Attestation: Evidence and Trust

Justin Sheehy
MITRE Corporation
justin@mitre.org

George Coker
National Security Agency
gscoker@nsa.gov

Joshua Guttman
MITRE Corporation
guttman@mitre.org

Peter Loscocco
National Security Agency
loscocco@tycho.nsa.gov

Amy Herzog
MITRE Corporation
aherzog@mitre.org

Jon Millen
MITRE Corporation
jmillen@mitre.org

Leonard Monk
MITRE Corporation
lmonk@mitre.org

John Ramsdell
MITRE Corporation
ramsdell@mitre.org

Brian Sniffen
MITRE Corporation
bsniffen@mitre.org

Abstract

Attestation is the activity of making a claim about properties of a target by supplying evidence to an appraiser. An open-ended framework for attestation is desirable for safe support to sensitive or high-value activities on heterogeneous networks.

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 depend on several claims; and (v) the underlying attestation mechanism must be trustworthy.

We propose an architecture for attestation that is 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

A principal goal in trusted computing [2] is to provide a user, resource owner, or service provider with reliable knowledge about a platform. Through evaluation of the identity and integrity of a system, evidence is produced that the target will not engage in some class of misbehaviors. To this end, the Trusted Computing Group (TCG) has introduced the Trusted Platform Module (TPM) and the associated concepts of *system measurement* and *remote attestation*.

System measurement and remote attestation may be used to address a number of trust problems ranging from guaranteed invocation of software, delivery of premium content to trusted clients, assuaging mutual suspicion between clients, and more, with consideration of the assurance requirements of these varied applications. As the requirements of all such applications cannot be known *a priori*, attestation systems and measurement systems alike must be flexible, providing for privacy, completeness of measurement, and trust in the basic collection and reporting mechanisms.

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 desirable in those use-cases. For example, in [4], the author uses the term attestation to specifically mean the process of transmitting a sequence of hashes of certain system components and a digital signature of that sequence; in Microsoft's "NGSCB" [3] it refers to identification and authentication of known code via digital signatures; Copilot [9] 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.

Looking into the future, the trust desired in our systems will be far from monolithic. Users will seek attestation architectures that are flexible enough to accommodate all of these varying concepts of attestation and more. Also, each party to any system attestation will have their own desires regarding the measurement of their peers and themselves. A property that is sufficient for one party to use as evidence for a trust decision may not be for another; a set of information one user has no problem providing might cause privacy concerns for another.

These varying needs are directly driven by what each party has at stake. The more one has to lose by inappropriate disclosure or incomplete knowledge, the stricter one's needs will be regarding privacy or complete measurement. Parties with strong demands of each other regarding authentication and measurement may accordingly be willing to give up more of their own private information in order to achieve their goals.

In this paper, we put forth a flexible attestation architecture, along with definitions and guiding principles which have guided its development. Systems built according to this architecture can be composed to accomplish complex attestation scenarios and to provide more complete attestation than is currently achievable. We evaluate our architecture and other existing systems according to the principles we introduce.

2 Attestation

In this section we define the process of attestation and provide context for the understanding of security properties achievable via that process.

This 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, which has need to make a decision about some other party or parties. A* target *is a party about which an appraiser needs to make 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.

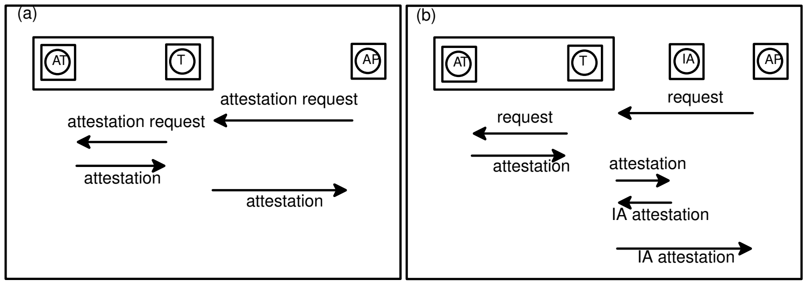

Figure 1: Attestation Scenarios

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 and local observation of it.*

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.

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 collected by a measurement tool, 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.

