as a possible facilitator of cybercrime.

1 Introduction

The anonymous and international nature of the Internet makes cybercrime a potentially low risk, high-return activity. Traditionally, the appropriation of user data through "phishing" attacks has been reliant on social engineering techniques in which a user is tricked into performing an action that results in the revelation of sensitive information. However, as attacks become more sophisticated, social engineering is being superseded by techniques that directly target vulnerabilities in end-user platforms. These vulnerabilities, once exploited, allow crimeware to be surreptitiously installed, leaving the platform prone to data exposure. Crimeware such as keystroke loggers, viruses, worms, rootkits and trojan horses can execute silently in the background and can monitor, log and report keystrokes entered by a user, steal commercially sensitive data or be used in the furtherance of additional criminal activities.

Recent studies that examine the evolution, proliferation and propagation of crimeware illustrate a marked and steady rise in both the number and complexity of applications used in the commission of cybercrime [20, 36]. Crimeware is beginning to combine characteristics of viruses, worms, trojan horses with server and Internet vulnerabilities, fusing numerous methods for compromising end-user systems with multiple means of propagation to other networked machines [36]. Additionally, it is predicted that mobile devices (such as smart phones and PDAs) will increasingly become targets of crimeware in the coming years, especially as organisations begin to allow corporate data to be stored on these

devices [36, 45]. Crimeware technology represents a serious and viable threat to both consumer and corporate data.

As a consequence of the perpetual increases in the volume, complexity and scope of crimeware attacks, the natural question arises, how can users trust the software environments with which they interact? Trusted Computing technology partially addresses this question by providing a means for end-users (and third-parties) to derive increased confidence in the platforms with which they interface, as well as providing standardized mechanisms to protect user data and information from software attack. In this chapter we examine how Trusted Computing technologies can be used to impede the distribution, infection and execution of crimeware applications. In doing so, we note a counterintuitive but important use of Trusted Computing, as a possible facilitator of cybercrime.

This chapter is structured as follows. In section 2, we examine the lifecycle of a crimeware attack. Section 3 provides an overview of Trusted Computing related concepts and examines how Trusted Computing enhanced platforms can be used to impede crimeware at each stage of its lifecycle. This section also examines how Trusted Computing may actually compound the threat posed by crimeware. In section 4, we examine two case studies. The first looks at the use of Trusted Computing in combatting on-line credit card fraud. The second looks at how Trusted Computing can aid rights management for content protection on mobile platforms.

2 Anatomy of an Attack

Cybercrime can broadly be defined as any "crime that is facilitated or committed using a computer, network, or hardware device", where the computer, network or device may be the agent of the crime, the facilitator of the crime, or the target of the crime [19]. More specifically, the Council of Europe's Convention on Cybercrime uses this term to refer to criminal activity ranging from offences against computer data and systems to digital content and copyright infringements [26]. However, irrespective of the actual motivation for such activity, a crimeware attack, or more generally a malware attack, must typically pass through three stages to fulfil its goal [46]. These are *distribution*, *infection* and *execution*.

- **Distribution**: Distribution refers to the means by which malware arrives at a platform. Traditionally, malware has been heavily reliant on social engineering as a means of distribution. Here a user is tricked into downloading malware from a compromised server, opening an email or instant message attachment containing malware, or indeed, installing malware that has been integrated into an apparently useful application. However, such distribution methods are being displaced by methods that directly target and exploit vulnerabilities on a platform.

- **Infection**: Infection is the process by which malware penetrates a platform. Malware may be ephemeral and leave behind no lasting executable,

as is the case with system reconfiguration attacks, such as DNS poisoning. Alternatively, malware may persistently reside in memory or be executed upon loading an infected component. The infection may target user-space objects, as is the case with malicious browser help objects and application programs, or kernel objects, as is the case with device drivers.

In order to disguise an infection, rootkits are sometimes deployed. Basic rootkits simply replace user-level executables with trojanized versions, for example, replacing the Linux Processes Status command with a version that incorrectly reports running processes. However, such attacks are trivially detectable given current anti-virus technology. Unfortunately, more recent rootkits are becoming adept at circumventing anti-viral defenses, such as variants of the FU rootkit [15] or new rootkits that exploit hardware-based virtualization, such as Microsoft's proof of concept rootkit, Subvirt [23]. Subvirt attempts to modify a system's boot sequence so that the legacy Operating System (OS) is loaded into a virtual machine monitor. This allows any system-call made by the OS to be observed and modified by the Subvirt application.

- **Execution**: It is during this stage that the malicious objectives of the malware are revealed. The malware may attempt to gain unauthorized access to information, capture user-entered details or steal proprietary data. For example, the Bankash.G trojan horse attempts to steal user account details such as user names and passwords or credit card information from a compromised computer [34]. This data is collated by the crimeware and transmitted back to the attacker for processing. As a further example, another trojan horse called Archiveus [35] (classified as ransomware) bundles randomly selected files on the platform on which it is executing into a password-protected archive and deletes the original files. The platform user is then requested to purchase any product from a specified site in exchange for the password required to retrieve their files.

