RC23595 (W0504-136) April 28, 2005
Computer Science

IBM Research Report

Trusted Virtual Domains: Toward Secure Distributed Services

Jonh Linwood Griffin, Trent Jaeger, Ronald Perez, Reiner Sailer, Leendert van Doorn, Ramón Cáceres
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

Trusted Virtual Domains: Toward secure distributed services

John Linwood Griffin, Trent Jaeger, Ronald Perez, Reiner Sailer,
Leendert van Doorn, and Ramón Cáceres
IBM Research
T. J. Watson Research Center, 19 Skyline Drive, Hawthorne, NY 10532 USA
{JLG, jaegert, ronpz, sailer, leendert, caceres}@us.ibm.com

Abstract

The focus of trusted computing efforts to date has been to create islands of trust in a sea of distrust, identifying these islands as dependable domains with a solid base that can be used for building applications upon which critical services depend. The aim of our work is to extend this solid base by building "bridges" among trusted islands, with such goals as enabling meaningful trade agreements between islands, enabling migration of individual island inhabitants, and enabling geography-independent affiliation among inhabitants of different islands.

1 Introduction: secure distributed services

Our vision of the future is that directed computation and data analysis will be securely offloadable onto any acceptable service computer, anywhere, that has excess processing capacity. This vision goes beyond the scope of grid computing as available today, in that we further believe that such service offloading should achieve levels of security and dependability that are equal or nearly equal to the levels achieved on the user's original, self-contained system. For usability and scalability, we additionally require that no extra user effort or interaction be necessary to ensure and maintain these properties.

Such a vision is difficult to achieve presently, as it is difficult to know or reason about the security and dependability of a computer system not under a user's direct, administrative control. Any offloading tends to have severe restrictions placed upon the location and manner in which the computation will take place: one business may enter into a lengthy and complicated legal negotiation to strictly specify the physical and software security for machines colocated in a service provider's machine room, or an organization may require that all connections to its internal network be made through a virtual private network (VPN) and a firewall—hoping that the integrity of the VPN software indicates the integrity of the users' systems.

We envision a new environment for distributed, secure computational offloading. Applications or services will be partitioned into selectable components, where each component may be serviced by remote entities acting in certain roles. (Legacy applications could each run as a single unpartitioned component.) Roles for remote entities will include statements of security and operational restrictions on each entity's behavior, and infrastructure around each remote entity will ensure that its behavior doesn't stray from that of its specified role. The crux of this idea is that any entity willing to take on a role—i.e., an entity trying to sell its services or resources for profit—must demonstrate its acceptability to the *initiator* of a service. The role-taking entity, which we henceforth refer to as a *responder*, must satisfactorily establish that it can carry out the responsibilities of the role while at the same time preserving any restrictions that the initiator may place upon it.

In the context of secure distributed computing, we view this evaluation of "acceptability" as that of remotely providing a believable description of the behavior and limitations of the operating environment—what it and its applications are restricted from doing, and what they are permitted to do. Put another way, validating how an environment protects or does not protect computation and the data it receives. This can stem from a direct analysis of the hardware and software components of a system, from a trusted third party certification of the configuration of those components, or from a combination thereof.

In this position paper we identify a new framework for distributed service-oriented processing that we call Trusted Virtual Domains (TVDs). The TVD framework opens the door to realizations of the above ideas by tying together new and previous work on trusted computing, policy specification and verification, virtualization and middleware technologies, and others. TVDs are designed to simplify the user's and administrator's interactions with large-scale distributed systems by offloading the "grunt work"—the analysis and enforcement of the security and operational properties associated with a workload or service—onto the TVD infrastructure itself.

2 Motivational scenarios

In this section we present two example scenarios that highlight the motivation for our position. As discussed in the introduction, our scenarios revolve around two or more parties who collaborate to create a distributed session: an initiator who desires that a service be performed, and one or more responders who perform the service by taking on a role. For the purpose of these scenarios, think of each party as being a process executing exclusively inside its own virtual machine; we expand on this notion in the following section.

