

DRAFT

TCG/OpenSolaris Project Plan

Kwang-Hyun Baek, Sean Smith

1 Introduction

One goal of the Sun-Dartmouth OpenSolaris Security Collaboration is to integrate the Trusted Platform Module (TPM) into OpenSolaris. Trusted Computing Group (TCG) designed the TPM in order to make trusted computing possible on commodity hardware. However, in our previous research of developing an open TPM platform, we identified a couple of problems with the current model of attestation.

New operating system techniques of OpenSolaris, such as ZFS and zones, offer a new way to address these issues. Moreover, since TPM support has not been implemented for OpenSolaris, we are offering a design that can accommodate legacy applications that use the Solaris Cryptography Framework, TCG applications, and new research ideas that would enhance the current model of attestation.

This document describes the project plan and the estimate of the development.

2 Background

The Dartmouth PKI/Trust Lab has developed the Enforcer LSM [3], the world's first open-source TCG platform. During our development of the Enforcer LSM, we have identified two open challenges.

The first notable challenge is strong isolation of the attested application, so that attestation¹ guarantees some property of the attested application and strong independence from the rest of the platform.

Without proper confinement, for example, the root account will be able to spy application's memory space to get the sealed secret out (right after it is unsealed and put into the memory space), or inject code into the memory space after attestation, even if the attested program, libraries, configuration files are known.

We addressed this issue with NSA's SELinux LSM [7, 1], but its fine-grained, local-specific policy was too complicated for a remote challenger to believe security properties of the platform and the application.

The other challenge is making attestation more meaningful. The current attestation model is hard to be used to present security properties of the platform that is needed to make trust judgement [10], and often has to settle with attesting the hash of the binary image and its related files. For example, when trust negotiation occurs between the two parties, what they really care about is some binding agreements: the Diffserv QoS

¹We will use the term attestation in general sense to mean trustworthy reporting of the platform's configuration via a trusted component: attesting the extended PCR values, IWG Integrity Measurement Architecture, and so on.

marker will follow the network’s rule [1]; the SSL server will not export private information [8]. Attesting that the program’s binary hash, configurations, libraries are in known states is just a way to reinforce these agreements. Richer attestation methods, such as property attestation [10] and semantic-based attestation [6], can give higher assurance to the challenger that the software will behave in a known way.

New features introduced by Solaris 10 and OpenSolaris (specifically, ZFS and zones) offer new ways to solve these open challenges.

3 Research

3.1 Motivation

The ultimate purpose of the attestation is to test if the requester’s platform configuration satisfies the security requirements of the challenger. The current model of the TCG attestation uses the TCG IWG Integrity Management Architecture, where the “run-time measurement” of the platform is compared to the database of known values of the measurement. Through knowing that the requester’s measured platform will behave in a known way, the challenger can *trust* the requester.

For attestation to be effective, however, the run-time measurement should include not only the hash of binary files and configuration files, but also the properties of the platform, such as whether some important application in question is running in a confined compartment. Moreover, if the requester’s platform does not meet the challenger’s criteria, the requester may want to increase its security properties (i.e. more confinement) in order to succeed in the negotiation.

Current ideas to use virtualization techniques with TCG attestation considered VM monitors, such as such as Terra VMM [4] and DeuTeRiuM [9]. However, using VMM rather than OS virtualization like Solaris zones and BSD jails lacks flexibility and scalability: The virtualization compartments need to be set up on premeditated partitions and once started, it is hard to change the settings of the virtualized environments. Thus, if the challenger’s criteria for the security setting is slightly higher (a matter of changing the security policy), the requester may not be able to succeed in the negotiation unless either he has another partition available for another virtual machine or he reboots the virtualized environment in the desired setting. Thus, there is a clear need for a more flexible scheme.

One way to resolve this issue is to run high-level virtual machine like Java VM at software stack to contain applications [2]. However, the application needs to be written in the languages that are compatible with the high-level virtual machines. This model does not have the flexibility and backward compatibility we desire.

3.2 Trusted Zones on Demand

We present *Trusted Zones on demand (TZoD)*.

To solve the inflexibility of the other virtualization TCG techniques, we consider a model where a platform, before starting an application, may ask the challenger what kind of environment is needed for the application to succeed in the negotiation. Platform can start a virtualized environment with the property specified by the challenger, run application and use attestation to prove that the attested application meets the needs.