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 Attestation Scenarios

We introduce the notion of *scenarios* to describe exchanges between principals in attestation systems. These scenarios are used to illustrate security properties that are not explicit properties of the principals but rather the exchanges between them.

Scenarios in this sense are easily described. A simple example, abstracted over the exact content of the messages, is shown in Figure 1a. It contains a Target ('T'), an Attester ('AT') and an Appraiser ('AP'). The Target and Attester are drawn so as to demonstrate that they reside on the same physical host, but that there is a separation property causing them to be effectively different principals. The appraiser demands an attestation from the target. The target asks its attester to examine it and produce an attestation reflecting that examination. This attestation is given back to the target in a fashion that is resistant to tampering by the target, perhaps with a digital signature from AT. The target passes it along to the appraiser that caused it to be requested in the first place. In order to present this as a simple yet illustrative example of an attestation scenario, details such as the specific content of the request and response messages are omitted.

At first glance, this scenario might not seem like the simplest possible scenario for attestation. After all, what about a scenario with only two principals instead of three? The problem with that idea is that for an attestation to be believable, the measurements used to root the attestation must be produced by a *trustworthy mechanism*. One cannot simply ask a target about itself, as the whole purpose of the process is to determine whether or not to trust the target. To believe claims by a target would be begging the question regarding that target's trustworthiness, and thus would not be a worthwhile method for attestation. Thus it is not plausible to deal with attestation for making trust decisions without considering the examiner/attester as a party separate from the target.

The example in Figure 1a is so simple in shape as to not be very flexible. The attester is subject to the specific measurement requirements of the appraiser, and its only privacy-preserving option is to opt out entirely if it does not wish to exactly fulfill the request. Also, it is hard in a scenario of this simple shape for the attester to do much to verify the identity of the party that will receive its measurements.

If our target of attestation wishes for another party to make trust decisions about it without having to directly disclose measurements, this might be achieved by introducing a new party that is trusted by the target for privacy and is trusted by the receiver for integrity. This scenario is shown in Figure 1b. (Not shown here is how the trust in this party is established. It could be via another attestation, or any other means acceptable to the respective parties.)

This scenario is much the same as the first scenario, but also includes the Intermediate Attester ('IA'). In this case, the attestation from the target is sent to IA, which performs an appraisal and then produces a new attestation which can be sent to AP. The attestation sent to IA is likely to include detailed platform and host-specific measurements, while the one sent to AP may be as simple as an IA-signed and timestamped message effectively claiming that T has some given abstract property that AP cares about. Not only does this solve a privacy problem from T's point of view, it means that parties such as AP that wish to make trust decisions do not need to know all of the individual details necessary to perform the right concrete local measurements on all remote parties that they may wish to trust. This concept of "nesting" attestations is a very powerful idea which can be used to introduce a great deal of flexibility in an attestation system. The SSL certificate authority mentioned in Section 2 is an example of nested attestation. It is much like our IA party here, but it happens to operate offline.

The view used here for scenarios and trust management abstracts away from the specifics of the properties desired and achieved by principals. That abstraction allows for easy development and analysis of a small number of flexible attestation scenarios to cover most common cases. That said, these scenarios are just intended to be illustrative examples, not to make up a canon of attestation interactions.

The ability to compose scenarios and measurement services in a flexible way to meet dynamic needs is more valuable from the point of view of attestation architecture than having a specific known set of possible known interactions. Trust engineering can become very complex, and that complexity is much worse to manage if one must start afresh for each situation instead of just finding the right set of pre-existing building blocks and composing them.

The meaning of attestation and some of the shapes of the motivating scenarios have been explored. What follows is an attempt to determine some general architectural principles. A system designer should strive to satisfy these in order to achieve the properties desirable to all of the possible parties involved in attestation.

4 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 5.

Principle 1 (Fresh information) Information about the target should reflect the *running system*, rather than just images stored on disk. While some measurement tools may collect and deliver start-up time information about the target, others will need to inspect the current state of an active target. An attestation architecture should ensure that such tools have 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) The 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) The target should be able to enforce a policy governing which measurements of itself are sent to any given appraiser. Hence, an attestation architecture must allow the appraiser to be identified to the target.