3 Combating Crimeware with Trusted Computing

Trusted Computing as discussed here, relates directly to the types of system proposed by the Trusted Computing Group (TCG). Namely, a Trusted System is one that will behave in a particular manner for a specific purpose.

The current documentation from the TCG encompasses a vast set of specifications ranging from Personal Computer (PC) [38] and server systems [37] to specifications for trusted networking [42] and trusted mobile platforms [43]. However, it is the TCG's specifications for microcontroller design that have perhaps become most synonymous with Trusted Computing. The Trusted Platform Module (TPM) specifications [39, 40, 41] form the core of all Trusted Computing implementations. These specifications describe a microcontroller with

cryptographic coprocessor capabilities that provides a platform with the following functionality: A number of special purpose registers for recording platform state; a means of reporting this state to remote entities; secure volatile and non-volatile memory; random number generation; a SHA-1 hashing engine; and asymmetric key generation, encryption and digital signature capabilities. However, the specification set produced by the TCG is by no means the only work on Trusted Computing. Trusted Computing also encompasses new processor designs [21, 2] as well as OS support [32, 33]. For the interested reader, introductory texts on Trusted Computing include [25, 31].

Since its release, Trusted Computing has become synonymous with three fundamental concepts: Integrity measurement and storage, attestation, and protected storage. However, recently the definition of what constitutes Trusted Computing functionality has been revised and extended to incorporate the concepts of secure boot and software isolation. In this section, we consider how these fundamental Trusted Computing concepts can be used to impede the distribution, infection and execution of crimeware.

3.1 Integrity Measurement and Storage

An integrity measurement as defined in [33] is the cryptographic digest or hash of a platform component (i.e., a piece of software executing on the platform). For example, an integrity measurement of a program can be calculated by computing a cryptographic digest of its instruction sequence, its initial state (i.e., the executable file) and its input. An integrity metric is a digest of one or more integrity measurements [31]. Integrity metrics are stored in special purpose registers within the TPM called Platform Configuration Registers (PCRs).

During a platform's boot sequence, the entire platform state can be reliably captured and stored, as shown in figure 1. During this process, the integrity of a pre-defined set of platform components (typically the POST BIOS, option ROMs, the OS loader and OS) are measured and the resulting measurements stored in the TPM's PCRs.

In isolation, integrity measurement and storage functionality do not provide a means of defending against crimeware. They do, however, provide the foundation for a number of services useful in combating the distribution, infection and execution of crimeware, as described in sections 3.2–3.5 below.

3.2 Attestation

Platform attestation enables a TPM to reliably report information about the current state of the host platform. On request from a challenger, a Trusted Platform can, using a private attestation key, sign integrity metrics reflecting (all or part of) the platform's software environment. The challenger uses this information to determine whether it is safe to trust the platform from which the statement has originated and (all or part of) the software environment running on the platform. This is achieved by validating the integrity metrics received

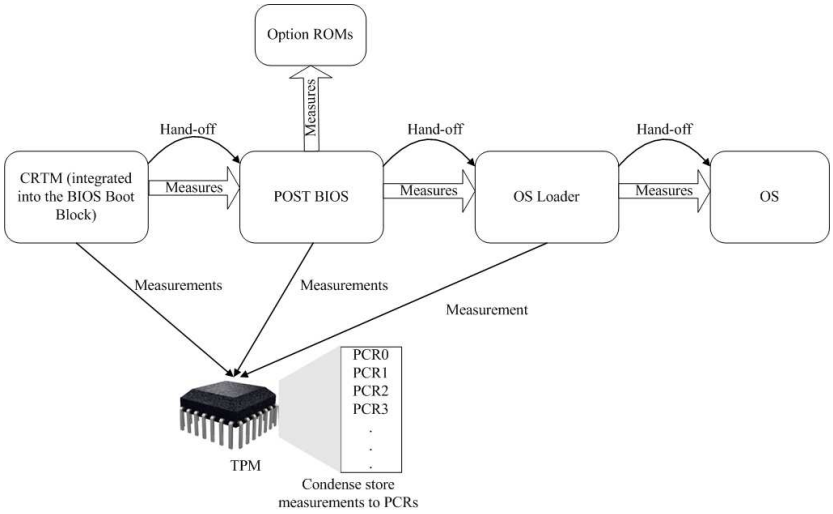

Figure 1: Integrity Measurement and Storage

from the Trusted Platform against software integrity measurements provided by a trusted third party, such as a software vendor.

Attestation provides a powerful technique to combat crimeware distribution and infection. A platform, upon requesting access to a company's intranet, may be required to demonstrate through attestation that it has up-to-date anti-virus software with the latest signature definitions, that its spam filters are operating correctly and that it has installed the latest OS security patches. Similarly, a client could request that a server attests to its operating environment prior to the disclosure of sensitive data.