TZoD uses ZFS and Solaris zones to achieve this idea. Zones on top of ZFS are flexible and dynamic: There is minimum install and startup time; zone creation is dynamic; and new zones can be added without carefully planning separate partitions. The confinement guarantee of virtualization can also simplify the platform policy that needs to specify confinement (as long as the challenger believes that mechanism is implemented correctly). The policy for the zone can be included in the zone configuration to specify what

kind of security policy will be enforced for the zone. Finally, a kernel component that has been bootstrapped to be trustworthy can measure the platform's policy for the zones and the applications that run on top of zones.

TZoD must be combined with security monitor to be fully effective. When Trusted Solaris Extension is available to give OpenSolaris Mandatory Access Control, we should be able to examine how effectively TZoD can simplify the MAC policies during attestation.

Another way to use TZoD is to use outbound authentication [11]. We shall collaborate with the CA project to have the platform resident CA create keypairs and certificates for the applications running on the platform so that the application would have access to the key if and only if the application and platform satisfy some policies. Thus, by showing the knowledge of the private key and the certificate showing the related properties associated with it, the application can access guarded resources.

We are currently thinking of using TZoD for different kinds of applications.

4 Goals

4.1 Improving Attestation

Key Goal: Implement a single point of access and management of the TPM from kernel-space to solve the two open challenges we mentioned in Section 2: isolation of the attested program and more meaningful attestation (TZoD).

The TCG specification states that the interface to the TPM should be built in user-space. However, we believe that without kernel level access to the TPM, it will be hard, if not impossible, to provide strong isolation and property attestation of the applications and the zones.

We also plan to build a prototype using the kernel-space interface to the TPM as a proof-of-concept to improve the attestation model, using ZFS and Solaris Zones to confine applications on demand and increase the level of trust provided by the attestation.

4.2 Interoperability

The other goals of the project is interoperability. We want compatibility with Solaris Cryptography Framework and TCG standards so that legacy applications using Solaris Cryptography Framework and TCG applications using TSS APIs [5] can run on Solaris and take advantage of the TPM. We need the following to achieve interoperability:

- Provide TCG Service Provider (TSP) Library [5] so that TCG applications on OpenSolaris can call TSPI (TSP Interface) functions to utilize TPM in order to call TPM functions, such as `TSPI_Seal` and `TSPI_Quote` from user-space applications.
- Implement Solaris kernel-space Cryptographic Framework (kCF) provider for the TPM, so that legacy softwares—both kernel modules and user-level applications—using PKCS11 may be able to benefit from TPM's cryptographic functions and secure key storage without any source code changes.
- Implement a kernel-level framework that provides threaded access to the TPM. This framework will manage scheduling of the kernel threads, context management, and key and credential management, event manager, TPM parameter block generator.

- Implement a user-level service (daemon) that is the user-level interface for the kernel-level TPM framework and functions as TCG Core Service provider. This multi-threaded user-level service will simply relay the TCG Core Services [5] commands to the kernel-level TPM framework.

