Attestation: Evidence and Trust

George Coker, Joshua Guttman, Peter Loscocco, Justin Sheehy, and Brian Sniffen

The MITRE Corporation
guttman,justin,bsniffen@mitre.org
National Security Agency
gscoker,loscocco@nsa.gov

Abstract. Attestation is the activity of making a claim about properties of a target by supplying evidence to an appraiser. We identify five central principles to guide development of attestation systems. We argue that (i) attestation must be able to deliver temporally fresh evidence; (ii) comprehensive information about the target should be accessible; (iii) the target, or its owner, should be able to constrain disclosure of information about the target; (iv) attestation claims should have explicit semantics to allow decisions to be derived from several claims; and (v) the underlying attestation mechanism must be trustworthy. We propose an architecture for attestation guided by these principles, as well as an implementation that adheres to this architecture. Virtualized platforms, which are increasingly well supported on stock hardware, provide a natural basis for our attestation architecture.

1 Introduction

Much economic activity takes place on heterogeneous networks of computers, involving interactions among autonomous principals, including individuals, retail companies, credit card firms, banks, and stock brokerages. Because the amount of money in these activities is large and increasing, the networks are attractive targets for criminals.

In many attacks, the adversary inserts software remotely, without physical access to the devices, and this software compromises secrets. For instance, in March 2008, an attack was announced against the large American grocery store chain Hannaford Brothers. Unauthorized code had been inserted on the servers in each of the company's 300 stores. This code retained the credit card information for each transaction occurring at a store, and periodically transmitted the information to a third party. As a consequence, over 4,200,000 credit and debit cards were compromised. At least 2,000 fraudulent transactions have been identified as results. Even though Hannaford's systems were designed not to store customer payment details, and to adhere to compliance standards of the credit card companies, changes to their application software led to large disclosures [15].

An even larger case led to indictments in August 2008. Over 40 million card numbers were stolen from US companies such as TJX, a clothing distributor

and retailer, and other large firms. According to the indictment papers, eleven criminals collaborated in this group of attacks. Members were located in the US, Estonia, Ukraine, Belarus, and China. In these attacks, wireless access points were the initial entry point. Newspapers described the inserted software as sniffers. However, the indictments mention that an insecure wireless access point at a Marshall's retail store in Florida allowed the defendants to compromise data stored in servers at TJX, located in Massachusetts [24].

In both of these cases, the inserted software appears to have remained undetected for months.

There are three characteristics of these attacks. First, the attacks are executed remotely, apparently without physical access to the computers attacked. Second, the computers are standard, general purpose systems, rather than specialized devices such as automated teller machines. Third, the networks involve transactions among independent organizations, such as a retailer, a distributor, and the credit card firms. No one organization controls the software configurations on all the relevant systems. The ubiquitous attacks that insert malware onto individually owned computers, to sniff for bank account and password information, share these characteristics. The bank cannot control the configurations of its customers' computers. Nevertheless, there would be benefits shared by the bank and its customers if the bank could ascertain that the customer's computer was free of malware, before allowing the customer to enter the account number and password.

Attestation means providing reliable evidence about the state of software executing on a system. Many computing problems could be solved if attestation were reduced to practice, particularly attestation that provides evidence of the behavioral properties similar to those mentioned in the attacks we have mentioned. To achieve this goal, attestation must make sense for general-purpose systems, running varied configurations, and under the control of different individuals and organizations. The participant's privacy goals must be respected, while providing evidence that distributed transactions are not being subverted.

One might think that attestation in this sense would be impossible. What evidence of a good state can be provided to a remote party, that could not be synthesized by bad software? We do not—by analogy—ask someone we suspect of being a scam artist whether he is honest, or at least we do not count it as evidence when he says that he is. However, a starting point for trust appraisal now exists, in the form of the Trusted Platform Module (TPM), introduced by the Trusted Computing Group (TCG) [2]. The TPM provides primitive cryptographic capabilities, and some long-term storage that is secure from remote attacks, which can be used to provide signed evidence from Platform Configuration Registers that reflect key events of the software controlling the machine.

Existing attestation proposals, including those put forth by the TCG, are generally aimed at specific use-cases and typically lack flexibility to address a more general attestation problem. Further, existing *definitions* of attestation primarily focus on describing the particular properties [19] desirable in those use-cases. For example, in [7], the author uses the term attestation to specifi-

cally mean the process of transmitting a sequence of hashes of certain system components and a digital signature of that sequence; in Microsoft's "NGSCB" [6] it refers to identification and authentication of known code via digital signatures; Copilot [13] makes use of direct hashes of kernel memory, and so on. We prefer a general definition of platform attestation that abstracts from specific desired properties [3].

In this paper, we describe a flexible attestation architecture, based on a few guiding principles. Systems built according to this architecture can be composed to carry out attestation scenarios. We believe that this attestation architecture provides the mechanisms needed for systems to interrogate each other before sensitive interactions, so as to ensure that those interactions will be safe.

2 Attestation

Our approach to system attestation departs significantly from the notion put forth by the TCG, in great part due to increased flexibility. Emphasis is placed on attestation based upon *properties* of the target, useful in a variety of scenarios, rather than solely on attestation based upon *identity*.