This policy may distinguish kinds of information that may be delivered to different appraisers. The policy may even be dynamic, relying on current run-time information for individual disclosure decisions.

For instance, a decision to disclose information may require that the appraiser provide an attestation to the target about its own state. □

Principle 4 (Semantic explicitness) The semantic content of attestations should be explicit. The appraiser should be able to infer consequences from a number of different attestations, e.g. when several measurements are needed in order to make a prediction about the behavior of the target. Hence, the semantics of individual attestations should be uniform, and they should be composable using valid logical inferences.

The identity of the target should be explicitly represented in a form that the appraiser can use to determine whether a number of different attestations concern the same target. □

Principle 5 (Trustworthy mechanism) The attestation mechanism itself should be able to provide the appraiser with evidence that individual attestations are reliable. In particular, the attestation architecture being used 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.

Based on these principles, various contraints emerge that lead to an attestation architecture that could assist in their achievement.

5 Proposed Attestation Architecture

There are five main constraints, imposed by the principles of Section 4, 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.

5.1 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.

5.2 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. 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 8.3) 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.

5.3 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 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 would be 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 7. One possible alternative is that a hardware component, part of our self-protecting trust base, provides runtime measurements of the domain separation mechanism.

5.4 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. 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 [6], and may use the strand space/trust management connection from [8, 7].

5.5 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.

One mechanism which can be used to implement both aspects of Attestation Management is Call by Contract, as explained in [11].

In order to defend this capability from both the untrusted target of attestations and also from potentially-vulnerable measurement tools, it 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.

5.6 The elements of the architecture

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

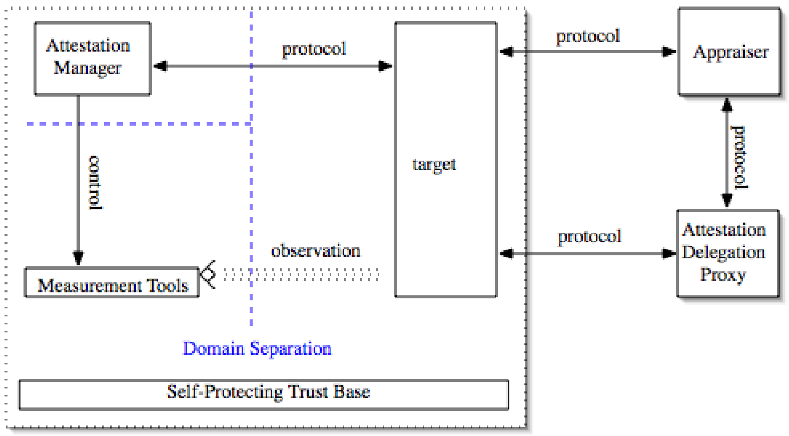

Figure 2: Attestation Architecture

6 Trusted Research Platform

An implementation of our attestation architecture has been developed. An illustration of this platform is shown in Figure 3. The hypervisor is represented abstractly, as the implementation as described here is not tied to features specific to any one virtual machine monitor. The Supervisor guest (S guest) contains platform support, while the User guest (U guest) runs the user's "normal" operating system. The TPM hardware resource has been virtualized ("vTPM") for the S and U environments. Both environments possess measurement and attestation capabilities.