Scenario 1: computational offloading. Our first scenario involves initiator-specified computation: in particular, grid-style computational offloading. In this scenario the initiator wishes to run an extensive but sensitive data mining query over a confidential data set, requiring computational and storage resources beyond those the initiator has available. For example, a pharmaceutical company wishes to measure the death rate in patients that are prescribed an experimental drug mix: the query is sensitive, in that there should be no external indication of the search parameters; and the patient data must remain confidential. Computational offloading scenarios are ideally suited for grid computing environments, where a goal is to create homogeneous, widely-available, distributed processing nodes to absorb excess local computational needs. However, although grid servers are gaining in popularity and popular usage, they currently offer few if any remotely verifiable guarantees about the security and integrity of their operating environments, making them unsuitable for application in this scenario.

One of our aims is to eliminate this barrier, enabling such offloading scenarios to become commonplace for anyone who could benefit from them, while simultaneously addressing any security concerns. At role-acquisition time, the initiator might specify that it requires attestations to the effect of processing-time reservations, memory and communication isolation (with respect to other processes or entities running on the responder's system), encrypted on-disk storage of any swapped memory or source data, and confirmation that the responder's execution environment will be reset and zeroed upon completion of the service. The responder's virtual machine would then believably attest or assert that it will enforce each of these requirements. In an expanded scenario, the query may be provided by the initiator, who specifies one set of security requirements regarding the query text, whereas the patient data may come from a third-party source with much stricter requirements of verifying identity and ensuring confidentiality.

Scenario 2: business services. Our second scenario involves responder-specified computations (i.e., advertised services): in particular, online business services. In this scenario the initiator identifies a responder who advertises that it is programmed and willing to accomplish the initiator's high level task. For example, a consumer wishes to order a book from an online broker, but desires to prevent the distributor of the book from learning any information about the consumer other than his or her address—in particular, preventing the exposure of bank or financial information that the consumer discloses in order to pay the broker. Online business services are in widespread use today, but suffer in that their usage is ad-hoc, with consumers relying only on past experience or reputation when verifying the expected behavior of different brokers, and in that users encounter different interfaces for each different broker for a given requested service. In this scenario, the consumer may desire to securely audit the broker's communications—in essence, obtaining a guarantee that the consumer will have knowledge of any unauthorized exposure—in lieu of specifying security parameters for each responder.

Another of our aims is to generalize the parameters under which a responder will operate on the initiator's behalf. The types of attestations and statements exchanged are different in this scenario: early in the book-ordering transaction the responder could specify well-understood labels for the data it will require to perform the service: financial debit data, shipping address data, etc. The initiator can respond by requiring that the responder locate and incorporate two additional responders: one to handle the conveyance of the financial-labeled data to the bank, and the other to handle the conveyance of the address data to the warehouse, specifically directing that these two additional responders must reside on physically separate hardware platforms.

What's missing? Two problems prevent today's technologies from realizing our vision. First is the inability to represent trusted properties: there are no generally-accepted, useful mechanisms for an initiator to negotiate the security properties and requirements it expects from a remote responder. Second is the inability to verify trusted properties: there are no mechanisms for a responder to demonstrate its acceptability upon request.

Solving these problems requires the confluence of three layers. First, it requires establishing that all systems involved in a negotiation are under self-control, uncorrupted by an attacker: a basis for *negotiation*. Second, it requires establishing the operational requirements to which each system must adhere: a basis for *control*. Third, it requires establishing roles for all parties involved in the aggregation: a basis for *service execution*. Although there has been individual work in each of these areas, to our knowledge no directed effort has successfully tied the areas together. The architecture we describe in the following section aims to address the two problems by establishing each of these bases.

3 TVDs: Building upon verifiable trust

We are developing an architecture to build upon the notion of negotiated roles for responders, wherein the roles are defined by the security-related attestations they can provide to an initiator. In particular, our architecture focuses on enabling and supporting execution environments to realize secure distributed services.