Terminology. *An* appraiser *is a party, generally a computer on a network, making a decision about some other party or parties. A* target *is a party about which an appraiser makes such a decision.*

The trust decision made by an appraiser often supports an access request made on behalf of the target, and is usually a decision about the expected behavior of that target. To make a decision on this basis, a diligent appraiser needs a significant amount of information—essentially, the knowledge that the state of the target is such that it will not transition into an unacceptable state while the appraiser still continues to trust it. There is some inevitable tension between the human organizations behind the appraiser and target, as the appraiser's owner wishes to have complete and correct information about any given target while the target's owner wishes to give up no more than the minimal information necesssary for the success of its request (and perhaps even less).

Terminology. Attestation *is the activity of making a claim to an appraiser about the properties of a target by supplying evidence which supports that claim. An* attester *is a party performing this activity. An appraiser's decision-making process based on attested information is* appraisal.

In the most commonly addressed class of attestations, each attestation provides a means for appraisers to infer that the target of the attestation will not engage in a class of misbehaviors. For example, if the target reports its kernel is unmodified, the attester has reason to trust reports from the target, and some appraiser trusts information provided by the attester, then that appraiser can infer that the target will not engage in misbehaviors that might have occurred had the target's kernel been corrupted at the time of its measurement. It is important to note that not all attestations are about lack of misbehaviors, even though most of the commonly discussed use cases are in that class.

This broader point of view makes a rich understanding of the related concepts of system measurement, attestation protocols, and system separation vital to successful attestation. Here there is a distinction between the measurement of a target system (the evidence) and the attestation itself.

Terminology. *To* measure *a target means to collect evidence about it through direct, local observation.*

Attestation about a target system will report measurements or conclusions inferred using measurements and possibly also other attestations. In this paper, measurement is discussed only as necessary to support our architecture for attestation.

Terminology. *An* attestation scenario *is a cryptographic protocol involving a target, an appraiser, and possibly other principals serving as trust proxies. The Trusted Platform Module residing on the target may also be considered a principal in an attestation scenario. The purpose of an attestation scenario is to supply evidence that will be considered authoritative by the appraiser, while respecting privacy goals of the target (or its owner).*

Evidence may be attested to in a number of equivalent but semantically different forms depending on the attestation scenario. For example, the attestation may report raw evidence as directly observed, as reduced evidence (e.g. a hash of the raw evidence), or by substitution with a credential provided by a third party evaluator of the raw evidence. For example, an SSL certificate authority consumes many attestations as to the identity and practices of a target, then produces a certificate attesting to the quality of a target [3].

Also, a given target may wish to provide different information to different appraisers depending on the current trust relationships it has with those parties. A worthwhile desire in developing an attestation system is to resolve the mutual tension as well as posssible given the contradictory nature of the parties' interests. One approach to defusing this tension is for the appraiser to demand frequent repeated attestations, re-evaluating its trust decisions often. It may be possible to determine that a party will be sufficiently trustworthy for the 15 minutes after performing a given attestation, but not feasible to determine that it will be so for a day.

3 Principles for Attestation Architectures

Five principles are crucial for attestation architectures. While an ideal attestation architecture would satisfy all five, they may not all be satisfiable in some kinds of systems. Thus, attestation mechanisms may emphasize some features over others. The five principles motivate the architecture presented in Section 4.

Principle 1 (Fresh information) *Assertions about the target should reflect the* running system*, rather than just disk images. While some measurement tools may provide start-up time information about the target, others will inspect the*

current state of an active target. An attestation architecture should ensure access to the live state of the target. □

The architecture cannot predict the uses to which appraisers will put the information it delivers. Appraisers may need to make very different decisions, and—to justify them—need to make different predictions about the future behavior of the target. This suggests the next principle.

Principle 2 (Comprehensive information) *Attestation mechanisms should be capable of delivering comprehensive information about the target, and its full internal state should be accessible to local measurement tools.* □

With comprehensive information come worries about the consequences of disclosure. Disclosure may cause loss of privacy for a person using the target platform. It can also subject the platform to attack, for instance if the attestation discloses an unpatched vulnerability to an adversary.

Principle 3 (Constrained disclosure) *Targets should be able to enforce policies governing which measurements are sent to each appraiser.* □

Hence, an attestation architecture must allow the appraiser to be identified to the target. Policies may distinguish kinds of information to be delivered to different appraisers. The policy may be dynamic, relying on current run-time information for individual disclosure decisions. For instance, a target may require that the appraiser provide an attestation of its own state, before the target discloses its own.

Principle 4 (Semantic explicitness) *The semantic content of attestations should be explicit.* □

The identity of the target should be included, so an appraiser can collect attestations about it. The appraiser should be able to infer consequences from several attestations, e.g. when different measurements of the target jointly imply a prediction about its behavior. Hence, attestations should have uniform semantics, and so that they are composable using valid logical inferences.

Principle 5 (Trustworthy mechanism) *Attestation mechanisms should provide evidence of their trustworthiness. In particular, the attestation architecture in use should be identified to both appraiser and target.* □

There will be a good deal of natural variation in how different systems meet these principles, and in the choices they make when some principles are only partially satisfied. In specific situations, it may be sufficient to satisfy these principles only partly. For instance, limited forms of evidence about the target may suffice for an appraiser, or evidence that has aged to an extent may be accepted. When different degrees of adherence to the principles are designed into a system, then the variation is static. When the system adjusts at runtime to provide different degrees of evidence in different situations, or when different peers are the appraiser, then the variation is dynamic.

