A High-assurance, Virtual Guard Architecture

Mark R. Heckman, Roger R. Schell, Edwards E. Reed
Aesec Global Services
Palo Alto, CA, USA
{mark.heckman, roger.schell, ed.reed}@aesec.com

***Abstract* — Although one senior security professional has emphasized that "it is unconscionable to use overly weak components" in a multilevel security (MLS) context, the majority of current transfer guards do exactly that. Basic guard technology is well-developed and has a long history, but most guards are built on low-assurance systems vulnerable to software subversion, and the lack of assurance limits the range of transfers. This paper describes a virtual guard architecture that leverages mature MLS technology previously certified and deployed across domains from TS/SCI to Unclassified. The architecture permits a single guard system to simultaneously and securely support many different transfer functions between many different domain pairs. Not only does this architecture substantially address software subversion, support adaptable information transfer policies, and have the potential to dramatically reduce (re)certification effort, the virtualized guard execution environment also promises to significantly enhance efficient and scalable use of resources.**

***Index Terms*— Assured pipeline, Downgrading, GEMSOS, Guard, High-assurance, Multilevel security, Sanitization, Virtualization**

I. Introduction

The Committee on National Security Systems defines a "Cross Domain Service (CDS)" as "a form of controlled interface that provides the ability to manually and/or automatically access and/or transfer information between different security domains [1]." The Unified Cross Domain (CD) Management Office (UCDMO) has further categorized CD mechanisms as *transfer*, *access*, and *multilevel.* "A *transfer* device permits the movement of data from one domain to another. An *access* device allows a user to sit on one workstation and access multiple domains but not move data between them. A *multilevel* device stores and processes information of different security levels in a common repository, but only allows a user to view appropriate information based on his/her credentials [5]."

In this paper, we are chiefly concerned with transfer devices. Transfer devices transfer from one domain information that is authorized for a second domain, ensuring that the authorized information and only the authorized information is transferred to the second domain. Examples of data transfer functions include the following:

- Anti-virus scans on information transferred from a low domain to a high domain, to protect the integrity of the high domain.
- "Dirty word" (specific content) searches on data being transferred from a high domain to a low domain, to prevent the leakage of sensitive data.
- Creating "sanitized" data releasable to a low domain out of sensitive data in a high domain. For example, summary U.S. census data may be released soon after a census, but the raw data cannot legally be released to the public domain for 72 years [2].

(Note that the latter two types of policies often require human review before information can be downgraded to the low domain, due to the inability of a computer algorithm in many cases to reliably examine data and determine that it contains no sensitive information [3].) We refer in this paper to transfer devices that implement any of these types of policies as transfer "guards".

The UCDMO maintains a "Baseline List" of commercially-available CDSs that are available for deployment by U.S. Government agencies. This list, as of 27 January 2012, has 28 entries, 19 of which are of the "Transfer" type [6]. Although offered by diverse vendors, a cursory examination of the transfer device products in the UCDMO Baseline List indicates that they likely share several weaknesses. In particular, most of these transfer guards are based on low-assurance, commodity technology that is liable to introduce a high risk of information compromise due to software subversion. The low-assurance systems necessitate inflexible and highly constrained transfer policies, demand endlessly repetitive and tedious certification and recertification of uncertain effectiveness, and lead to a large amount of wasted and duplicative resources.

The Aesec Virtual Guard (AVG) architecture introduced in this paper leverages many years of science and engineering experience with building highly secure systems [18]. These techniques were systematically codified in the U.S. National Security Agency's "Trusted Computer System Evaluation Criteria" (TCSEC, also known as the "Orange Book") [27]. Two important system composition techniques, Partitioned TCBs [23] and TCB Subsets [21], were later approved to help simplify the evaluation of complex systems, The potential to apply these techniques was largely carried forward in the more recent "Common Criteria for Information Technology Security Evaluation" (CC) [28].

Through the application of the science of knowing how to build high-assurance components and how to compose them, the AVG architecture avoids or improves on the serious

978-1-4673-3/12/$31.00 ©2013 IEEE

limitations of current guard technology.

II. Limitations of Current Guard Technology

The high-assurance AVG, built on a trusted computing base (TCB) that was evaluated at the highest level of the TCSEC, Class A1 (meeting CC EAL7 equivalent requirements), addresses each of the following weaknesses identified in current guard technology.

A. Susceptibility to Software Subversion

Guard implementations are commonly software applications hosted on an operating system and hardware platform. But the hosts typically are not high-assurance and, hence, are susceptible to software subversion. Addressing the threat of software subversion requires verifiable protection as systematically codified, for example, in the TCSEC's Class A1, which is distinguished by "substantially dealing with the problems of subversion of security mechanism" [7], but the commodity platforms typically used for guards have few of the protection features of a true high-assurance system.

A current, well-known guard example is the Information Support Server Environment (ISSE) [8]. Vendor literature for one recent (added to the Baseline List 2 April 2012 [9]) commercial instantiation of the ISSE says, "Guard software will execute on the Trusted Solaris multi-level, secure operating system" [10]. Trusted Solaris provides, at best, a very weak level of protection. In fact, it has been used as a specific example of an operating system that is "inadequate to counter any focused attack" [18].

Another transfer CDS in the UCDMO Baseline List runs on "commodity commercial off-the-shelf servers running Red Hat Enterprise Linux 5 with a Strict SELinux policy [11]", and SELinux has also been proposed as a suitable base for guards by others [12][13], but the NSA says that "Security-enhanced Linux is only intended to demonstrate mandatory controls in a modern operating system like Linux and thus is very unlikely by itself to meet any interesting definition of secure system" [14]. A system is only as secure as its weakest link. A guard built on a low-assurance system has a foundation of sand.