A TVD is an abstract union entered into by an initiator and one or more responders, in which the mutual requirements for all parties are specified and confirmed during the process of joining the union. The nature of the TVD is that application-level programmers and users are simply aware that their execution environment supports and enforces semantic operational and security primitives, via a well-defined and straightforward programming interface. The humans are therefore relieved of the complexity of correctly implementing and configuring their programs to achieve the desired secure operational properties. Instead, the mechanisms comprising the execution environment—in our current thinking, Trusted Platform Module (TPM)-based hardware support and one or more virtual machine monitors—transparently handle the connections among and execution monitoring of each of the parties.

3.1 TVD components

More specifically, there are three levels of components in our architecture, corresponding with the three layers presented in the preceding section. The relationships among these components is illustrated in Figure 1.

A basis for negotiation: The mutually-trusted computing base (MTCB). Before any attestations can be made or roles accepted, each party must be assured of the identity and integrity of the remote party's computer system. A good candidate for achieving this involves making use of secure hardware extensions, such as high-end secure coprocessors or commodity embedded security subsystems such as the TPM that many companies (including Dell, Hewlett Packard, IBM, and Toshiba) have announced will be included in their COTS computer system offerings. From our previous experiences with engineering trust inside a single computer system, we believe it is feasible to bridge trust across multiple systems using the TPMs passively as the root of trust on each system.

A basis for control: Attesting virtual environments (AVEs). Building on an established framework for negotiations, AVEs work with the underling hardware (and potentially with underlying software) to create execution environments enforcing the types of attestations described in the previous section. We envision AVEs as supporting a wide variety of environments, ranging from virtualized hardware (e.g., VMware^{TM}, Xen, or the IBM Research Hypervisor) to simpler sandboxed environments such as the Java 2 Platform, Enterprise Edition (J2EE^{TM}). An AVE is confined to a single hardware system, but there can be multiple AVEs running simultaneously on a system.

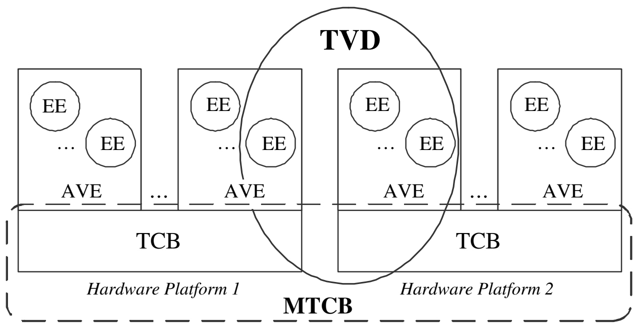

Figure 1: TVD architecture. *The acronym expansions and a description of the functions of each component are defined in Section 3.1.*

A basis for service execution: Execution entities (EEs). Building on an established framework for control, EEs are the individual responders who take on roles on behalf of the initiator. There can be one or many EEs per AVE—i.e., entities under the execution limitations defined by the AVE's configuration and attestation—subject to the restrictions stated by the initiator. We anticipate that there will often be reverse attestations required by the responders: for example, an initiator who desires to access sales figures on IBM's corporate intranet may need to make reverse attestations to the responder (the latter being an internal entity, offering real-time sales data) before the responder consents to releasing the information.

3.2 TVD construction

A TVD is composed of an established MTCB with AVEs and EEs that cooperate to perform a service on behalf on an initiator. When an initiator desires a service to be performed, it creates a TVD consisting of itself and, if present, its AVE and secure hardware. It then determines the roles that need to be satisfied and locates suitable role-taking responders for the roles. (This step is beyond the scope of this discussion, other than to say the AVE may provide support for querying a database or peer-to-peer group to identify potential roles.) It then contacts the responder and invites it to join the TVD. The two TCBs attempt to establish an MTCB; if this fails, both parties are notified and the initiator locates an alternate responder. Once the MTCB is established, the initiator or its agent specifies the requirements to the responders' AVEs, and the responders specify any requirements they may have to the initiator's AVE. The respective AVEs generate attestations that satisfy the requirements (or fail, as above); once these attestations are verified then the initiator and responder proceed as normal.