4 Proposed Attestation Architecture

There are five main constraints, imposed by the principles of Section 3, that provide the content for the proposed architecture. In this section, each constraint is briefly described in the context of how it is motivated by the principles. A system designed according to this architecture must have the following abilities:

1. To *measure* diverse aspects of the target of attestation;
2. To *separate domains* to ensure that the measurement tools can prepare their results without interference from the (possibly unreliable) target of attestation;
3. To *protect itself*, or at least a core *trust base* that can set up the domain separation mechanism, ensuring that it cannot be weakened without this fact being evident from the content of attestations;
4. To *delegate attestation* so that attestation proxies can collect detailed measurements and convincing evidence, and summarize them to selected peers, when the target would not permit the full facts to be widely shared;
5. To *manage attestation* to handle attestation queries by invoking suitable measurement tools, delivering the results to the appraiser or a proxy as constrained by policies.

These constraints are discussed in turn.

4.1 Measurement Tools

Providing comprehensive information about a system (satisfying Principle 2) requires the ability to provide a collection of tools that (jointly) comprehensively measure the target.

Comprehensive measurement of a system requires more than simply the ability to read all of the data contained in that system. It also means that some measurement tools must understand the structure of what they are measuring. For example, just being able to scan and hash the memory used by an operating system kernel may not suffice to provide useful measurements of it. *Usefulness*, here, is in the eye of the appraiser, and typically involves evidence about the past or future *behavior* of the target. The state of a program changes during execution, and therefore cannot be measured by simple hashing. For this reason, measuring complex system components requires knowing the structure of the targets. Some trust decisions require these structure-sensitive measurements.

As a result of this, there cannot be a "one size fits all" measurement capability for attestation. Different measurement tools must be produced for measuring components with different structure. Further, the complete set of such tools that will be desired is not knowable ahead of time without restricting the target systems from ever adding any new future applications.

Our architecture must support a collection of specialized measurement tools, and in order to be able to provide evidence for arbitrary future attestations it must also support adding new tools to that collection over time.

In addition to measurement capacity being comprehensive, freshness is also a goal. (Principle 1) This means that our measurements cannot always be performed a priori – one must be able to measure various parts of a system on demand. These demands are made from the point of view of an appraiser. A remote party must be able to trigger measurement; it is insufficient to only have runtime measurement occur via periodic automatic remeasurement triggered by the measurement system or tools.

4.2 Domain Separation

For a measurement tool to provide information about a target of attestation, the measurement tool must be able to deliver accurate results even when the target is corrupted. This is an important consequence of Principle 5.

There are two parts to this. First, it must have access to the target's state so as to be able to distinguish whether that target is corrupted or uncorrupted. This state includes the target's executable code but also modifiable data structures that determine whether its future behavior will be acceptable. Second, the measurement tool's state must be inaccessible to the target, so that even if the target is corrupted, it cannot interfere with the results of the measurement.

There are different ways that this separation can be achieved. One is to virtualize the target, so that the measurement tool runs in a separate virtual machine (VM) from the target [12]. The virtual machine monitor must then be able to control cross-VM visibility so that the measurement tool has read access to the target. It must also ensure that the target does not have any control over the measurement tool. There may be a message-passing channel established between them, but the hypervisor/VMM must be able to ensure transparent visibility of the measurement tool into the target and protection of those tools from the target.

Alternatives are possible. For instance, CoPilot (Section 7) uses a form of hardware separation in which the measurement tool runs on a coprocessor and the visibility constraints are expressed via hardware instead of being based on the configuration of a hypervisor.

Given the improved virtualization facilities that new processors from Intel and AMD provide, the VM approach seems like a natural approach that makes minimal requirements beyond standard commodity hardware.

4.3 Self-Protecting Trust Base

We have established that domain separation is necessary in order to have trust in attestations and specifically in the integrity of our measurement tools. This raises a question: how to produce assurance for the integrity of the domain separation itself?

The core of our system's trust must come from components which are simple enough or sufficiently evaluated that one can be convinced that they do not require remeasurement after they have been running. Part of this core must

obviously include the hardware used as our Trusted Computing Base. Any other component must either be measurable from a place that it cannot control or must be sufficiently measured via a static startup measurement taken before it begins operating.

Note that what is needed here is more than just a trusted static subset of our system. The difficulty here is that our trust base must be sufficiently simple and static to not require remeasurement, but also sufficiently rich to bootstrap our measurements and attestations. Anything performing measurement and attestation on the platform will in turn require measurement and attestation about itself in order to convince an appraiser of its trustworthiness. It must be ensured that this chain "bottoms out" at something sufficient to perform certain essential measurements and attestations to support the chain above it while being simple enough that static startup-time measurements are sufficient to determine trust.