Proposed guards based on the separation kernel-based MILS architecture are a recent development [15][16]. MILS is intended to enable the creation of high-assurance systems by composing untrusted, commercial, off-the-shelf (COTS) components [17]. Separation kernel approaches, by definition, however, do not include a Reference Monitor, so the security policy enforcement mechanism is diffused throughout the system and development and certification of high-assurance systems is potentially difficult and risky.

Current guard implementations are typically missing technology to provide high-assurance that only approved and authenticated software is distributed to each guard installation. Because the enforcement of a mandatory access control (MAC) policy is the responsibility of the underlying operating system TCB, the TCB must be responsible for trusted distribution, system integrity, and system recovery. The TCB needs to implement integrity-checking mechanisms on system software, detecting any tampering and preventing a damaged system from running.

Additional constraints on guard operations and management arise from the lack of integrity features for application software, configuration information, and data. In summary, current guard technology fails to significantly mitigate the threat of supply-chain subversion of the trusted system and untrusted applications software.

A determined and skilled adversary can easily embed complex subversions with little fear of detection and there is evidence that software subversion is the external threat with the highest potential payoff for determined adversaries, constituting the attack "of choice" [19]. (Recent events, such as the spread of Stuxnet and Flame malware, provide additional empirical support for this view [29].) As David Bell has written, since guards, intrinsically "are multilevel, it is unconscionable to use overly weak components. Such connections require high security, meaning A1 [18]."

B. Rigid Constraints on Transfer Security Policies

Implementations of the current guard technology have generally evolved in response to a particular class of environments with monolithic transfer security policies. Yet this approach neglects distinct, underlying policies, which tend to be lumped together in an ad hoc manner.

For example, for the U. S. Government, the underlying executive order for protecting classified information can be directly modeled as a MAC policy for the TCB of the underlying platform. As codified in the Class A1 criteria, the TCB can employ installation-specific configuration adaptations for diverse contexts from the U.S. government to foreign partners to commercial enterprises. On top of this MAC policy, for a given set of security domains there may be a policy on cross domain transfers, for example a "dirty word search" or human review. Furthermore, on top of those two policies, a given installation may have further "safety" policy constraints on information that can flow between domains, e.g., requiring virus checking on data before it is transferred.

Current guard technology does not employ a systematic, scientifically sound policy composition architecture that uses the proven techniques of Partitioned TCBs [23] and TCB Subsets [21] that have been available for several decades.

Current guard technology also generally lacks the ability to limit the range of trust for a guard (e.g., TSABI versus SABI) in the underlying MAC TCB. The absence of this capability constrains deployment because there little assurance that the limited range of trust cannot be circumvented.

C. Inflexible Certification and Accreditation Support

Real-world requirements call for constant adaptation – new hardware, new software, new transfer criteria – against adversaries who are continually probing for vulnerabilities, but current guard technology does not have proven invariant protection properties and implementations are generally accredited for only one configuration. This means that guard implementations must be reevaluated for each variation,

because introducing variations breaks the assumptions that formed the basis for deployment approval.

D. Resource Intensive Replication

Without a high-assurance platform, guards cannot themselves be sufficiently trusted to share their operating system and hardware platform with other guards. This means that there must be a different platform for each guard and domain. In a multi-domain environment, the lack of high-assurance leads to the creation of guard server "farms", using duplicative and isolated hardware and software.

This wasteful duplication of resources is exacerbated by platforms with limited scalability for the complementary class of environments requiring high performance for data intensive transfers through the guard. A basic tool for scalability is multiprocessor support, where processing resources can be added for any deployment without impacting the underlying basis for security, but such scalable multiprocessing is a challenge that few high-assurance platforms have been able to meet.

III. GEMSOS, A HIGH-ASSURANCE TCB

The AVG is built on a high-assurance TCB - the COTS Gemini Secure Operating System (GEMSOS) [24]. The NSA has previously evaluated the GEMSOS security kernel and product Ratings Maintenance Phase (RAMP) plan at Class A1 as part of the evaluation of the Gemini Trusted Network Processor (GTNP) [25], confirming that it meets the highest standards for security, protection against subversion, and certifiability. Because the AVG architecture depends on features of the underlying TCB, we briefly describe some of the key features of GEMSOS here.

A. Access Control Policy

GEMSOS is a real-time, multi-processing operating system that implements a mandatory access control policy based on the hierarchical lattice of security labels described in the Bell and LaPadula access control model [33]. Labels include both a secrecy component and an integrity component (viz., strict integrity based on the Biba interpretation of the Bell-LaPadula model [32]). Each of the secrecy and integrity components consists of a hierarchical level and a set of non-hierarchical categories [34].

All objects (storage segments, synchronization objects, and other types of protected resources) have permanent access labels that the system attaches to the objects when the objects are created. Subjects (effectively processes, for purposes of this discussion), have two access labels – a maximum *read* label and a minimum *write* label. When a subject attempts to access an object, the TCB compares the subject's access labels to the object's access label. A subject's read label must dominate an object's access label in order for the subject to be granted read access. An object's access label must dominate the subject's write label in order for the subject to be granted write access [25].

Subjects whose read and write labels are equal are single-level, *untrusted* subjects. Subjects whose read and write labels are not equal are multi-level and *trusted* within the range defined by their read and write labels. The Final Evaluation Report for the GTNP notes that, although the GTNP evaluation addressed only single-level application subjects, GEMSOS correctly "restricts multi-level subjects to operating within the defined range of the subject [25]." The AVG architecture depends on this multi-level subject feature of GEMSOS to create guards.