Attestation features form an important focus of the TCG's Trusted Network Connect (TNC) specifications [42]. TNC offers a way of verifying an endpoint's integrity to ensure that it complies with a particular predefined policy before being granted network access. The process of assaying end-point integrity for compliance with policy occurs in three distinct phases: Assessment, isolation and remediation. The assessment phase primarily involves a platform that requires access to a restricted network, attesting to its current state. A server examines this attestation and compares the platform's integrity metrics to its network access policies. Based on the outcome of this comparison the server will allow access, deny access or place the platform in an quarantined network (isolation). In this isolated network, a platform would typically be able to obtain the requisite integrity-related updates that will allow it to satisfy the server's access policy and be granted access.

3.3 Protected Storage — Binding and Sealing

Protected storage functionality uses asymmetric encryption to protect the confidentiality of data on a TPM host platform. Protected storage also provides implicit integrity protection for TPM objects. Both data and keys can be associated with a string of 20 bytes of authorization data before being encrypted. When decryption is requested, the authorization data must be submitted to the TPM. The submitted authorization data is then compared to the authorization data in the decrypted string, and the decrypted object is only released if the values match.

The notions of *binding* and *sealing* are of fundamental importance to Trusted Computing. Binding refers to the encryption of data with a public key for which the corresponding private key is non-migratable from the recipient's TPM. In this way only the TPM that manages the non-migratable private key will be capable of decrypting the message. Sealing takes binding one step further. Sealing is the process by which sensitive data can be associated with a set of integrity metrics representing a particular platform configuration, and encrypted. The protected data will only be decrypted and released for use by a TPM when the current state of the platform matches the integrity metrics to which the data is sealed.

Using sealed storage, an end-user can protect their private data (e.g., credit card numbers) by making revelation of that data contingent on a platform being in a particular state. For example, a user can seal credit card data to a state that requires a particular banking application to be running on the platform, and nothing more. The presence of crimeware would change the platform state. This feature would therefore ensure that malicious software, such as the Bankash.G Trojan horse, could not gain access to security sensitive data that has been sealed.

In [1], Alsaid and Mitchell attempt to address the problem of a user unknowingly revealing sensitive data to a phishing site using the TPM's protected storage capabilities. They propose SSL client-side authentication to establish a mutually authenticated SSL tunnel over which a username/password can be communicated. In this approach the SSL private key (for which the public key has received certification) is non-migratable from the client's TPM. Consequently, when a user visits a phishing site and is tricked into revealing a username and password, a phisher will not be able to impersonate the user as the phisher will not have access to the private key used to complete client-side SSL authentication. Unfortunately, such an approach does not prevent crimeware resident on the TPM-host platform from capturing the username/password and using the TPM-protected private key to establish illegitimate sessions and fraudulently impersonate a user. This approach could be enhanced by sealing the TPM-protected private key to a trustworthy platform state.

3.4 Secure Boot

A secure boot process extends the integrity measurement and storage functionality described in section 3.1. During a secure boot, a platform's state is reliably captured, compared against measurements indicative of a trustworthy platform state and stored. If a discrepancy is discovered between the computed measurements and the expected measurements then the platform halts the boot process.

Secure boot functionality can detect the malicious or accidental modification or removal of security-critical software at boot time. For example, the Subvirt rootkit, which modifies a system's boot sequence, could be detected by such functionality. In a similar way, secure boot functionality could be used to prevent a maliciously modified server from helping to distribute crimeware; this would reduce the effectiveness of a server modification attack to a denial of service.

Secure boot functionality is not currently described as part of the TPM specifications. However, there is a TCG specification describing how it can be enabled on a trusted mobile platform [43]. Secure boot has also been independently studied by Tygar and Yee [44], Clark [9], Arbaugh, Farber and Smith [3] and Itoi *et al.* [22].

3.5 Hardware-enforced Isolation

Isolation technologies have evolved from OS-hosted virtual machine monitors (VMMs) [47] through stand-alone VMMs [18] to para-virtualization techniques [8]. More recent developments in isolation technology, such as Microsoft's Next Generation Secure Computing Base (NGSCB) [32, 33], incorporate the concept of an isolation layer designed to take advantage of CPU and chipset extensions described in Intel's LaGrande [21] and AMD's Presidio initiatives. An isolated execution environment, independent of how it is implemented, should provide the following services to hosted software [33]:

- No interference: Ensures that the program is free from interference from entities outside its execution space.
- Trusted path: Ensures the presence of a trusted path between a program and an input device.
- Secure inter-process communication: Enables one program to communicate with another, without compromising the confidentiality and integrity of its own memory locations.
- Non-observation: Ensures that an executing process and the memory locations it is working upon are free from observation by other processes.

Hardware-enforced software isolation enables the segregation of security-critical software and data so that they cannot be observed and/or modified

in an unauthorized manner by software executing in parallel execution environments. Additionally, the presence of isolated execution environments can ensure that any infection is contained within the execution environment which the crimeware has infected.

In [16], Gajek *et al.* use isolated execution environments to protect a secure wallet application from any legacy OS or other applications running on the platform. In this approach, user credentials are sealed (as described in section 3.3) to the isolated secure wallet application. Only when the secure wallet is in the pre-requisite state will credentials be unsealed and used to authenticate the user to sensitive services. The authors also suggest the use of visual cues, such as a colored status bar, to enable the user to assess the trustworthiness of the secure wallet application with which they are interacting, an idea originating in [5].