It is not trivial to determine the content of this static trust base. One of the difficulties arises around the element of domain separation. It is preferable for the domain separation mechanism to be simple and secure enough to belong in this element, but no hypervisor exists today that satisfies those properties and is also rich enough to provide the services desired. This difficulty is addressed in Section 6. One possible alternative is that a hardware component provides runtime measurements of the domain separation mechanism.

4.4 Attestation Delegation

In practice, the appraiser will need to delegate many aspects of determining the quality of the target to specialists called *attestation proxies*. There are two essential reasons for this.

First, Principle 2 contains an intrinsic conflict with Principle 3. The former states that comprehensive insight into the state of the target must be available. The latter says that the target should be able to choose and enforce a policy on the disclosure of information about its state.

A natural way to reconcile these two principles is to allow appraiser and target to agree on an attestation proxy that is partially trusted by each [3]. The target trusts the proxy to disclose only information about its state which is of limited sensitivity. The appraiser trusts the proxy to make statements only when they are warranted by appropriately fresh and comprehensive information about the target.

The second reason why attestation proxies are important is that they can function as *specialists*. Enormous expertise is needed to interpret detailed measurements, such as those needed to predict behavioral properties about an operating system. An appraiser may get more reliable information and more usable information from an attestation proxy than it would be able to extract on its own from the comprehensive data. The maintainers of an attestation proxy can ensure that it has up-to-date information about the strengths and weaknesses of specific system versions or configurations.

Naturally, these delegations require protocols that allow the principals to ensure they are communicating with appropriate proxies. These protocols must

supply the principals with messages that unambiguously answer the principals' questions. The design of such *attestation protocols* may follow the methods of the strand space theory [9], and may use the strand space/trust management connection from [11, 10].

These delegations, combined with attestations satisfying Principle 4, enable a powerful new capability. An appraiser may compose separate layered or orthogonal attestations, involving different proxies, in order to make a judgment. Another approach, "Layering Negotiations [14]," has been proposed for flexible and composable attestation. We have used many of the same tools as this work, such as Xen and SELinux. The layered negotiations have a fixed two-level structure and are intended to enable distributed coalitions. Our approach is intended to enable general, arbitrarily flexible composability regardless of application usage model.

4.5 Attestation Management

A goal of our architecture is flexibility. It is essential that our system be able to respond meaningfully to different requests from different appraisers without having pre-arranged what every possible combination of attestations might be.

One way to support this notion is with an architectural element referred to as the *Attestation Manager*. A component realizing this idea contains a registry of all of the measurement and attestation tools currently on the platform, and a description of the semantic content produced by each. As a consequence of Principle 4, this component can select at runtime the appropriate services needed to answer any query which could be answered by some subset of the measurement and attestation capabilities currently on the system.

As an Attestation Manager will clearly be involved in nearly every remote attestation, it is also a natural place to enforce some of the constrained disclosure called for by Principle 3. It can restrict the services it selects based on the identity of the party the information would be released to, according to locally-stored access policies. In order to defend this capability from both the untrusted target of attestations and also from potentially-vulnerable measurement tools, the Attestation Manager must be protected via domain separation.

Our attestations will have to use cryptography in order to protect communications from adversaries. This same protection, taken together with domain separation, means that the target can be safely used as an intermediary for communication with appraisers or proxies. This leads to the very beneficial result that an Attestation Manager can be a purely local service; it does not need to be directly accessible by any remote parties.

4.6 The elements of the architecture

One might envision the elements of our architecture fitting together conceptually like so:

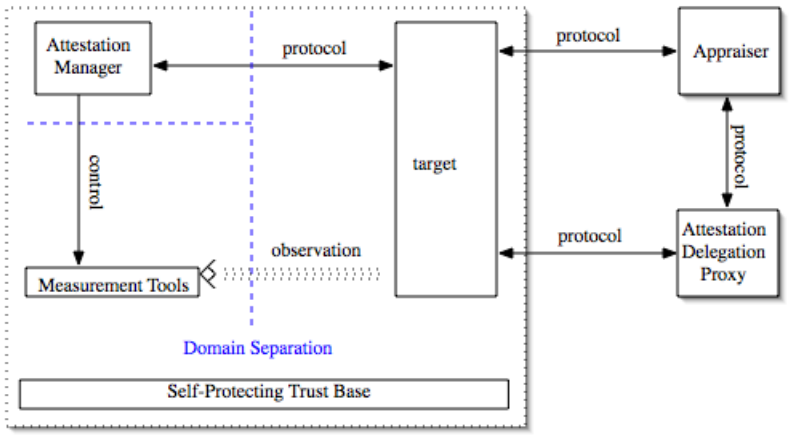

Fig. 1. Attestation Architecture

5 Composable Attestation Platform

An implementation of our attestation architecture has been developed and is illustrated in Figure 2. The hypervisor, together with the CPU, serves as the self-protecting trust base; however, the representation here is abstract, as the implementation is not tied to features specific to any one virtual machine monitor or microprocessor. The Supervisor guest (S guest) contains the platform support package, while the User guest (U guest) runs the user's "normal" operating system. The TPM hardware resource has been virtualized ("vTPM") to provide TPM capabilities for both the S and U environments. Both environments possess measurement and attestation services ("M&A").

The construction and operation of the hypervisor and each guest coincides with the collection of evidence reportable in attestations of the platform. The reason for multiple separate M&A capabilities is that evidence reported from a single party may not be sufficient.