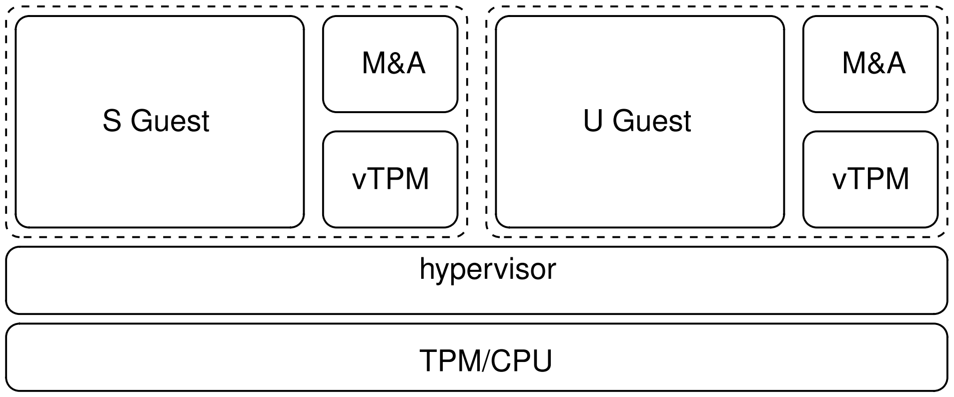

Figure 3: Trusted Research Platform

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

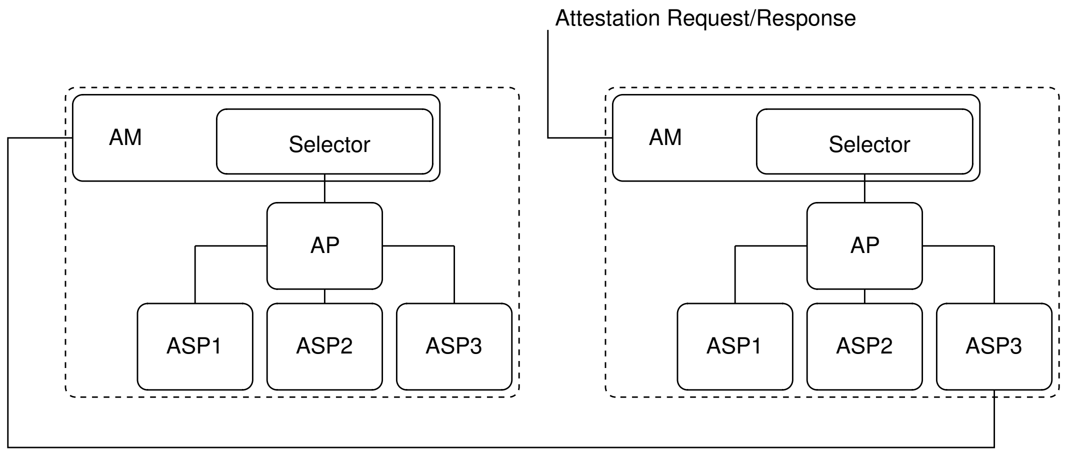

Figure 4: M&A

illustrative example, look back to Figure 1 and imagine the U M&A acting for the “T” role, and the S M&A acting in the “AT” role.

To manage a diversity of measurement and attestation requirements, a virtual machine is dedicated to measurement and attestation (M&A) of the guest. The hypervisor is integral to the trust base for the system, controlling sharing and providing domain separation. Additional domain separation and controlled sharing is obtained by instrumenting the M&A on SELinux [10]. The separation of the M&A and guest environments is transparent to the platform and the attestation. Upon receiving attestation requests, the guest will direct them 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. Some example ASPs are integrity measurement systems, wrappers for calls to a TPM/vTPM, or invocation of other services. 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. The implementation uses a method referred to as “Call by Contract” [11] for the Selector.

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 4 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.

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 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.

7 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 and AMD 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. Specifically required is a means to establish domain separation in order to support a trustworthy mechanism for attestation. The current implementation uses an off-the-shelf virtualization system – but none of those available today offer the balance needed 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 operating 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.

8 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 [16] [17] [5], Microsoft [3], and multiple independent researchers [9] [13]. 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.

8.1 Trusted Network Connect

Trusted Network Connect (TNC) is a specification from the Trusted Computing Group [17] 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 [17], 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 [18]. 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 [18]. 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" [17] 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 both 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.

8.2 Pioneer and BIND

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

BIND [14] is a runtime code attestation service intended for use in securing distributed systems. The basic approach of BIND centers around a specific integrated measurement capability which binds a proof of process integrity to the data produced by that process. For embedded special-purpose systems which will not have flexible attestation needs, an approach like BIND may be useful.

Pioneer [13] is an attempt to provide a "first-step toward externally-verifiable code execution on legacy computing systems." In this context, legacy means systems without any hardware trust base – Pioneer attempts to solve the attestation problem entirely in software. This approach 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 of the target as well as the memory latency. The system must not be overclocked, it must not support symmetric multi-threading, and it must not generate system management interrupts during the execution of Pioneer. In general, this level of dependency suggests that an attacker with sufficient understanding of the hardware might subvert the system. In specific, at least one such subversion is known in the case of systems with 64-bit extensions. The specific weaknesses referred to are publicly acknowledged by the authors as implementation issues [12].