Hardware extensions, as developed as part of Intel's LaGrande [21] and AMD's Presidio initiatives, support the establishment of trusted channels between input and output devices and programs running within isolated execution environments. In this way, user I/O data can be secured in transit to protect it from crimeware, such as keyloggers, which may have infiltrated the platform.

We are yet to see, however, the ubiquitous presence of hardware-enabled trusted channels. In the absence of such support, McCune *et al.* [24] have proposed the use of a trusted mobile device to establish an encrypted and authenticated channel between the user and a TPM host. Their approach, however, only considers the issue of user-level malware and does not effectively address problems relating to kernel-level subversion.

3.6 Trusted Computing — A Panacea?

We have briefly examined a range of Trusted Computing functionality and how it may be used to reduce the impact of crimeware. In reality, Trusted Computing, *as currently deployed*, can do little to protect an end-user platform from crimeware attack. It will be several years before we begin to see the ubiquitous presence of enhanced processors, chipset extensions, BIOS modifications and augmented Operating Systems necessary to implement the fundamental Trusted Computing concepts identified in sections 3.1–3.5. The capabilities offered by currently deployed Trusted Platforms are more akin to those of a high-end smartcard; thus Trusted Computing, as it currently stands, provides a limited, but useful, set of cryptographic functionality. We expect that this will be extended over time to provide a more secure operating environment.

In addition, it is important to note that problems associated with software vulnerabilities will not be ameliorated by the presence of Trusted Computing. Neither is Trusted Computing designed to prevent a platform from being infected. Instead, it provides a set of services that can be used to detect if a platform has been modified from a known "good" state. Unlike the way in which signature-based anti-virus mechanisms operate, a Trusted System has a known "good" configuration and any deviation from this configuration is perceived as a possible breach of security. Such an approach tells a user that something is

awry but not necessarily what.

An abundance of information exists on the potential positive applications of Trusting Computing. However, as the technology becomes more widely deployed, it seems likely that Trusted Computing functionality will be increasingly targeted by crimeware. Proof-of-concept rootkits that exploit Intel's VT-X and AMD's SVM virtulization technologies, cornerstones of Intel's LaGrande and AMD's Presidio initiatives, have already been developed by Dino Dai Zovi (Vitriol) [28] and Joanna Rutkowska (Blue Pill) [27].

In addition, Trusted Computing mechanisms may also be used as a means of enhancing crimeware functionality. For example, Trusted Computing provides trivial means for crimeware to launch denial of service attacks, of various types, against a platform. If a crimeware application can be installed on a platform, thereby altering the system state, then:

- In the case of attestation, access to networked services may be denied, since the platform will (correctly) not be considered trustworthy.
- In the case of sealing, access to data may be denied, since the current state of the platform will not match the integrity metrics to which the data has been sealed.
- In the case of secure boot, system startup may be suspended if the presence of crimeware is detected during the boot sequence.

Finally, ransomware (such as Archieveus, as described in section 2), could abuse the TPM's sealing mechanism to encrypt data to a platform state that is contingent on the malware being present on the platform. This could potentially be used as a method to extort money from the platform owner.

4 Case Studies

In this section we look at two case studies that examine the use of Trusted Computing as a means of protecting sensitive data from malicious applications. In the first case study, we look at the issue of on-line credit card fraud and the potential use of Trusted Computing as a means of protecting credit card transactions. In the second case study, we look at how Trusted Computing can be used to enable a robust implementation of Open Mobile Alliance Digital Rights Management (OMA DRM) v2 on mobile platforms.

4.1 Securing Credit Card Transactions

Over the past ten to fifteen years, the Internet has been transformed from a research-oriented tool into a global platform for electronic commerce. With this transformation, the use of one particular method of Internet-based payment has emerged as the predominant means through which on-line goods are purchased. This method, often referred to as a Card Not Present (CNP) transaction, uses

data from a customer's physical card (typically the Personal Account Number (PAN) and the corresponding Card Security Code (CSC)) to purchase on-line goods and services. Unfortunately, Internet-based CNP transactions provide a particularly attractive target for phishers and crimeware authors, as the ability to provide card details is typically deemed a sufficient form of transaction authorization. Once this data is captured, it allows a fraudster to impersonate a legitimate cardholder and purchase items on-line.

Balfe and Paterson examine how the staged roll-out of Trusted Computing technology, beginning with ubiquitous client-side TPMs [6] and culminating in Trusted Computing with processor, chipset and OS support [7], can be used to enhance the security of Internet-based CNP transactions. In [7], a system is described that makes use of the full spectrum of Trusted Computing technologies to securely emulate point-of-sale Integrated Circuit Cards (ICCs) compliant with the Europay Mastercard and Visa (EMV) specifications [11, 12, 13, 14]. Emulation of EMV-compliant cards confers "tamper resistant" properties that are normally associated with physical EMV card use at point-of-sale terminals making it possible to demonstrate card ownership and authentication.