To manage a diversity of measurement and attestation requirements, a virtual machine is dedicated to measurement and attestation (M&A) of a guest. The hypervisor is integral to the trust base for the system, controlling sharing and providing domain separation. Additional domain separation and controlled sharing requirements are met by instrumenting the M&A on SELinux [16]. The separation of the M&A and guest environments is transparent to the target and the attestation. Upon receiving attestation requests, the guest directs the incoming request to its M&A and proxy responses from that M&A back out to appraisers. To deeply assess the platform, one may need to connect attestations together across the S and U environments. This need can only be satisfied with semantically explicit attestations as described by Principle 4.

An M&A consists of three components: attestation manager (AM), attestation protocols (APs), and attestation service providers (ASPs) The AM manages the attestation session, listening for incoming attestation requests and using a "selector" subcomponent for initiating APs. An AP is a running instance of an attestation protocol initiated by the Selector in response to an attestation request. The ASPs are subcomponents of the attestation protocol. Each ASP performs a well-defined service in the attestation protocol and as defined serve a critical role in satisfying Principles 1- 5 for the platform. Some example ASPs are integrity measurement systems, wrappers for calls to a TPM/vTPM, or invocation of other services. As a result of separating these key services into ASPs which may be used by different APs, and abstracting over APs using the AM, we gain an extensible system with the ability to add new services and protocols without the need to redesign or re-evaluate the entire system for each addition.

The Selector is the mechanism for enforcing the policy of the client by instantiating APs and ASPs that conform to the policy for a given scenario thereby satisfying Principle 3. The implementation uses a method referred to as "Call by Contract" [17] for the Selector.

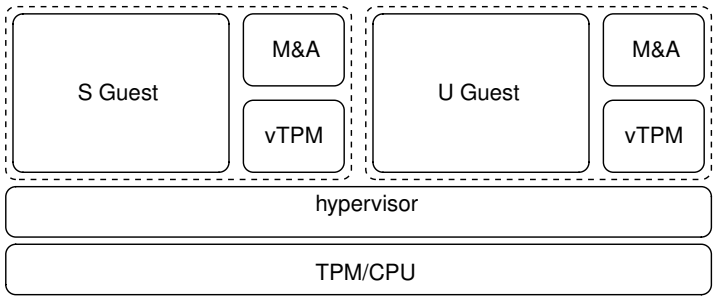

Fig. 2. Composable Attestation Platform

Attestations may be chained across the platform by the use of ASPs that make attestation requests to the other M&A environments and relay or use the attestation responses. Figure 3 shows a possible set of components that might be used in an attestation, including an ASP in the User M&A which makes an attestation request to the Supervisor M&A, enabling attestations which satisfy Principle 5.

The attestation research to date has focused exclusively on the attestation of the User OS kernel and the core platform (the Supervisor guest and hypervisor components). The attestation of these components forms the trust base for the attestation of higher level components, i.e. User guest applications. To support attestation of User guest applications, one could instrument an M&A in user-space that is similar in form and function to the M&A described above. The user-space M&A may be chained to the User and Supervisor M&A's to enable

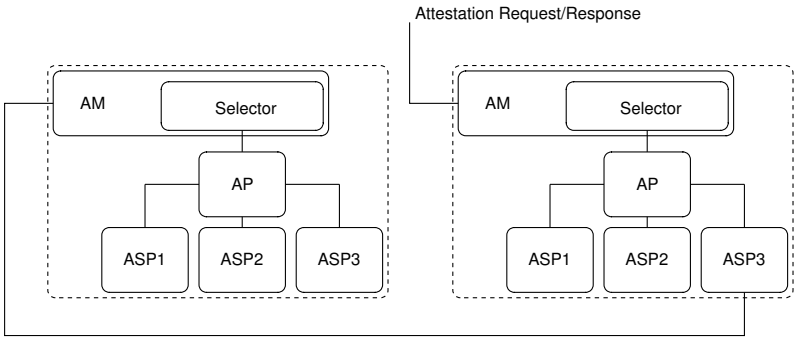

Fig. 3. Composable Measurement and Attestation (M&A) Architecture

complete platform attestations. Furthermore, the implementation is guest OS agile, as the only guest specific components exist entirely within the individual ASPs and the future user-space M&A.

Mutual attestation is an important area to consider, and our architecture provides a natural place for this. APs within an M&A may demand attestations as well as providing them, and may use ASPs for verification of the properties asserted by such attestations.

6 Open Problems

Even with our architectural constraints and system design, some aspects of the attestation problem remain difficult to solve. The most difficult principles to satisfy with today's technology are the **Trustworthy Mechanism** and gathering **Comprehensive Information**.

The Trusted Platform Module and related technology from Intel (TxT) [5] and AMD (SVM) [4] are useful means for bootstrapping certain aspects of a self-protecting trust base, but a richer trust base is needed than can be provided by this sort of hardware alone. The emergent hardware technologies only start the problem from a known origin, the core root of trust for measurement, but ultimately the integrity of the trust base depends on the assurance of the "hypervisor" implementation. Specifically required is a means to establish domain separation in order to support a trustworthy mechanism for attestation. Our current implementation uses an off-the-shelf virtualization system – but none of those available today offer the desired balance between flexibility and security. Solutions to this problem might be found either by extending traditional separation kernels or possibly by producing a small, assurance-focused virtual machine hypervisor.