TVD membership is still a nascent topic, and many interesting questions remain about the nature and application of TVDs. For example, our descriptions present components as having asymmetric security goals: an initiator has one set of requirements to map onto the responder, and the responder has a different set of requirements for the initiator. An alternative view is that the TVD itself has a set of security properties, and all components of the TVD must be configured to uphold those properties. Another open issue involves the susceptibility of commodity trusted hardware to physical attacks: although some secure coprocessors are built with physical tamper-resistance features, many of the commodity TPM hardware solutions have only low-cost measures available and are therefore not impregnable to a determined physical attack. This may mean that there are "flavors" of MTCBs that could be established, where a party can refuse to establish mutual trust with potentially susceptible hardware—or, alternatively, the party could evaluate and manage its per-task risk based on its knowledge of the type of remote hardware in use.

A more difficult question involves how the set of components that form a TVD can change over time. A TVD could expand in several ways: the initiator can identify additional roles and invite additional responders to take on those roles, or one of the responders may require several additional roles to be farmed out to fully complete its role. Ultimately such expansions are capabilities that are grantable by the initiator. Looking ahead, a useful application of TVDs may involve Internet-scale, multi-domain service processing, where a TVD of many components is itself coalesced into a single large entity; this new entity is perhaps capable of acting as a new initiator or responder in taking on another role—with composed, higher-level operational characteristics—to accomplish a higher-level service.

4 Responder attestations

Reasoning about what secure distributed services are necessary requires a common language to describe the properties enforced by the individual execution environments. Such a language must strike the right balance between abstract and concrete: abstract enough that the system administrators and service designers can easily understand and make use of the attestations, yet concrete enough that the statements are actually demonstrable and preferably computable and enforceable by all parties. As an effort toward achieving this language, we offer an initial list of useful and potentially attestable properties in this section.

4.1 Data-related attestations

These attestations relate primarily to the reception, handling, and transmission of code or data provided by the initiator or a third party.

Confinement (isolation, confidentiality). These properties refer to the secrecy of the code and data entrusted to the execution environment, in terms of isolating the memory or other physical resources (especially I/O resources) used by the execution environment. One desired result is that the resources are not visible or accessible by other execution environments sharing the same platform. At a different level, another result is that data identified by a particular label is maintained and accessed separately from data identified by a mutually exclusive label.

Immutability. These properties refer to the enforced read-only nature of code or data provided to the execution environment by the initiator or another third party. This could be accomplished for example by static or dynamic code analysis of the responder, by external management of the memory pages visible to the execution environment, or by creating a read-only API across the interface between the AVE and the execution environment.

Integrity. These properties refer to the nature of integrity labels for data: preventing the acquisition or generation of external low-integrity data by a high-integrity responder, as doing so would degrade the responder into a low-integrity state.

Secure I/O primitives. These properties refer to the availability or required use of any secure I/O primitives available to the responder. The secure I/O can represent a hardware-enabled property—for example, a secondary storage device or network interface card that automatically encrypts data before placing it on the medium—or can represent the manual encryption of data by the execution environment itself before its conveyance to the appropriate hardware device.

4.2 Processing-related attestations

These attestations relate primarily to the computations performed by the responder on behalf of the initiator.

Availability. These properties, normally found in service-level agreements, refer to quality-of-service-type guarantees regarding the resources reserved for the responder's computations. This can include the rate or sum of the allocations of the processor, network, storage, or other real or virtualized devices present in the system. Statements can be made about peak, average, minimum, or maximum usage of the aggregate resources.

Cost and metering. These properties refer to non-repudiable agreements by both parties as to the methods by which the resource usage will be metered, as well as the rate at which the initiator will be charged for resource usage by the responder. This may also specify minimum and maximum charges for performing the service. These properties could also be used to agree upon arbitration scenarios for disputes.

Auditing. These properties refer to the capabilities of the execution environment that allow the initiator to interpose on resources utilized by the responder. As examples, having the initiator mediate which point-to-point network connections may be joined by the responder at connection-time, or having all network traffic from the initiator pass through additional responders (each assuming a role on behalf of the initiator) to monitor the quantity and frequency of network resources allotted to the initial responder.

Reset. These properties refer to an initiator's requirements that the execution environment of the responder be freshly reset (memory scrubbed, all resources reset) at either the beginning or end of its execution. For example, this could be a statement by the execution environment that it will permanently halt the responder when the service is complete, and that it will further securely delete any files written by the responder during its operation.