Because the TCB is "the totality of protection mechanisms … responsible for enforcing a computer security policy [27]," trusted subjects that extend the security policy enforced by GEMSOS, such as transfer guards, must be evaluated as part of the TCB. Most typical operating system features, however, such as a file system [22], can be implemented with single-level subjects.

B. High-assurance features

GEMSOS is "high-assurance" due to a high level of trust that the protection mechanisms of the system correctly enforce the security policy during all phases of the system lifecycle. This trust is engendered by eight factors: system architecture (using techniques such as hardware segmentation, layering, information hiding, and minimization), integrity testing, covert channel analysis, trusted recovery, security testing, formal design specification and verification, configuration management, and trusted distribution [25].

In particular, the following Class A1 requirements satisfied by the GEMSOS TCB specifically address the threat of subversion [27]:

- Strict configuration management and special safeguards must be used to protect master copies of all material, including tools, used to generate the TCB. (Helps avoid, for example, the famous "Thompson Trojan compiler" problem [4].)
- Formal methods must be used to detect and analyze covert channels. (Confines potential "Trojan Horses".)
- Design documents must include a clear description of internal TCB mechanisms that are not described in the Formal Top Level Specification (FTLS).
- The Descriptive Top Level Specification (DTLS) and FTLS must include descriptions of hardware and firmware components (such as segmentation), if properties of those components are visible at the TCB interface, and a mapping of the FTLS to the actual source code must be performed. (Mitigates potential "Trap Doors".)
- Testing must demonstrate that the TCB implementation is consistent with the FTLS. Mapping of the FTLS to the source code can serve as the basis for precise penetration testing.

The GEMSOS kernel runs on the Intel IA-32 architecture, which provides segmentation hardware with four privilege levels [20]. The kernel executes in the highest privilege level to protect it from tampering. GEMSOS leverages the remaining three hardware privilege levels to create eight

classical protection rings and gates between them. The hardware segmentation mechanisms are foundational to enforcement of access controls. Rigorous software engineering principles (layering, in particular) and formal specification techniques used to develop the kernel, plus the kernel's compact size, support Class A1-level testing and formal analysis to show the correspondence of the implemented code to the FTLS and security policy model [25]. The system firmware (BIOS) is part of the TCB and is also custom-built with attention to security-related software engineering.

GEMSOS maintains system integrity beginning with secure configuration management and continuing with trusted distribution of the system hardware and software. Kernel software is distributed on encrypted and sealed volumes. The crypto-seals protect software from alteration during shipping, installation, and system operation. Similarly the custom GEMSOS BIOS is protected by a crypto-seal authenticator based on a checksum computed on the ROM contents [25].

At boot time, the system first executes diagnostic tests to verify the integrity and correct operation of the hardware and firmware. Secure booting continues by verifying the integrity of the boot volume and kernel modules using cryptographic checksums before transferring control to the kernel. The system shuts itself down if a problem is detected at any point. Special, off-line Trusted Recovery procedures must be followed before a system can resume operation. The cryptographic checksum feature can be used to protect any system, application, or data files from corruption or tampering.

IV. AVG Architecture

The AVG architecture is implemented through GEMSOS-provided process isolation, GEMSOS-supported multi-level subjects, and assured pipelines created using GEMSOS mandatory access class labels. Network clients communicate with the AVG through standard protocols such as Network File System (NFS) [35]. The processes that implement the communications protocols (equivalent to Unix "daemons") are single-level processes outside the TCB. The AVG architecture is depicted in figure 1.

A. Assured Pipelines

An assured pipeline limits communication within a sequence of processes so that each process in the pipeline can only receive information from the previous process and send information to the next process [30]. Processes outside the pipeline cannot interfere with data in the pipeline. Assured pipelines have been a feature in other guard designs as well as the AVG, although on low-assurance operating systems [12][13].

Assured pipelines in the AVG are implemented using integrity categories [31], which are part of the integrity component of the mandatory security labels assigned to every subject and object managed by the TCB [24]. Each guard has a unique integrity category, called the *guard identifier*. The guard identifier protects the guard from outside processes because the lattice-based MAC policy enforced by GEMSOS requires that the write label of subjects contain an integrity category in order to be able to modify objects that have that same integrity category [34]. Only the processes in a guard, however, have the guard identifier integrity category assigned to that guard. Additional integrity categories are used to keep the pipelines of the guard ordered and separate.

For example, in figure 1, the Input Queue Manager is trusted within a very specific integrity range so that it can read from the Input Message Queue, which has a secrecy level of "High" and an integrity category "ic1", and write to the High Message Buffer, which has a secrecy level of "High" and two integrity categories: "ic1" and "ic2". This is an example of an assured pipeline. Integrity category "ic1" is the guard identifier, while "ic2" is used to implement the assured pipeline, because only the Input Queue Manager has access labels that permit writing to the High Message Buffer. (Technically, the Trusted Guard Downgrade Function also has sufficient privilege to write to the High Message Buffer, although there is no functional reason for it to do so and, if it did, there are no security ramifications.) There is a second assured pipeline on the output side of the guard.

The use of guard identifier integrity categories and assured pipelines means that there can be multiple guards with different ranges on the same host system. Even trusted processes that are part of different guards cannot interfere with one another.

B. Trusted Subjects

The AVG architecture uses the multi-level, trusted subject feature of GEMSOS in two ways:

1. The process that performs the downgrade is trusted with respect to secrecy by the high source domain to maintain secrecy in the transfer to the low destination. For example, a sanitizer is "trusted" to remove sensitive information before putting data into the low destination domain. The range of trust of the downgrade process is explicitly limited by the GEMSOS-enforced labels assigned when the system is configured.
2. Assured pipeline processes are trusted with respect to integrity to create higher integrity results. In strict Biba integrity, low-integrity data cannot flow to a higher-integrity domain. Assured pipeline processes, however, read from a lower-integrity domain and write to a higher-integrity domain (viz., a domain with an additional integrity category), which protects the pipeline against modification.