The major problem in gathering comprehensive information is that in order to establish trust in application-level software one first needs to establish trust in the operating system that the software depends on. Today's mainstream oper-

ating systems were not designed with assurance or measurement in mind. They are large and complex, containing many dynamic features that make them very difficult to analyze even in the absence of a hostile party. It seems unlikely that this situation will improve until there is either a major shift in the structure of mainstream operating systems or the adoption of a new operating system designed from the beginning with measurement and assurance as a design goal.

7 Existing approaches to attestation

There are several early steps toward system attestation in the research community and commercial market today. It is clearly a major component and focus of work being done within the Trusted Computing Group [25] [26] [8], Microsoft [6], and multiple independent researchers [13] [21]. Many of these solutions may act as useful components in a general attestation architecture as described in this paper. Still, none of them fully address this broader notion of attestation or the needs of a flexible architecture.

Trusted Network Connect. Trusted Network Connect (TNC) is a specification from the Trusted Computing Group [26] intended to enable the enforcement of security policy for endpoints connecting to a corporate network.

While Trusted Network Connect is an architecture for attestation, it is of much narrower scope than our approach. Its purpose is to provide trust in endpoints connecting to a network [26], and for this reason it is generally seen as supporting activity at network layers 2 or 3. For this reason, the TNC architecture makes some assumptions about the relationships between parties that make it of limited value for application-level attestations. Once a party has network access, it moves outside the scope of TNC.

In our framework, TNC is best seen not in comparison to our entire architecture but as a special kind of attestation manager. Much of the purpose of the TNC Client (TNCC) is to select the appropriate Integrity Measurement Collectors (IMCs) based on requests from Integrity Measurement Verifiers.

Useful domain separation is not possible in TNC. At load time, each IMC registers what kinds of messages it wishes to receive from the client. If it registers `0xffffffff` then it will receive all messages delivered to all IMCs [27]. Further, it is explicit in the specification that IMCs are loaded into the same memory space as the TNCC, and that a rogue IMC can read and write memory in the TNCC or in other IMCs, misusing credentials, privileges, and message data arbitrarily [27]. Thus, even if the overall TNC process is separated somehow from the target, it is clear that no separation is possible between measurement tools and either the attestation management function or other measurement tools in a system compliant with TNC.

The notion of attestation delegation exists in TNC, but in a very constrained way. The relationships between Policy Enforcement Points and Policy Decision Points is made explicit, making arbitrary delegation difficult at best.

TNC can provide identification of the appraiser to the target, though it is constrained to one very specific identification. Before the integrity measurements

are taken, "mutual platform credential authentication" [26] can occur. In the TCG context, this means that the two parties can each verify that the other has a valid unrevoked TPM AIK. However, truly mutual authentication is impossible in TNC due to its nature as a network access protocol. Given that the "server" implicitly already has access, no attestations from the server to the client other than this initial credential exchange is possible. If the client only requires a basic identification then this may be sufficient, but if clients wish to negotiate with servers and proceed differently depending on attested properties, then TNC is unsuitable.

It should be noted that TNC is not a networking or messaging protocol, but rather is intended to be tunneled in existing protocols for managing network access, such as EAP [1].

Due to the asymmetric nature of TNC and the protocols it expects to live within, implementation of complex attestation protocols or nested attestations is unlikely to occur in a way that interoperates with TNC.

Pioneer and BIND. Pioneer and BIND are attestation primitives developed at CMU with very specific design constraints.

BIND [22] is a runtime code attestation service for use in securing distributed systems. It centers around a specific measurement capability which binds a proof of process integrity to data produced by that process. For embedded systems which without flexible attestation needs, BIND may be useful.

Pioneer [21] is an attempt to provide a "first-step toward externally-verifiable code execution on legacy computing systems." Here, legacy means systems with no hardware trust base – Pioneer attempts to solve the attestation problem entirely in software. This faces serious challenges in the presence of a malicious OS, and at least one method is known for an OS to fool Pioneer. Also, the success of Pioneer on any given system requires an immense degree of knowledge about (and control of) the underlying hardware. A trusted third party must know the *exact* model and clock speed of the CPU as well as the memory latency. The system must not be overclocked, must not support symmetric multi-threading, and must not generate system management interrupts during execution of Pioneer. This level of dependency suggests that an attacker with sufficient understanding of the hardware might subvert attestation. In specific, at least one such attack is known in the case of systems with 64-bit extensions. The specific weaknesses referred to are acknowledged by the authors as implementation issues [20].

Another requirement for Pioneer is that the checksum code is the most time-optimal possible code that performs its checksum. No proofs of such optimality exist for any Pioneer-sufficient checksum functions. It remains to be seen if Pioneer can succeed, as newly emerging hardware optimizations will continue to provide attack vectors and make it very difficult to be certain that a given piece of code is time-optimal on all architectures that a user may care about.