Another requirement for proper results from Pioneer is that the checksum code must be known to be the most time-optimal possible code that performs its checksum. No proofs of such optimality for any Pioneer-sufficient checksum functions are known to exist. It remains to be seen if something like Pioneer can succeed in the long run, as newly emerging hardware optimizations will continue to provide new 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.

8.3 Copilot

CoPilot [9] is a measurement and attestation system that is used to detect root kits in a Linux kernel. CoPilot works by periodically computing static hashes over key parts of the kernel memory that impact kernel execution, comparing them against a known baseline, and then reporting results to an administration system that enables manual decisions to be made regarding the acceptability of any detected changes. CoPilot runs on a PCI add-in card, accessing system memory using DMA. It uses a dedicated port on the card to communicate with the appraiser, although there exists the possibility of using an encrypted channel through the host in future versions.

CoPilot performs fairly well with respect to some of the attestation principles introduced earlier. The system is well protected due to existing in a separate hardware environment and its direct connection to the appraiser. It produces fresh information about the running system that can be refreshed over time. The measurements it produces are hashes of the text sections of the kernel and any loaded modules as well as hashes of some kernel data.

CoPilot is an advance in attestation technology, but it still has limitations. Although its measurements are as comprehensive as can be expected from a system that relies on hashing, it does not provide a true 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 truly associate any of its measurements with what is actually running on the processor. Also, the timing of measurement cannot be coordinated with the actions on the target. This means that appraisal is difficult as any given measurement might be taken during a transitory moment of inconsistency. Achieving any kind 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 5.4 mentioned that *specialists* can be valuable because of their ability to make use of knowledge of the structure of the object being measured. However, CoPilot (even though it is only ever used to measure Linux kernels) does not perform any structural analysis of data – it only reports hashes of kernel memory. As a result, changes are easily 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 our notion of nested attestation is impossible.

The fact that CoPilot runs on add-in hardware may increase trust in the attestation mechanism and does not directly impact target execution, but comes at the cost of requiring extra hardware for every target system.

8.4 Nexus

Nexus [15] is an effort at Cornell University to develop an operating system for trustworthy computing, with particular attention to the new idea of "active attestation." It makes heavy use of separation via secure memory regions and moving device drivers into userspace. It introduces the notion of "labeling functions," a mechanism for providing measurement tools with dynamic runtime data. Additionally, these 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 are clearly ones that would 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.

9 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] Microsoft Corporation. Next-generation secure computing base official page. http://www.microsoft.com/resources/ngscb/default.mspx, 2007.

[4] David Grawrock. *The Intel Safer Computing Initiative*. Intel Press, 2006.

[5] TCG Best Practices Group. *Design, Implementation, and Usage Principles for TPM-Based Platforms*. Trusted Computing Group, May 2005. Version 1.0.

[6] Joshua D. Guttman. Authentication tests and disjoint encryption: a design method for security protocols. *Journal of Computer Security*, 12(3/4):409–433, 2004.

[7] 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.

[8] 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.

[9] 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.

[10] P. Loscocco and S. Smalley. Integrating flexible support for security policies into the linux operating system. Technical report, NSA, NAI Labs, April 2001.

[11] Jonathan Millen, Joshua Guttman, John Ramsdell, Justin Sheehy, and Brian Sniffen. Call by contract for cryptographic protocol. In *FCS-ARSPA*, 2006.

[12] Arvind Seshadri. Pioneer web page. http://www.cs.cmu.edu/

[13] 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.

[14] 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.

[15] 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.

[16] Trusted Computing Group. *Main Specification*, version 1.1b edition, 2001. https://www.trustedcomputinggroup.org/downloads/tcg_spec_1_1b.zip.

[17] Trusted Computing Group. *TCG Trusted Network Connect: TNC Architecture for Interoperability*, May 2006. Version 1.1.

[18] Trusted Computing Group. *TCG Trusted Network Connect TNC IF-IMC*, May 2006. Version 1.1.