C. AVG Design

The AVG implements a guard in three phases: input, release, and output.

1) Guard Input

In figure 1, a network client on the High network requests sanitized transfer of a message through the guard. In our initial prototype, this is implemented by copying a file containing the message from its local file system to the High Message Queue

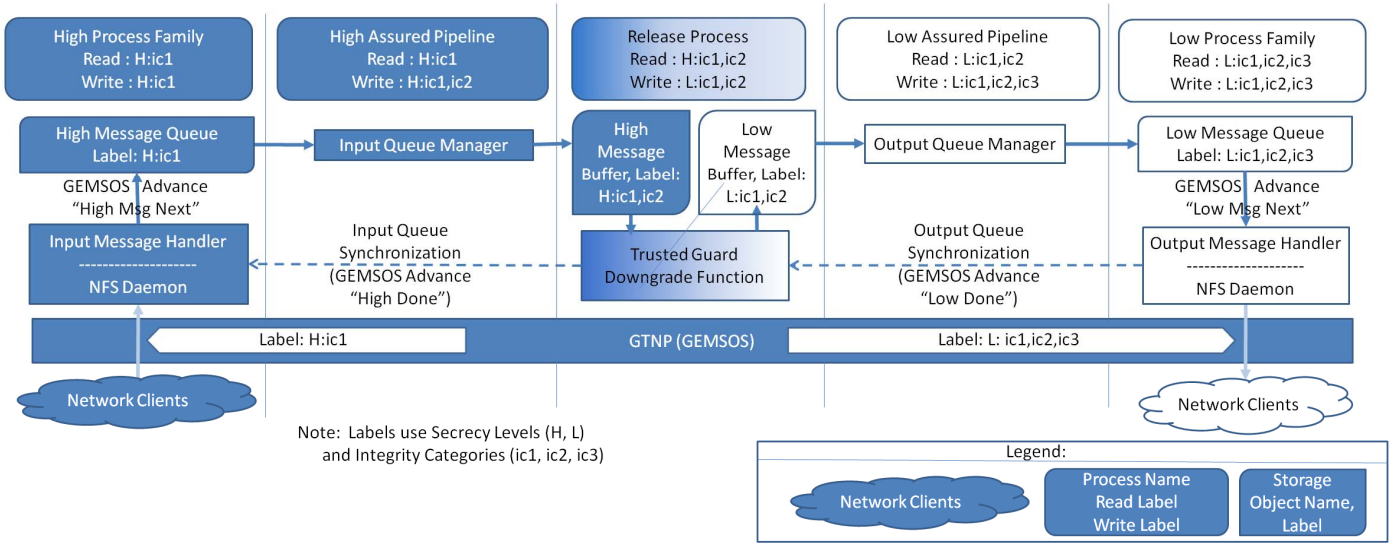

Fig. 1. Aesec Virtual Guard Architecture

– a directory on the AVG system mounted by the client using the Network File System (NFS) protocol [35].

The High Message Queue has a secrecy level of "High", but also an integrity category "ic1". The "ic1" integrity category is the guard identifier that is unique to this guard, and every subject and object in the guard has the same guard identifier.

The actual transfer of the file from the client is accomplished by the single-level Input Message Handler process (which effectively serves as the NFS daemon for the High network). The Input Message Handler has the same access class ("High" secrecy and integrity category "ic1") as the High Message Queue so it can write to the Queue.

The Input Message Handler informs the Input Queue Manager that a message is in the Input Message Queue by incrementing an eventcount – a type of secure synchronization object implemented in GEMSOS [36]. The operation to increment an eventcount is called "advance". The eventcount has the same access class as the Input Message Queue, so it can be read by the Input Queue Manager. Upon initialization, the Input Queue Manager reads the current eventcount value associated with the Input Message Queue and then blocks while it waits for the eventcount to be incremented. When the Input Message Handler increments the eventcount, the Input Queue Manager wakes up so it can process a message in the queue.

The Input Queue Manager process transfers each message from the High Message Queue to a High Message Buffer that it shares with the Trusted Guard Downgrade Function process. The Input Queue Manager is an assured pipeline trusted within a very specific integrity range so that it can read from the High Message Queue, which has a secrecy level of "High" and integrity category: "ic1", and write to the High Message Buffer, which has a secrecy level of "High" and integrity categories: "ic1" and "ic2".

2) Guard Release

The Trusted Guard Downgrade Function process is trusted within a secrecy range "High" to "Low". The installation and configuration of the guard creates this limited downgrade range, and the underlying TCB enforces it with high assurance, regardless of any attempts to exceed that range that might occur in the downgrade function.

The Downgrade Function can read and process input messages from the High Message Buffer and write downgraded data to the Low Message Buffer, which has secrecy level "Low" and the same two integrity categories ("ic1" and "ic2") as the High Message Buffer. The use of the two restricted integrity categories ensures that the Downgrade Function can only read from and write to the message buffers, but nowhere else.

The Input Queue Manager informs the Trusted Guard Downgrade Function that a message is in the High Message Buffer by incrementing an eventcount associated with the buffer. The eventcount has the same access class as the High Message Buffer, so it can be read by the Trusted Guard Downgrade Function. When the Trusted Guard Downgrade Function has processed the message in the High Message Buffer, so that the buffer is ready to accept another message, it signals the Input Queue Manager using another eventcount.

The Trusted Guard Downgrade Function performs sanitization processing on the message and writes the results to the Low Message Buffer. It then signals the Output Queue Manager through an eventcount. The Low Message Buffer has a secrecy level of "Low" and, from this point on, all processing in the guard is done at the "Low" secrecy level.