In order to enroll in the virtual EMV architecture, a customer must formally register as a legitimate cardholder. During the enrollment process, the customer's platform generates an attestation key pair (internal to their TPM) specifying an authorization requirement for the private key usage, as per section 3.3. The customer's card issuer certifies the public component of the customer's newly generated TPM-resident attestation key pair and the customer downloads an e-EMV application to his/her platform. This application replicates the functionality of a standard EMV payment card (that has been personalized to the customer) as well as certain aspects of merchant terminal processing.

During merchant enrollment, the merchant's acquirer certifies the public component of a merchant's TPM-resident attestation key pair and downloads a small application bundle to the merchant server that replicates the merchant terminal commands for interacting with an EMV ICC. This software bundle enables the merchant to communicate with a customer's e-EMV card. In addition, the merchant application bundle implements any additional requirements for payment processing as laid down by the acquirer's Merchant Operator Guidelines (MOGs). The MOG lays out the procedures that should be followed when processing CNP transactions. An example of such a procedure would be a requirement to use an Address Verification Service (AVS) which compares the billing address, as entered by the customer, to that of the card issuer's records.

Prior to transaction initiation, a customer launches his e-EMV application in an isolated execution environment. At this point, the customer and the merchant platform mutually attest to their respective states using their respective private attestation keys, the public keys corresponding to which have been certified during the enrollment process. This provides a guarantee that, at this particular point in time, both the customer's e-EMV application and the merchant's plug-in are operating as intended.

In addition, by using Trusted Computing functionality, PIN authentication and authorization can be made intrinsic to transaction processing, just as with

EMV in PoS transactions. A trusted path between the keyboard and TPM, enabled by chipset extensions (as outlined in section 3.5), can be used to securely transfer authorization data in the form of a PIN or a passphrase from the customer's keyboard to the TPM. A correctly entered PIN/passphrase then assures the TPM of the physical presence of the cardholder and "unlocks" the TPM's private attestation key. The assurance of physical presence can be transferred to the merchant through the TPM's subsequent use of this attestation key to attest to the state of the customer's platform. This combined functionality provides a much stronger form of authentication and authorization than is currently employed for CNP payments. The mechanism used to prevent crimeware from launching a dictionary attack against key authorization data would be TPM vendor specific. However, the TPM specifications [39] detail an example mechanism where a a count of failed authorization attempts is recorded. If this count exceeds a threshold, the TPM is locked and remains non-responsive to further requests for a predetermined time out period.

In conjunction with this, by allowing a customer to inspect a merchant's platform state prior to transaction authorization, a customer will be able to satisfy himself that the merchant will behave in a manner that will protect his sensitive card data. Likewise for a merchant, any divergence from intended operating state (due to unwanted memory resident applications, such as keyloggers) will be detected on verification of a customer's attestation, allowing merchant risk management routines to terminate a transaction. Once mutual attestation is complete, transaction processing can continue as defined in the EMV specifications [11, 12, 13, 14]. This work builds upon the pre-existing EMV infrastructure to provide a secure and extensible architecture for CNP payments.

4.2 Content Protection

Our second case study relates to DRM in a mobile environment. As highlighted in [36, 45], it is predicted that mobile devices (hosting sensitive corporate data) will increasingly become the target of crimeware. Current third generation (3G) mobile telecommunications systems are already capable of delivering a wide range of digital content to subscribers' mobile telephones. As network access becomes more ubiquitous and content becomes more easily accessible, media and data objects are exposed to increased risks of illegal consumption and use. DRM facilitates the safe distribution of various forms of digital content in a wide range of computing environments, and gives assurance to the content providers that their media objects cannot be illegally accessed.

The model under consideration is taken from [29] and is summarized in figure 2. A user requests a media object from a content issuer. The requested content, which is packaged in order to prevent unauthorized access, is then sent to the user's device. The packaging of the content may either be completed by the content issuer or by the content owner, before it is dispatched to the content issuer. The rights object associated with the requested media object is delivered to the user by the rights issuer. In practice, this rights issuer may be the same

entity as the content issuer.