4.3 Environment-related attestations

These attestations relate primarily to the configuration of the execution environment in which the responder operates.

Redundancy. These properties refer to the physical setup of the initiator's hardware platforms, and any verifiable failure resilience or fault tolerance mechanisms that are in place over the hardware or individual hardware components. For example, the measured presence of redundant power supplies or other hardened hardware components, or the installation of hardware- or software-based data distribution schemes for storage or network components.

Identity. These properties refer to the author or source chain of the software executing inside the responder's execution environment, and may (or may not) uniquely identify a particular instance of the software. These properties may also include identifiers describing execution environment itself (e.g., the hardware's hard-wired identifiers, certificates from the author of the virtual machine monitor, a reproduction of the monitor's configuration file), although such identifiers are more likely to be required to establish an initial trust basis than to establish whether a responder is properly configured to assume a role.

Administration. These properties refer to the physical characteristics of the environment in which the system hardware resides: the identity and contact information of the system administrators; the physical geography in which the server is located; legal, privacy, or ethical considerations identified by the administrators as to the physical operation of the respective computer systems; capabilities or willingness to enter into long-term support contracts for providing computational services.

Accountability. These properties refer to the legal ramifications to the responder's administrators for failing to maintain support of the hardware or software configuration necessary to complete the agreed-upon task: for example, overloading the number of simultaneous services being provided and therefore dropping below the agreed-upon quality of service for one or more resources, or taking redundant backup systems offline during critical operations.

5 Discussion

We are exploring the development of virtual operating environments that use trusted computing components to verifiably self-attest to property statements that describe their own behavior, as well as statements that describe the constrained behavior of applications that execute (or are interpreted) inside the environments. Previous work in this space focuses on the simpler problem of conveying third-party attestations and assertions about a system—for example, attesting that a certain hardware and software configuration has been integrally loaded, and asserting that any system with the loaded configuration meets certain high-level security goals.

Achieving trustable self-attestations is a difficult proposition, and we may not be able to achieve complete success. As this is a speculative position paper, we have neither encouraging nor discouraging results to report. One of our goals with this paper is to engage the community in a discussion of which properties are potentially self-attestable, and how such verifiable self-attestation can portably be achieved—whether using our trusted computing architecture or using other frameworks.

We expect the list of operational properties (Section 4) to be a useful starting point for continuing discussions on the specific impact of trusted computing in distributed, heterogeneous environments, regardless of the particular attestation or assertion mechanisms—self or third-party—used to convey the individual properties among cooperating systems. Although property-based attestation is not itself a novel concept, we are not aware of previous work toward identifying specific composable security-related properties in a trusted computing environment.

More generally, we believe that the TVD abstraction is a useful concept toward the composition of secure distributed services. Current work suggests that each of the three component layers of TVDs are realizable. In addition, TVD deployments should solve problems involving computational offloading and business services as described in our example scenarios in Section 2.

Thought experiments involving these scenarios support our belief in the usefulness of TVDs. Scenario 1 decomposes into a straightforward mapping to both attestations and roles as defined for TVDs. For attestations, the scenario relies on a subset of the attestations in Section 4 passing between a pair of AVEs that have established an MTCB. For roles, the scenario could make use of any of the grid computing environments available today.

Scenario 2 represents a more challenging thought experiment, given that the roles defined by the responder (the book broker) must be automatically partitionable into individual EEs whose AVEs collectively uphold the data labeling and isolation policies required by the initiator. This flexibility requirement for Scenario 2 suggests that a significant looming challenge for enabling secure distributed services may lie in the creation of common, well-structured, role-oriented services by service providers or by third parties writing on behalf of potential initiators. We postulate that this task will be simplified in the context of TVDs, due to the availability of the common security language of attestations that provide an exact mapping of potential customers' security requirements for a particular service.

6 Related work

The enabling technologies for secure distributed services are hot research topics these days. Work by Sadeghi and Stüble [7] aims to enable evaluating which security properties a remote system upholds, while obscuring the details of which hardware and software components are used in the system. The focus of that work is on the protocol-based conveyance of properties, which complements our intended evaluation of the properties themselves.