Copilot. CoPilot [13] is a system for detecting root kits in a Linux kernel. It periodically computes hashes over key parts of memory that impact kernel execution, compares against a known value, and reports to an external system

that enables manual decisions to be made regarding detected changes. CoPilot runs on a PCI add-in card, accessing system memory using DMA. It uses a dedicated port to communicate with the appraiser.

CoPilot does well with respect to some of our principles. It is protected due to existing on separate hardware and via a direct connection to the appraiser. It produces fresh information about a running system.

CoPilot is an advance in attestation technology, but it has limitations. It does not provide a truly comprehensive measurement of the target system because the measurements it produces do not include important information residing in the kernel's dynamic data segment. In addition, since CoPilot does not have direct access to the CPU's registers, it only is able to perform measurements at known memory locations and cannot associate any of its measurements with what is actually running on the processor. Also, the timing of measurement cannot be coordinated with actions on the target. This means that appraisal is difficult as any given measurement might be taken during a moment of inconsistency. Achieving any kind of constrained disclosure is not possible, as there is exactly one appraiser (connected by a cable) and there is no opportunity for the target to participate in the attestation process.

CoPilot is a very specialized kind of attestation system, but it strangely does not take advantage of this. The discussion in Section 4.4 mentioned that *specialists* can be valuable because of their ability to make use of knowledge of the structure of the object being measured. CoPilot, even though it is only used to measure Linux kernels, does not perform structural analysis of data – it only reports hashes of kernel memory. As a result, changes are detected but the meaning of those changes is not feasible to determine. The way that CoPilot's specialized nature is implemented (via dedicated hardware) also means that supporting nested attestation is impossible.

The fact that CoPilot runs on add-in hardware may increase trust in the attestation mechanism and avoid impact on target execution, but at the cost of requiring extra hardware for every target system.

Nexus. Nexus [23] is an effort at Cornell to develop an operating system with particular attention to "active attestation." It enables separation via secure memory regions and moves device drivers into userspace. It introduces "labeling functions," a mechanism for providing dynamic runtime data to appraisers. Measurement tools may be sent to the target system by the appraiser and do not need to be pre-integrated with the base system.

As it involves an entirely new microkernel-based operating system, there are clearly adoption hurdles in the path of Nexus. It is perhaps most useful to think of Nexus not in the role of a guest operating system in our framework, but rather as something one might use for separation purposes instead of a traditional hypervisor. This relationship seems even more relevant in light of the fact that the Nexus project intends to be able to run Linux on top of a Nexus process.

Nexus is not yet released, but one can imagine it playing a part in a variant of our architecture. The ideas of fresh information, comprehensive information,

constrained disclosure, and a trustworthy mechanism would clearly resonate with the developers of Nexus. While it does not account for some of the elements in our architecture, it also does not appear to be contradictory with them. As this work emerges into public view it will be worth watching in order to determine how it might be used to satisfy attestation needs.

Property Based Attestation. Our concept of delegated attestation and appraisal is a more general variant of the idea known as *Property-based Attestation*. Jonathan Poritz [18] and Sadeghi and Stüble [19] have each pointed out that the end consumer of an attestation ought to care about security properties of the attester, as opposed to the specific mechanisms employed to achieve those properties. As should be clear from this paper, we strongly agree with this fundamental idea.

Poritz suggests that a solution might include virtualization and trusted third parties, but does not propose a concrete approach. Sadeghi and Stüble go farther, suggesting multiple approaches. Their abstract model of the TCG specifications takes an unusual view of the TPM Seal [25] functionality, which may impact the viability of the proposed solution. Some of their other suggestions for PBA are attractive but require significant changes to the TPM or the TCG software stack.

These authors and several others go on to propose a protocol for performing such property-based attestations [3]. This protocol could be implemented as an AP/ASP combination in our system, as long as some specialist appraiser for it was also implemented.

We go farther than the capabilities shown in work thus far, showing that there can be multiple layers of delegated attestation, that there can be arbitrarily many layers of the platform being appraised, and that the proper appraisers for each of these may be different. The data stored in a hardware TPM is not the only data for which one would wish to delegate the appraisal. Data in virtual TPMs, and even data stored at higher levels in the operating system of a host may be appropriate to delegate to specialists and describe via abstract, property-based attestations.

8 Conclusion

Attestation is an area which will see many technological innovations and developments in the near future. In particular, since the major vendors are introducing improved support for virtualized systems, architectures such as ours should be increasingly easy to implement in a trustworthy way. The semantic explicitness and freshness of the attestations that we propose should allow a common vocabulary across many architectures. Constrained disclosure should encourage systems owners to allow their systems to participate in attestations. Comprehensive information should encourage appraisers to place credence in well-supported claims, particularly given underlying trustworthy attestation mechanisms. We have attempted to clarify the way that existing work can be used to contribute to our goals.