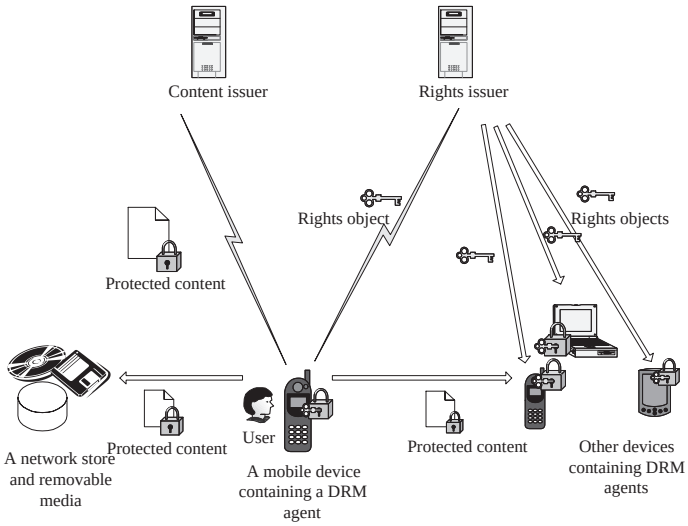

Figure 2: OMA DRM system model

With respect to crimeware, the protection of content can be examined from two contrasting perspectives. From the content provider's perspective, an end-user may deploy crimeware in order to circumvent rights that have been assigned to digital content. From a corporate perspective, DRM solutions can be used to prevent the unauthorized or inadvertent disclosure of corporate data. This is particularly relevant in light of the recent trend towards an increasing reliance on mobile devices for storing corporate data [45].

The OMA was founded in June 2002. Version 1 (v1) of the OMA specifications [48, 49], released in 2004, represents the OMA's initial attempt to define a DRM solution for a mobile environment. Three main goals were specified for OMA DRM v1. The solution was required to be timely, easy to implement and inexpensive to deploy. Finally, it was required that the initial OMA DRM solution did not necessitate the roll-out of a costly infrastructure. In the development of OMA DRM v1 a trade-off was made, so that the objectives listed above could be met at the expense of certain security requirements. OMA DRM version 2 (v2) builds upon the version 1 specifications to provide higher security and a more extensive feature set. Devices other than mobile phones are also supported by OMA DRM v2. The OMA DRM v2 specification set defines [29]:

- the format and the protection mechanism for protected content;
- the format and the protection mechanism for rights objects;

- the security model for the management of encryption keys; and
- how protected content and rights objects may be transferred to devices using a range of transport mechanisms.

The term "DRM agent" refers to the DRM functionality of a device responsible for enforcing permissions and constraints associated with protected content. A DRM agent must be trusted with respect to its correct behavior and secure implementation [29]. The OMA DRM v2 specification set defines the protocols (the Rights Object Acquisition Protocol (ROAP) suite), messages and mechanisms necessary to implement a DRM system in a mobile environment [29, 30]. Stipulation of a trust model, within which robustness rules are defined, is one method of specifying how secure a device implementation of a DRM agent must be, and what actions should be taken against a manufacturer that builds devices that are insufficiently robust. It is the responsibility of the Content Management Licensing Administrator for Digital Rights Management (CMLA DRM), or a similar organization, to provide such a model.

In order to comply with the definition of a "robust" OMA DRM v2 implementation, as defined by the CMLA [10], a number of requirements must be met, as summarized below.

- It is required that "an OMA DRM v2 agent can perform self-checking of the integrity of its component parts so that unauthorized modifications will be expected to result in a failure of the implementation to provide the authorized authentication and/or decryption function" [10].
- A robust implementation of OMA DRM v2 must protect the confidentiality and integrity of an OMA DRM v2 agent's private key when loaded, stored or used on a device.
- OMA DRM v2 security-critical data must be either integrity-protected or integrity and confidentiality-protected when loaded, stored or used on a device.
- OMA DRM v2 security-critical data and the OMA DRM v2 agent's private key should only be accessible by authorized entities, namely the correctly functioning OMA DRM v2 agent.
- A robust OMA DRM v2 implementation must incorporate a DRM time-source synchronization mechanism which is reasonably accurate and resistant to malicious modifications by the end user.
- Finally, nonces generated on the OMA DRM v2 device, and used in the ROAP protocols, must be both non-repeating and unpredictable.

As described in [17], Trusted Computing functionality can be used to help meet the CMLA requirements for a robust implementation of OMA DRM v2, therefore making it, and the content it protects, less susceptible to crimeware attacks. While Trusted Computing functionality cannot guarantee the integrity

of the OMA DRM v2 agent in storage, secure boot functionality can be used to help detect its malicious or accidental modification or removal. Security-critical data associated with the OMA DRM v2 agent can also be verified as part of a secure boot process.

Sealing can be used to store data which needs to be confidentiality and/or integrity-protected against crimeware attack. It can also ensure that sensitive data is only accessible by authorized entities when the mobile device is in a predefined state, for example, when a legitimate OMA DRM v2 agent is executing in an isolated execution environment. In addition, the TPM can be used to generate the required OMA DRM v2 agent asymmetric key pair.

Trusted Computing functionality also enables the isolation of security-critical software and data in a secure execution environment so that it cannot be observed and/or modified in an unauthorized manner by software executing in parallel execution environments.

A good quality random number generator is provided by a TPM, enabling the generation of non-repeating unpredictable nonces for use in the ROAP suite protocols, thereby mitigating replay and preplay attacks. The TPM may also be used to provide accurate time source synchronization, as described in [39]. A comprehensive examination of this use case can be found in [17].

5 Conclusions

In this chapter we have explored the use of Trusted Computing technologies as both a defence against, and enabler of, crimeware. We initially described the lifecycle of a crimeware attack and subsequently investigated how the technologies examined could be used to disrupt crimeware distribution, infection and execution. We also investigated how Trusted Computing may potentially be exploited to facilitate cybercrime. We finally considered two use-cases in which this technology may be used in the prevention of crimeware attacks.