3) Guard Output

The Output Queue Manager is an assured pipeline that can read from the Low Message Buffer, which has a secrecy level of "Low" and integrity categories: "ic1" and "ic2", and write to the Low Message Queue, which has a secrecy level of "Low" and integrity categories: "ic1", "ic2", and "ic3".

The Output Queue Manager copies messages from the Low Message Buffer and puts them in files in the Low Message Queue. Using eventcounts, the Output Queue Manager tells the Trusted Guard Downgrade Function that the Low Message Buffer has been emptied and tells the Output Message Handler that a message has been added to the Low Message Queue.

The secrecy level of "Low" and the set of integrity categories "ic1" and "ic2" ensure that only the Output Queue Manager (and the Trusted Guard Downgrade Function) can read from the Low Message Buffer. The integrity categories of "ic1", "ic2", and "ic3" ensure that only the Output Queue Manager (and the Output Message Handler) can write to the Low Message Queue.

Clients on the Low network can retrieve "Low" files containing sanitized data from the AVG system via an NFS-mounted directory. The NFS file transfer is accomplished by the single-level Output Message Handler.

D. Pre- and Post-processing Ability

An important advantage of the assured pipeline architecture is the ability to add pre-processing and post-processing steps. By using additional integrity categories, additional pipelines can be added to the guard. An example of a pre-processing step is verification of a digital signature on the data. An example post-processing step could be an anti-virus scan.

V. AVG Principles of Operation

The AVG architecture described in section IV is designed to address all four of the limitations of current guard technology described in section II: susceptibility to software subversion, constraints on transfer security policies, inflexible certification and accreditation, and replication of resources.

A. Substantially address software subversion

As described in section III, the GEMSOS TCB on which the AVG is built meets TCSEC Class A1 standards for high-assurance, including formal analysis, configuration management and trusted distribution. Cryptographic checksums and other mechanisms are used to protect system, application, and data files from corruption and tampering. Section III identifies these and other high-assurance features that help assure the integrity of the system throughout its life-cycle, from design and development to distribution and operation, and significantly mitigate the threat of supply-chain subversion of the trusted system and application software.

B. Modular but confined transfer security policies

Unlike current guard technology, which lacks a high-assurance basis to ensure limits on the range of trust for a guard, MAC enforcement by the GEMSOS TCB provides verifiable assurance in the AVG that the range of trust of the guard cannot be circumvented. Specifically, as described in Section III, each subject has two access labels – a maximum *read* label and a minimum *write* label. Subjects whose read and write labels are not equal are multi-level and *trusted* to transfer information within the range defined by the read and write labels. The transfer of information from a source domain to a different destination domain can only be accomplished by such a trusted subject. GEMSOS ensures that a multi-level subject cannot exceed its defined range.

Moreover, the policy implemented by the guard is modular, and additional pre- and post-processing modules can easily be added simply by using additional integrity categories to implement assured pipelines. The pipelines ensure that the modules are organized as a hierarchy, so that no step can be skipped. Current guard technology is not designed to compose multiple, modular policies of this type in a systematic, scientifically sound manner, but the AVG, built on a high-assurance TCB, is.

Consider, for example, an AVG system with a single module. The guard policy is a hierarchical extension to the mandatory security policy enforced by GEMSOS because the trusted guard can only enforce its policy on objects where the TCB has previously enforced its own policy. The policy enforced by the entire system is, therefore, a composition of the TCB MAC policy, which permits trusted subjects, and the downgrade policy implemented by the trusted subject guard.

The TCB and guard each enforce a subset of the overall system security policy. A technique called "TCB Subsets" can be used to validate the correct composition of hierarchical components like the AVG and the TCB on which it runs [21]. The conditions under which a compositional evaluation using TCB subsets can be performed include clear identification of, and policy allocation between, the subsets and a definite hierarchical relationship between the subsets. An additional requirement is to demonstrate that the guard subset is protected from tampering by other subjects running on the TCB [21]. All of these conditions are met in the AVG architecture. This compositional evaluation process can be extended for an arbitrary number of additional modules connected via assured pipelines, leveraging composability to simplify certification and accreditation.

A TCB "partition" is a TCB subset that does not depend hierarchically on another TCB subset. For example, a set of unrelated guards hosted on the same AVG system implement an overall system policy, but those policies are not hierarchical. Instead, each guard enforces a *partition* of the overall policy. Similarly, a network of guards that run on different AVG systems but that are pipelined in some fashion are also not hierarchically related because they do not share a defined subset of subjects, objects, and hardware. Although their policies cannot be composed using the technique of TCB Subsets, "Partitioned TCB" composition can be used [23].

C. Incremental evaluation

It is an intrinsic property of high-to-low transfer that the

MLS-enforcing TCB alone cannot provide assurance of the transfer security. A guard process must be a trusted subject that is, by definition, "trusted" with the capability to perform downgrades not otherwise permitted by the security policy. To assure overall system security, a guard trusted subject must be certified (and accredited) in the context of the overall system.

The TCB subsets evaluation approach, however, can dramatically constrain the scope of the system certification and accreditation effort and enable controlled and rapid upgrades in response to dynamic operational environments. Because the system consists of well-defined subsets, each subset can be examined separately, permitting incremental evaluation of the system.

Incremental evaluation means that guard trusted subjects can be added and modified, requiring evaluation only of the new or modified trusted subjects and without the need to reevaluate the entire system.

The incremental evaluation capability of GEMSOS itself can also constrain the scope of system certification and accreditation that may be necessary due to future enhancements in the underlying TCB. Although, for example, the current GEMSOS implementation has refreshed technology (e.g., to a 32-bit TCB interface and large memory segments), the security foundation of the formal specification and model have remained inviolate. Thus, the inviolable, conceptually simple, overall security architecture of the AVG and its underlying TCB can reduce the initial as well as recurring certification and accreditation time and effort.