Sailer and colleagues [8] demonstrate the use of trusted computing hardware to verify the integrity of the software stack loaded on a system. Garfinkel and colleagues [1] use trusted third-party certificates to establish a remote basis for believing the authenticity of a virtual operating environment and to demonstrate that both the environment and the application running therein are unmodified. Haldar and colleagues [3] build upon these concepts by including a trusted Java bytecode analyzer in the virtual operating environment, to monitor the application's adherence to a security policy during execution. They expand upon an attestation taxonomy that is related to ours, but is more limited to the environment of language-based virtual machines. Trusted third party certifications and code analyses may end up being important aspects of deployed TVDs, both in terms of the EEs and the applications running therein; especially as other groups within IBM are exploring middleware-based internal frameworks for EEs.

Aspects of the decentralized enforcement of a comprehensive, centralized secure operational policy are discussed by Gasser and colleagues [2] and more recently by Ioannidis and colleagues [4]. We extend these concepts in the context of trusted computing, and propose merging the operational specification of roles for services with their associated security properties.

Secure distributed services, remote attestation, and the use of trusted computing hardware are also topics that are actively being advanced by various corporations and industry groups, such as with Microsoft's Next-Generation Secure Computing Base and the Trusted Computing Group. Aspects of existing commercial middleware, such as the Microsoft® .NET Web services framework and Common Language Runtime, are very similar to TVDs—and would be even more so if they were coupled with trusted computing technologies.

7 Conclusion

We envision an environment where computing services can be dependably offloaded into execution environments that demonstrably and satisfactorily meet a desired set of security requirements. Toward this end, we present a new abstraction whose purpose is enabling computer systems to autonomously reason about the security properties provided (or not provided) by other systems in a widely distributed environment. This new abstraction, that we call Trusted Virtual Domains, is intended to serve as a foundation for dynamically constructing secure distributed services.

Acknowledgements

This work represents one portion of a larger effort among our colleagues toward developing the concept of Trusted Virtual Domains and exploring the impact of TVDs on software and service deployment; complementary work is proceeding at IBM's Tokyo Research Laboratory [5] and Zurich Research Laboratory [6]. This paper benefits greatly from comments and context provided by Günter Karjoth, Hiroshi Maruyama, Matthias Schunter, Sachiko Yoshihama, and the four anonymous reviewers.

References

[1] T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh. Terra: A virtual machine-based platform for trusted computing. In *Symposium on Operating System Principles*, pages 193–206. ACM Press, Oct. 2003.

[2] M. Gasser, A. Goldstein, C. Kaufman, and B. Lampson. The Digital distributed system security architecture. In *National Computer Security Conference*, pages 305–319. NIST/NCSC, Oct. 1989.

[3] V. Haldar and M. Franz. Symmetric behavior-based trust: A new paradigm for Internet computing. In *New Security Paradigms Workshop*, Sept. 2004.

[4] S. Ioannidis, S. M. Bellovin, J. Ioannidis, A. D. Keromytis, and J. M. Smith. Design and implementation of Virtual Private Services. In *IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 269–274. IEEE Computer Society, June 2004.

[5] H. Maruyama, F. Seliger, N. Nagaratnam, T. Ebringer, S. Yoshihama, S. Munetoh, and T. Nakamura. Trusted platform on demand. Research Report RT0564, IBM Corporation, Feb. 2004.

[6] J. Poritz, M. Schunter, E. V. Herreweghen, and M. Waidner. Property attestation—scalable and privacy-friendly security assessment of peer computers. Research Report RZ3548, IBM Corporation, May 2004.

[7] A.-R. Sadeghi and C. Stüble. Property-based attestation for computing platforms: Caring about properties, not mechanisms. In *New Security Paradigms Workshop*, Sept. 2004.

[8] R. Sailer, X. Zhang, T. Jaeger, and L. van Doorn. Design and implementation of a TCG-based integrity measurement architecture. In *USENIX Security Symposium*, pages 223–238. USENIX, Aug. 2004.