5 Design Overview

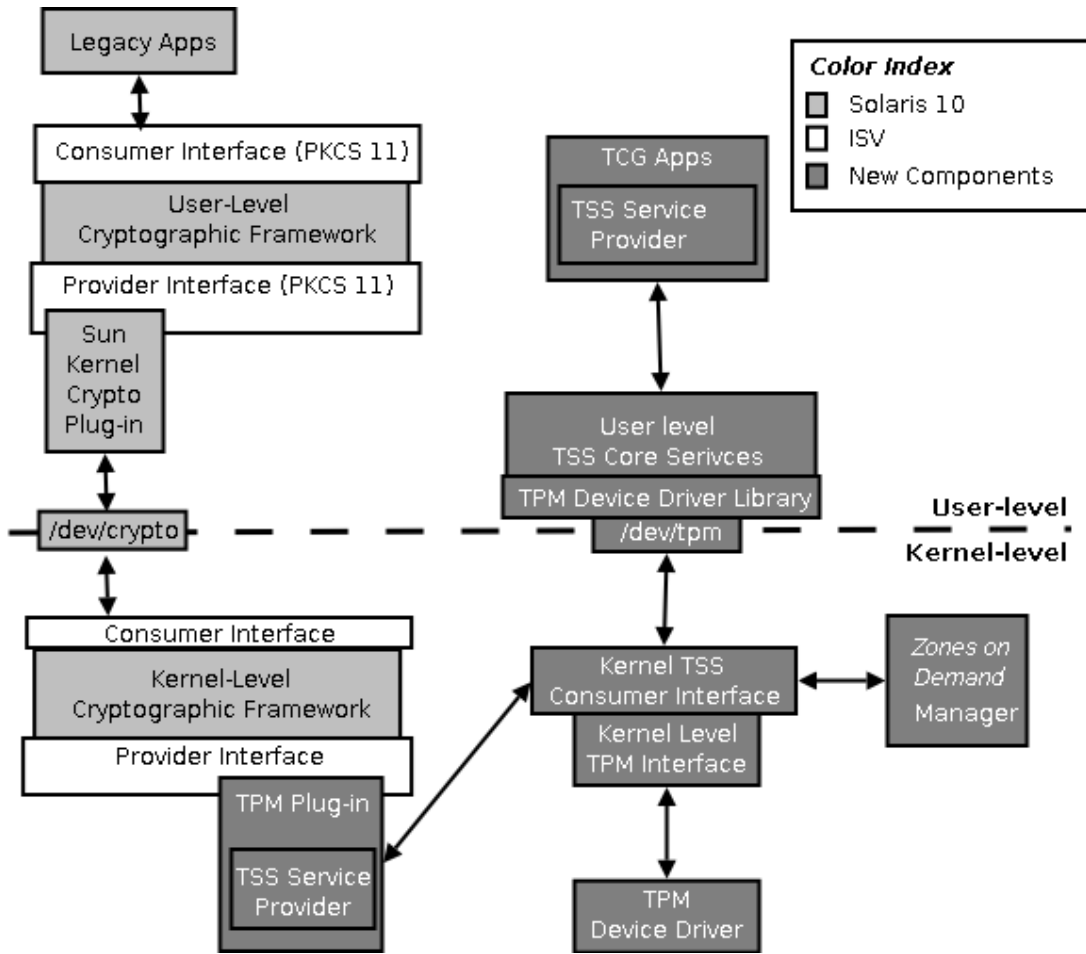


Figure 1: The high-level design of the TCG Solaris Project.

Figure 1 shows the high level picture of the design.

Milestones

1. We shall implement a kernel-level TPM management framework that essentially acts as TCS. This kernel-level framework will provide context manager, key and credential manager, event manager, TPM parameter block generator (the services that are typically provided by the TCG Core Services, TCS, in user-space).
2. We shall implement provider interface for the kernel-level framework in order to serve kernel-level consumers.

Tspi_ChangeAuth	Tspi_ChangeAuthAsym
Tspi_Context_Close	Tspi_Context_CloseObject
Tspi_Context_Connect	Tspi_Context_Create
Tspi_Context_CreateObject	Tspi_Context_FreeMemory
Tspi_Context_GetCapability	Tspi_Context_GetDefaultPolicy
Tspi_Context_GetKeyByPublicInfo	Tspi_Context_GetKeyByUUID
Tspi_Context_GetRegisteredKeyByUUID	Tspi_Context_GetTpmObject
Tspi_Context_LoadKeyByBlob	Tspi_Context_LoadKeyByUUID
Tspi_Context_RegisterKey	Tspi_Context_UnregisterKey
Tspi_Data_Bind	Tspi_Data_Seal
Tspi_Data_Unbind	Tspi_Data_Unseal
Tspi_GetAttribData	Tspi_GetAttribUint32
Tspi_GetPolicyObject	
Tspi_Hash_GetHashValue	Tspi_Hash_SetHashValue
Tspi_Hash_Sign	Tspi_Hash_UpdateHashValue
Tspi_Hash_VerifySignature	
Tspi_Key_CertifyKey	Tspi_Key_ConvertMigrationBlob
Tspi_Key_CreateKey	Tspi_Key_CreateMigrationBlob
Tspi_Key_GetPubKey	Tspi_Key_LoadKey
Tspi_Key_UnloadKey	
Tspi_PcrComposite_GetPcrValue	Tspi_PcrComposite_SelectPcrIndex
Tspi_PcrComposite_SetPcrValue	
Tspi_Policy_AssignToObject	Tspi_Policy_FlushSecret
Tspi_Policy_SetSecret	Tspi_SetAttribData
Tspi_SetAttribUint32	
Tspi_TPM_ClearOwner	Tspi_TPM_CollateIdentityRequest
Tspi_TPM_CreateEndorsementKey	Tspi_TPM_GetCapability
Tspi_TPM_GetEvent	Tspi_TPM_GetEventLog
Tspi_TPM_GetEvents	Tspi_TPM_GetPubEndorsementKey
Tspi_TPM_GetRandom	Tspi_TPM_GetStatus
Tspi_TPM_GetTestResult	Tspi_TPM_PcrExtend
Tspi_TPM_PcrRead	Tspi_TPM_Quote
Tspi_TPM_SetStatus	Tspi_TPM_StirRandom
Tspi_TPM_TakeOwnership	

Table 1: List of the partial TCG Service Provider Interface (TSPI) that we will implement for Milestone 4. We follow the TSS specification [5], and these functions provide the basis for developers to implement TCG-enabled user-level applications.