D. Secure hardware sharing and extension

The GTNP Final Evaluation Report prepared by NSA explicitly notes GEMSOS's ability to support "a virtual machine on top of the Virtual Machine Monitor provided by the GTNP" [25]. The AVG leverages this feature to implement guards.

Unlike guards built on low-assurance platforms, the high-assurance platform on which the AVG is built provides verifiable isolation within a single multi-domain system to allow the AVG to support multiple "virtual" guards running simultaneously within the same system. The non--bypassable, assured transaction pipelines used in the AVG insure that the trusted guards will be invoked and prevent them from interfering with one another. The guards are not restricted to the same transfer security policy and domains, but may support different transfer security policies, between multiple domains, with confidence that each guard is properly isolated and protected.

GEMSOS, moreover, is a multi-processing operating system and supports a scalable processing capability to provide additional computing capacity in resource-intensive environments. This expandability can reduce the number of systems needed for a given workload. Combined with the guard virtualization capability, the scalability makes the AVG suitable for use in multi-domain environments and sharply reduces or eliminates the need for redundant and duplicative resources.

VI. GUARD VIRTUALIZATION

The design presented in section IV is for the one-way transfer of data for a single transfer security policy. This is somewhat typical of current guard implementations. Each instance of a guard is hosted on its own (often dedicated) platform. The AVG architecture, however, virtualizes this guard host environment, which allows the AVG to support multiple instances of isolated and independent guards on the same platform. This virtualization depends only on the GEMSOS TCB, without introducing the serious risks and limitations of using a general-purpose, low-assurance, commodity, virtual machine monitor.

A. Hosting Multiple Virtual Guard Services

In figure 1, two network connections are shown: one for "High" network clients and one for "Low" network clients, and there is only one guard service for transfer between the two separate network interfaces for high and low clients. End-to-end control of the data transfer, including any pre- and post-processing for a particular guard, is implemented using the guard identifier integrity category, two other integrity categories for implementing the assured pipelines, and the MAC policy enforced by the underlying GEMSOS TCB.

GEMSOS has available a large number of independent integrity categories, so another, completely isolated and protected guard can be created by using a different guard identifier integrity category, along with the same two assured pipeline categories, in the same way as shown in figure 1. The new guard created using the different guard identifier category creates a separate and distinct virtual guard service. Multiple guard services created in this way can run on the same platform at the same time, each supporting an independent transfer security policy. The virtual guard services can potentially support different data flows between any paired combinations of the same or different domains.

B. Virtual Guard Service over Shared Network Interface

Additional physical network interfaces can be used to provide network connections between new guard instances and their high and low clients (which may be in security domains different from other instances). Separate physical network interfaces can help keep the domains separate. Guard virtualization supported by the AVG, however, does not require a separate network interface per domain. Another type of guard can be used to encrypt, seal, and forward packets from different domains to create virtual networks, securely multiplexing multiple access domains on the same network.

Consider, for example, the network depicted in figure 2. A "system high" network connects Internet or other untrusted sources to lower-domain hosts at the same time as it carries high-domain data. Stand-alone crypto-seal guard appliances, called "GemSeal" guards [37], sit on the network in front of all low-domain hosts and bridges to the Internet and other low-domain networks.

The GemSeal guards, which are themselves built on GEMSOS, seal packets with their source label and forward