References

1. B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, and H. Levkowetz. Extensible Authentication Protocol (EAP). RFC 3748 (Proposed Standard), June 2004.
2. Boris Balacheff, Liqun Chen, Siani Pearson (ed.), David Plaquin, and Graeme Proudler. *Trusted Computing Platforms: TCPA Technology in Context*. Prentice Hall PTR, Upper Saddle River, NJ, 2003.
3. Liqun Chen, Rainer Landfermann, Hans Löhr, Markus Rohe, Ahmad-Reza Sadeghi, and Christian Stüble. A protocol for property-based attestation. In *STC '06: Proceedings, First ACM Workshop on Scalable Trusted Computing*, pages 7–16, New York, NY, USA, 2006. ACM Press.
4. AMD Corporation. Amd64 architecture programmer's manual volume 2: System programming rev 3.11. http://www.amd.com/us-en/assets/content_type/white_papers_and_tech_docs/24593.pdf, January 2006.
5. Intel Corporation. Intel trusted execution technology. http://download.intel.com/technology/security/downloads/31516803.pdf, November 2006.
6. Microsoft Corporation. Ngscb official page. http://www.microsoft.com/resources/ngscb/ default.mspx, 2007.
7. David Grawrock. *The Intel Safer Computing Initiative*. Intel Press, 2006.
8. TCG Best Practices Group. *Design, Implementation, and Usage Principles for TPM-Based Platforms*, May 2005. Version 1.0.
9. Joshua D. Guttman. Authentication tests and disjoint encryption: a design method for security protocols. *Journal of Computer Security*, 12(3/4):409–433, 2004.
10. Joshua D. Guttman, Jonathan C. Herzog, John D. Ramsdell, and Brian T. Sniffen. Programming cryptographic protocols. In Rocco De Nicola and Davide Sangiorgi, editors, *Trust in Global Computing*, number 3705 in LNCS, pages 116–145. Springer, 2005.
11. Joshua D. Guttman, F. Javier Thayer, Jay A. Carlson, Jonathan C. Herzog, John D. Ramsdell, and Brian T. Sniffen. Trust management in strand spaces: A rely-guarantee method. In David Schmidt, editor, *Programming Languages and Systems: 13th European Symposium on Programming*, number 2986 in LNCS, pages 325–339. Springer, 2004.
12. Vivek Haldar, Deepak Chandra, and Michael Franz. Semantic remote attestation – a virtual machine directed approach to trusted computing. In *Proceedings of the Third virtual Machine Research and Technology Symposium*, pages 29–41. USENIX, May 2004.
13. Nick L. Petroni Jr., Timothy Fraser, Jesus Molina, and William A. Arbaugh. Copilot - a coprocessor-based kernel runtime integrity monitor. In *USENIX Security Symposium*, pages 179–194. USENIX, 2004.
14. Yasuharu Katsuno, Yuji Watanabe, Sachiko Yoshihama, Takuya Mishina, and Michiharu Kudoh. Layering negotiations for flexible attestation. In *STC '06: Proceedings, First ACM Workshop on Scalable Trusted Computing*, pages 17–20, New York, NY, USA, 2006. ACM Press.
15. Ross Kerber. Advanced tactic targeted grocer: 'Malware' stole Hannaford data. *The Boston Globe*, page 1, 18 March 2008.
16. P. Loscocco and S. Smalley. Integrating flexible support for security policies into the linux operating system. Technical report, NSA, NAI Labs, April 2001.
17. Jonathan Millen, Joshua Guttman, John Ramsdell, Justin Sheehy, and Brian Sniffen. Call by contract for cryptographic protocol. In *FCS-ARSPA*, 2006.

18. Jonathan A. Poritz. Trust[ed — in] computing, signed code and the heat death of the internet. In *SAC '06: Proceedings of the 2006 ACM symposium on Applied computing*, pages 1855–1859, New York, NY, USA, 2006. ACM Press.
19. Ahmad-Reza Sadeghi and Christian Stüble. Property-based attestation for computing platforms: caring about properties, not mechanisms. In *NSPW '04: Proceedings, 2004 Workshop on New Security Paradigms*, pages 67–77, New York, NY, USA, 2004. ACM Press.
20. Arvind Seshadri. Pioneer web page. http://www.cs.cmu.edu/
21. Arvind Seshadri, Mark Luk, Elaine Shi, Adrian Perrig, Leendert van Doorn, and Pradeep Khosla. Pioneer: Verifying integrity and guaranteeing execution of code on legacy platforms. In *Proceedings of ACM Symposium on Operating Systems Principles (SOSP)*, pages 1–16, October 2005.
22. Elaine Shi, Adrian Perrig, and Leendert Van Doorn. BIND: A time-of-use attestation service for secure distributed systems. In *Proceedings of IEEE Symposium on Security and Privacy*, May 2005.
23. Alan Shieh, Dan Williams, Emin Gün Sirer, and Fred B. Schneider. Nexus: a new operating system for trustworthy computing. In *SOSP '05: Proceedings of the twentieth ACM symposium on Operating systems principles*, pages 1–9, New York, NY, USA, 2005. ACM Press.
24. Brad Stone. 11 charged in theft of 41 million card numbers. *The New York Times*, page B 1, 5 August 2008.
25. Trusted Computing Group. *TPM Main Specification*, version 1.1b edition, 2001. https://www.trustedcomputinggroup.org/ downloads/tcg_spec_1_1b.zip.
26. Trusted Computing Group. *TCG Trusted Network Connect: TNC Architecture for Interoperability*, May 2006. Version 1.1.
27. Trusted Computing Group. *TCG Trusted Network Connect TNC IF-IMC*, May 2006. Version 1.1.