References

[1] A. Alsaid and C. J. Mitchell. Preventing Phishing Attacks using Trusted Computing Technology. In *Proceedings of the 6th International Network Conference (INC '06)*, pages 221–228, July 2006.

[2] T. Alves and D. Felton. TrustZone: Integrated Hardware and Software Security — Enabling Trusted Computing in Embedded Systems. White paper, ARM, July 2004. `http://www.arm.com/pdfs/TZ_Whitepaper.pdf`.

[3] B. Arbaugh. Improving the TCPA Specification. *IEEE Computer*, 35(8):77–79, August 2002.

[4] Visa International Service Association. 3-D Secure Protocol Specification: System Overview, May 2003. `http://international.visa.com/fb/paytech/secure/main.jsp`.

[5] B. Balacheff, D. Chan, L. Chen, S. Pearson, and G. Proudler. Securing Intelligent Adjuncts using Trusted Computing Platform Technology. In *Proceedings of the 4th Working Conference on Smartcard Research and Advanced Applications on Smartcard Research and Advanced Applications*, pages 177–195. Kluwer Academic Publishers, Norwell, MA, USA, 2001.

[6] S. Balfe and K.G. Paterson. Augmenting Internet-based Card Not Present Transactions with Trusted Computing: An Analysis. Technical report, RHUL-MA-2006-9, (Department of Mathematics, Royal Holloway, University of London, 2006). http://www.rhul.ac.uk/mathematics/techreports.

[7] S. Balfe and K.G. Paterson. e-EMV: Emulating EMV for Internet payments using Trusted Computing technology. Technical report, RHUL-MA-2006-10, (Department of Mathematics, Royal Holloway, University of London, 2006). http://www.rhul.ac.uk/mathematics/techreports.

[8] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauery, I. Pratt, and A. Warfield. Xen and the art of virtualization.

[9] P.C. Clark and L.J. Hoffman. BITS: A Smartcard Protected Operating System. *Communications of the ACM*, 37:66–94, November 1994.

[10] CMLA. Client Adopter Agreement. Technical Report Revision 1.00-050708, The Content Management License Administrator Limited Liability Company (CMLA, LLC), August 2005.

[11] EMVCo. *Book 1 - Application Independent ICC to Terminal Interface Requirements*, 4.1 edition, May 2004. `http://www.emvco.com`.

[12] EMVCo. *Book 2 - Security and Key Management*, 4.1 edition, May 2004. `http://www.emvco.com`.

[13] EMVCo. *Book 3 - Application Specification*, 4.1 edition, May 2004. `http://www.emvco.com`.

[14] EMVCo. *Book 4 - Cardholder, Attendant, and Acquirer Interface Requirements*, 4.1 edition, June 2004. `http://www.emvco.com`.

[15] Fuzen_op. The FU Rootkit. Available on-line. `http://www.rootkit.com/`.

[16] S. Gajek, A.-R. Sadeghi, C. Stüble, and M. Winandy. Compartmented Security for Browsers — Or How to Thwart a Phisher with Trusted Computing. In *Proceedings of the 2nd International Conference on Availability, Reliability and Security (ARES '07)*, pages 120–127, Los Alamitos, CA, USA, 2007. IEEE Computer Society, Washington, DC, USA.

[17] E. Gallery. *Authorisation Issues for Mobile Code in Mobile Systems*. PhD Thesis, Department of Mathematics, Royal Holloway, University of London, 2007.

[18] T. Garfinkel, M. Rosenblum, and D. Boneh. Flexible OS Support and Applications for Trusted Computing. In *Proceedings of the 9th USENIX Workshop on Hot Topics on Operating Systems (HotOS-IX)*, pages 145–150, Kauai, Hawaii, USA, 18-21 May 2003. USENIX, The Advanced Computing Systems Association.

[19] S. Gordon and R. Ford. On the Definition and Classification of Cybercrime. *Journal in Computer Virology*, 2(1):13–20, July 2006.

[20] Anti-Phishing Working Group. Phishing Activity Trends Report. Available on-line, April 2007. `http://www.antiphishing.org/reports/apwg_report_april_2007.pdf`.

[21] Intel. LaGrande Technology Architectural Overview. Technical Report 252491-001, Intel Corporation, September 2003.

[22] 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 (ACISP '01)*, volume 2119 of *Lecture Notes In Computer Science (LNCS)*, pages 130–141, Sydney, Australia, 11–13 July 2001. Springer–Verlag, London, UK.

[23] S.T. King, P.M. Chen, Y-M Wang, C. Verbowski, H.J. Wang, and J.R. Lorch. Subvirt: Implementing malware with virtual machines. In *SP '06: Proceedings of the 2006 IEEE Symposium on Security and Privacy (S&P'06)*, pages 314–327, Washington, DC, USA, 2006. IEEE Computer Society.