Milestone	Dates
Milestone 1	June–August 2006
Milestone 2	September 2006
Milestone 3	October 2006
Milestone 4	November 2006–January 2007
Milestone 5	February 2007
Milestone 6	November 2006–February 2007
Milestone 7	March 2007–May 2007
Milestone 8	May–June 2007

Table 2: Project Roadmap.

3. We shall implement a user-level service daemon that serves as the TCS in user-level space.
4. We shall implement TSPI as a DLL written in C, according to the TSS specification so that TCG applications can use the TSS APIs (See Table 1)
5. We shall modify the current TPM Device Driver from Sun to fit our model. (This TPM driver currently includes kernel-level Cryptographic Framework Provider for the TPM, but we will modify it so that it will use the kernel-level TPM management framework. Once this work is done, legacy applications can use PKCS11 APIs to utilize the TPM’s cryptographic functions.)
6. We shall implement the Trusted Zones on Demand basic framework.
7. If the Trusted Solaris Extension is available by then, we shall build MAC policies for TZoD and evaluate how effectively TZoD simplifies the MAC policy.
8. Combined with the CA project, we shall implement outbound authentication protocol with TZoD.

Development Machines We will use IBM Netvista 8310 desktop machines (Pentium 4 2.00GHz, 512MB RAM, one IDE hard drive, running OpenSolaris) with Atmel TPM 1.1b (until the Sun provides a Sun x86 machine with a TPM?).

6 Project Roadmap

Table 2 shows the roadmap and deadline associated with the project. Kwang-Hyun Baek and a supplemental programmer (potentially, Jed Dobson) will be the main developers for this project. Milestone 1, 2, and 3 will be developed by both developers. The supplemental programmer will develop Milestones 4 and 5, and Kwang-Hyun Baek will work on Mileston 6 and 7 and 8.

References

- [1] Kwang-Hyun Baek and Sean W Smith. Preventing theft of quality of service on open platforms. In *First IEEE/CREATE-NET Workshop on Security and QoS in Communication Networks (IEEE/CREATE-NET Sec-QoS 2005)*, Athens, Greece, September 2005.
- [2] R. Cáceres and R. Sailer. Trusted mobile computing. In *Proceedings of IFIP Workshop on Security and Privacy in Mobile and Wireless Network*, May 2006.
- [3] Enforcer homepage. <http://enforcer.sourceforge.net/>.
- [4] T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh. Terra: A virtual machine-based platform for trusted computing. In *Proceedings of Symposium on Operating System Principles*, October 2003.

- [5] Trusted Computing Group. TCG Software Stack (TCG) Specification Version 1.2 level 1. <https://www.trustedcomputinggroup.org/groups/software/>, January 2006.
- [6] Vivek Haldar, Deepak Chandra, and Michael Franz. Semantic remote attestation—a virtual machine directed approach to trusted computing. In *3rd Virtual Machine Research and Technology Symposium (USENIX VM)*, pages 29–41, San Jose, California, May 2004.
- [7] John Marchesini, Sean Smith, Omen Wild, Josh Stabiner, and Alex Barsamian. Open-Source Applications of TCG Hardware. In *20th Annual Computer Security Applications Conference*, December 2004.
- [8] John Marchesini, Sean W. Smith, Omen Wild, and Rich Macdonald. Experimenting with TCG/TCG Hardware, Or: How I Learned to Stop Worrying and Love The Bear. Technical Report TR2003-476, Department of Computer Science, Dartmouth College, December 2003.
- [9] Jonathan M. McCune, Stefan Berger, Ramón Cáceres, Trent Jaeger, and Reiner Sailer. DeuTeRium—a system for distributed mandatory access control. Technical report, IBM Research Division, February 2006.
- [10] Ahmad-Reza Sadeghi and Christian Stübke. Property-based attestation for computing platforms: caring about properties, not mechanisms. In *NSPW '04: Proceedings of the 2004 workshop on New security paradigms*, pages 67–77, New York, NY, USA, 2005. ACM Press.
- [11] S. W. Smith. Outbound authentication for programmable secure coprocessors. *International Journal of Information Security.*, 3(1):28–41, October 2004.