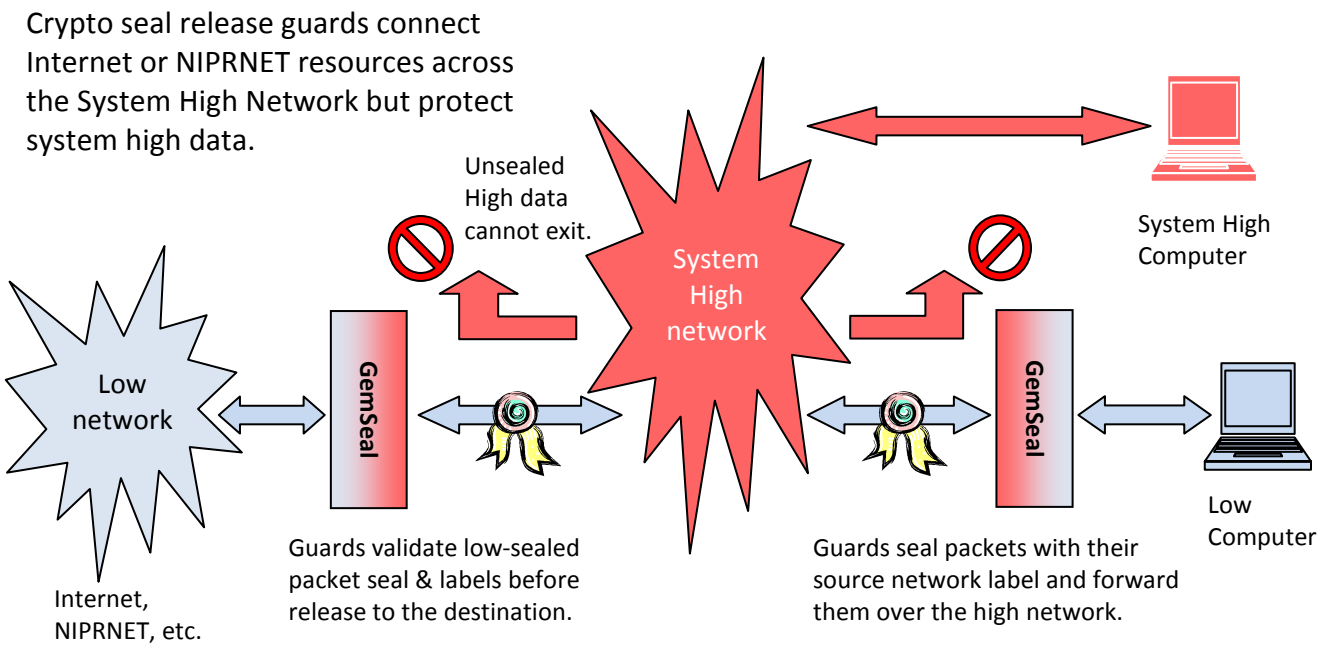

Fig. 2 Crypto Seal Guard Concept

them over the system high network. The seals protect the integrity of the labels and data. Unlabeled or altered packets cannot enter a guarded destination because they will not have a crypto seal that binds a label to a matching destination label, preventing high-domain data from being released to low-domain hosts.

The TCSEC requires that "Sensitivity labels shall accurately represent security levels of the specific … objects with which they are associated. When exported by the TCB, sensitivity labels shall accurately and unambiguously represent the internal labels and shall be associated with the information being exported [27]." GEMSOS uses crypto seals internal to its TCB to protect the label and data integrity of non-volatile storage. GemSeal applies this same crypto seal concept to network packets to ensure that packet data is not altered and that the source sensitivity label is authentic.

We can generalize the GemSeal architecture shown in figure 2 to support multiple domains – not just the two in the figure – by putting GemSeal guards in front of every host of every domain. The guards would forward labeled and encrypted packets across the shared network to a guard at the destination. Destination guards validate the data and label of each packet against the destination label before releasing it. In effect, GemSeal guards would implement VPN tunneling with a different tunnel for every domain pair.

The next step is to put the GemSeal appliance into an AVG system instead of having it as a separate appliance. The GemSeal would run as a higher-integrity subject than the transfer guard applications and would be part of the AVG TCB. The internal GemSeal could reliably determine the access domain of each message from the message's sealed label and place the message into the correct assured pipeline for trusted downgrade using the applicable virtual guard service.

A deployed application with significant similarities to the GemSeal concept is the NSA Class A1 BLACKER project to implement host-to-host secure communications across the Defense Data Network. BLACKER supports "eight security labels, from Unclassified to Top Secret" for messages. BLACKER used the GEMSOS kernel for key management and distribution, and for the access control center which "is the 'brains' of the system [26]." GemSeal has also been proposed for use in securing critical infrastructure networks [38].

VII. Conclusion

The raison d'être for a guard is to enhance the information security of system. The limited assurance of current guard technology, however, can have the opposite effect, by introducing serious vulnerabilities to subversion and by necessary deployment restrictions as a result of a lack of trust in the guard systems. The AVG architecture substantially mitigates those risks by leveraging the mature and proven GEMSOS TCB designed to meet the Class A1 requirements.

The AVG, furthermore, uses the proven policy composition tools of TCB subsets and TCB partitions to enable secure and systematically repeatable incremental evaluation. When applied effectively, such composition approaches can dramatically reduce the time and effort required for, and increase confidence in, certification and accreditation efforts. The benefits of incremental evaluation would be especially valuable in dynamically changing environments, such as those faced in military deployments, where frequent recertification may be necessary as the system is modified in the face of a rapidly evolving adversary.

Beyond those considerations for insuring adequate security, guard virtualization enables more effective use of resources. Not only can this reduce costs, it can also reduce the space, weight, and power footprint that may be even more important in aircraft and space deployments.

REFERENCES

[1] Committee on National Security Systems, "National Information Assurance (IA) Glossary", CNSS Instruction 4009, 26 April 2010. (CNSSI4009). Available: www.cnss.gov/Assets/pdf/cnssi_4009.pdf

[2] The "72 Year Rule", United States Census Bureau, http://www.census.gov/history/www/genealogy/decennial_census_records/the_72_year_rule_1.html

[3] *Multilevel Security in the Department Of Defense: The Basics*, Defense Information Systems Agency, Department of Defense Multilevel Security Program, 1 March 1995. Available: http://www.zoklet.net/totse/en/hack/hack_attack/multsec.html

[4] K. Thompson, "Reflections on trusting trust," *Communication of the ACM*, Vol. 27, No. 8, August 1984, pp. 761-763. Available: http://cm.bell-labs.com/who/ken/trust.html

[5] M. Bailey, "The Unified Cross Domain Management Office: Bridging Security Domains and Cultures," in *CROSSTALK: The Journal of Defense Software Engineering*, July 2008. Available: http://www.crosstalkonline.org/storage/issue-archives/2008/200807/200807-Bailey.pdf

[6] "UCDMO Cross Domain Baseline List: As of 27 January 2012", Available: http://www.owlcti.com/pdfs/certifications/UCDMO_v.3.5.0_Baseline_Inventory.pdf

[7] R. R. Schell and D. L. Brinkley, "Evaluation criteria for trusted systems", in *Information Security: An Integrated Collection of Essays*, Abrams, Jajodia, and Podell, eds., IEEE Computer Society Press, Los Alamitos, CA, pp. 137-159, 1995.