[24] J.M. McCune, A. Perrig, and M.K. Reiter. Bump in the Ether: A Framework for Securing Sensitive User Input. In *Proceedings of the 2006 USENIX Annual Technical Conference*, pages 185–198, June 2006.

[25] C. J. Mitchell, editor. *Trusted Computing*. IEE Professional Applications of Computing Series 6. The Institute of Electrical Engineers (IEE), London, UK, April 2005.

[26] Council of Europe. Convention on Cybercrime. Available on-line, November 2001. `http://conventions.coe.int/Treaty/en/Treaties/Html/185.htm`.

[27] J. Rutkowska. Subverting Vista Kernel For Fun And Profit. Available online, Black Hat USA 2006. `http://www.blackhat.com/presentations/bh-usa-06/BH-US-06-Rutkowska.pdf`.

[28] D. D. Zovi. Hardware Virtualization Based Rootkits. Available online, Black Hat USA 2006. `http://www.blackhat.com/presentations/bh-usa-06/BH-US-06-Zovi.pdf`.

[29] OMA. DRM architecture V2.0. Technical Specification OMA-DRM-ARCH-V2_0-2004071515-C, The Open Mobile Alliance (OMA), July 2004.

[30] OMA. DRM specification V2.0. Technical Specification OMA-DRM-DRM-V2_0-20040716-C, The Open Mobile Alliance (OMA), July 2004.

[31] S. Pearson, editor. *Trusted Computing Platforms: TCPA Technology in Context*. Prentice Hall, Upper Saddle River, New Jersey, USA, 2003.

[32] M. Peinado, Y. Chen, P. England, and J. Manferdelli. NGSCB: A Trusted Open System. In H. Wang, J. Pieprzyk, and V. Varadharajan, editors, *Proceedings of 9th Australasian Conference on Information Security and Privacy, (ACISP '04)*, volume 3108 of *Lecture Notes in Computer Science (LNCS)*, pages 86–97, Sydney, Austrailia, 13–15 July 2004. Springer–Verlag, Belin–Heidelberg, Germany.

[33] 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 7, pages 115–141. The Institute of Electrical Engineers (IEE), London, UK, April 2005.

[34] Symantec. Infostealer.Bankash.G. Available on-line, Febuary 2006. `http://www.symantec.com/security_response/writeup.jsp?docid=2006-010317-5218-99`.

[35] Symantec. Trojan.Archiveus. Available on-line, May 2006. `http://www.symantec.com/security_response/writeup.jsp?docid=2006-050601-0940-99`.

[36] Symantec. Symantec Internet Security Threat Report Volume XI. Available on-line, March 2007. `http://www.symantec.com/enterprise/theme.jsp?themeid=threatreport`.

[37] TCG. TCG Generic Server Specification. TCG Specification Version 1.0 Final, The Trusted Computing Group (TCG), Portland, Oregon, USA, July 2005.

[38] TCG. TCG PC Client Specific Implementation Specification For Conventional BIOS. TCG specification Version 1.2 Final, The Trusted Computing Group (TCG), Portland, Oregon, USA, July 2005.

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

[40] TCG. TPM Main, Part 2: TPM Data Structures. TCG Specification Version 1.2 Revision 94, The Trusted Computing Group (TCG), Portland, Oregon, USA, March 2006.

[41] TCG. TPM Main, Part 3: Commands. TCG Specification Version 1.2 Revision 94, The Trusted Computing Group (TCG), Portland, Oregon, USA, March 2006.

[42] TCG. TNC Architecture for Interoperability. TCG Specification Version 1.2 Revision 4, The Trusted Computing Group (TCG), Portland, Oregon, USA, September 2007.

[43] TCG MPWG. The TCG Mobile Trusted Module Specification. TCG Specification Version 0.9 Revision 1, The Trusted Computing Group (TCG), Portland, Oregon, USA, September 2006.

[44] J. Tygar and B. Yee. Dyad: A System for using Physically Secure Coprocessors. Technical Report CMU-CS-91-140R, Carnigie Mellon University, Pittsburgh, Pennsylvania, USA, May 1991.

[45] Economist Intelligence Unit. Symantec Ensuring Mobile Security (survey of 248 company executives and senior IT employees). Available online, January 2006. http://www.symantec.com/content/en/us/about/media/mobile-security_Full-Report.pdf.

[46] US Department of Homeland Security and SRI International Identity Theft Technology Council and the Anti-Phishing Working Group. The Crimeware Landscape: Malware, Phishing, Identity Theft and Beyond. Available on-line, October 2006. http://www.antiphishing.org/reports/APWG_CrimewareReport.pdf.

[47] VMWare. VMWare Server: Free Virtualization for Windows and Linux Servers. Available on-line. http://www.vmware.com/pdf/server_datasheet.pdf.

[48] Digital Rights Management v1.0. Technical Specification OMA-Download-DRM-V1_0-20040615-A, The Open Mobile Alliance (OMA) (June 2004)

[49] DRM architecture specification v1.0. Technical Specification OMA-Download-ARCH-V1_0-20040625-A, The Open Mobile Alliance (OMA) (June 2004)