[8] "Information Support Server Environment - ISSE-Guard ," http://www.globalsecurity.org/intell/systems/isse-guard.htm

[9] "ITT Exelis cross-domain solution approved for deployment on U.S. government networks," April 2, 2012, http://www.exelisinc.com/News/PressReleases/Pages/ITT-Exelis-cross-domain-solution-approved-for-deployment--on-U.S.-government-networks.aspx

[10] "ISSE (Information Support Server Environment)," http://www.exelisinc.com/solutions/ISSE/Pages/default.aspx

[11] "Raytheon High-Speed Guard," http://www.raytheon.com/capabilities/rtnwcm/groups/iis/documents/content/rtn_iis_highspeedguard_ds.pdf

[12] B. Fletcher, C. Roberts, and K. Risser, "The design and implementation of a guard installation and administration framework", 2007 SELinux Symposium and Developer Summit, 26 January 2007. Available: http://selinuxsymposium.org/2007/papers/10-GIAF.pdf

[13] K. MacMillan, S. Shimko, C. Sellers, F. Mayer, and A. Wilson, "Lessons learned developing cross-domain solutions on SELinux", Tresys Technology, LLC., March 2 2006, unpublished white paper. Available: http://www.tresys.com/pdf/Lessons-Learned-in-CDS.pdf

[14] "SELinux Frequently Asked Questions (FAQ)," http://www.nsa.gov/research/selinux/faqs.shtml#I13

[15] *Cross-domain solutions for airborne operations*, Small Business Innovation Research (SBIR) contract FA8750-10-C-0117, 2010, http://www.sbir.gov/sbirsearch/detail/3097

[16] A. Gehani, D. Hanz, J. Rushby, G. Denker, and R. DeLong. "On the (f)utility of untrusted data sanitization," in *Proc. MILCOM 2011 Military Communications Conference* (MILCOM 2011), IEEE. November 2011, Baltimore, MD.

[17] J. Alves-Foss, W. S. Harrison, P. Oman, and Carol Taylor. "The MILS architecture for high-assurance embedded systems," in *International Journal of Embedded Systems*, 2(3/4):239–247, 2006.

[18] D. E. Bell, "Looking back at the Bell-LaPadula model," *Proceedings of the 21st Annual Computer Security Applications Conference*, December 2005.

[19] E. A. Anderson, C. E. Irvine, and R. R. Schell, "Subversion as a threat in information warfare," in *Journal of Information Warfare*, Volume 3, No.2, June 2004, pp. 52-65.

[20] *Intel 64 and IA-32 Architectures Software Developer's Manual*, May 2011. Available: http://www.intel.com/content/dam/doc/manual/64-ia-32-architectures-software-developer-vol-1-2a-2b-3a-3b-manual.pdf

[21] W. R. Shockley and R.R. Schell, "TCB subsets for incremental evaluation", in *Proc. AIAA/ASIS/IEEE 3rd Aerospace Computer Security Conference*, 1987, pp 131-139.

[22] C. E. Irvine, "A multilevel file system for high assurance," in *Proceedings 1995 IEEE Symposium on Security and Privacy*, Oakland, CA, pp. 78-87, May 1995.

[23] "Trusted Network Interpretation of the Trusted Computer System Evaluation Criteria", DoD 5200.28–STD, 31 July 1987, NCSC–TG–005.

[24] R. R. Schell, T. F. Tao, and M. R. Heckman, "Designing the GEMSOS security kernel for security and performance," in *Proceedings of the 8th DoD/NBS Computer Security Conference*, 1985, pp. 108-119.

[25] "Final Evaluation Report, Gemini Computers, Incorporated, Gemini Trusted Network Processor, Version 1.01", National Computer Security Center, 1995. Available: http://www.aesec.com/eval/NCSC-FER-94-008.pdf

[26] C. Weissman, "BLACKER: security for the DDN examples of A1 security engineering trades," in *Proc. 1992 IEEE Computer Society Symposium on Research in Security and Privacy*. Oakland, CA: IEEE, 1992, pp. 286-292

[27] "Department of Defense Trusted Computer System Evaluation Criteria" (Orange Book), 5200.28-STD, United States National Computer Security Center, December 1985.

[28] Common Criteria for Information Technology Security Evaluation, Version 3.1, CCMB-2009-07-001, July 2009.

[29] "Cyberattacks on Iran – Stuxnet and Flame", Updated June 1, 2012. http://topics.nytimes.com/top/reference/timestopics/subjects/c/computer_malware/stuxnet/index.html

[30] W. Boebert and R. Kain, "A practical alternative to hierarchical integrity properties," in *Proceedings of the 8th DoD/NBS Computer Security Conference*, 1985, pp. 18-27.

[31] T. M. P. Lee. "Using mandatory integrity to enforce 'commercial' security." In *Proceedings of the 1988 IEEE Symposium on Security and Privacy*, IEEE Computer Society, Oakland, CA, USA, 18- 21 April 1988.

[32] K. J. Biba, "Integrity Considerations for Secure Computer Systems," MITRE Technical Report MTR-3153, April 1977.

[33] D. E. Bell and L. J. LaPadula, "Secure Computer Systems: Unified Exposition and Multics Interpretation," Mitre Technical Report MTR–2997, March 1976.

[34] R. Sandhu, "Lattice-Based Access Control Models," in *IEEE Computer*, 26(11): 9-19, Nov. 1993.

[35] B. Callaghan, B. Pawlowski, and P. Staubach, "NFS Version 3 Protocol Specification," Sun Microsystems, Inc., June 1995.

[36] D. P. Reed and R. K. Kanodia, "Synchronization with Eventcounts and Sequencers", Communications of the ACM, February 1979, Volume 22, No. 2, pp. 115-123.

[37] Aesec Global Services, "GemSeal Guard: High Assurance MLS," Unpublished white paper, 2007.

[38] M. R. Heckman, R. R. Schell, and E. E. Reed, "Composing a high-assurance infrastructure out of TCB components," presented at the 5th Annual Layered Assurance Workshop (LAW 2011), Orlando, FL, USA. December, 2011. Available: http://fm.csl.sri.com/LAW/2011/law2011-paper-heckman.pdf