Towards Formal Evaluation of a High-Assurance Guard

Mark R. Heckman   Roger R. Schell   Edwards E. Reed

Aesec Global Services, Inc.
Palo Alto, California

{mark.heckman, roger.schell, ed.reed}@aesec.com

ABSTRACT

A transfer guard built on a high-assurance multilevel secure (MLS) trusted computing base (TCB) must be a trusted subject with the capability to perform downgrades not otherwise permitted by the MLS security policy. Formal evaluations of MLS systems containing trusted subjects are complicated when the trusted subjects are evaluated as part of a monolithic TCB. While well-developed techniques of "TCB subsets" and "TCB partitions" for composing MLS systems exist, approaches for applying these techniques for reasoning about a guard downgrade policy are not as well-developed. If the trusted downgrade process must be evaluated as part of the TCB, an inability to adequately reason about the downgrade policy would make it difficult to reason about the policy implemented by the system as a whole. And if trusted subjects cannot be evaluated separately and composed with the underlying TCB, that could lead to to expensive and repetitive certification and recertification of the system when downgrade policies change. A necessary pre-requisite for feasible evaluation of high-assurance transfer guard systems is to be able to separately evaluate the assurance of the underlying system and the implementation of the downgrade policy. This paper suggests an approach to extending the well-developed technique of "balanced assurance" to the formal evaluation of high-assurance transfer guards that could permit the downgrade function to be evaluated separately from the underlying TCB and then composed with it into an overall system.

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: Security and Protection– *access controls, information flow*

General Terms

Design, Security, Verification

Keywords

Evaluation, GEMSOS, High-assurance, Multilevel security, TCB Subsets, TCB Partitions, Transfer Guard, Virtualization

1. INTRODUCTION

A transfer device is a type of "Cross Domain Service" (CDS) that permits the movement of data from one domain to another [1]. The information to be transferred is contained in the first domain, but is authorized for the second. The purpose of the transfer device is to ensure 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 moderate 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 [5].

(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.) We refer in this paper to transfer devices that implement any of these types of policies as transfer "guards".

Guards have generally been built on low-assurance, commodity technology. For example, a transfer device listed in a recent Unified Cross Domain (CD) Management Office (UCDMO) Baseline List, a list of CDSs that are available for deployment by U.S. Government agencies [6], runs on "commodity commercial off-the-shelf servers running Red Hat Enterprise Linux 5 with a Strict SELinux policy [7]", and SELinux has also been proposed as a suitable base for guards by others [8][10]. The U.S. National Security Agency (NSA), however, has said 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" [11].

The Aesec Virtual Guard (AVG) architecture is a design for a transfer guard built on a high-assurance TCB and intended to address weaknesses in current guard technology [12]. The AVG architecture incorporates the commercial, off-the-shelf (COTS) Gemini Secure Operating System (GEMSOS) TCB [14]. The NSA has previously evaluated the GEMSOS security kernel and product Ratings Maintenance Phase (RAMP) plan at the highest level, Class A1, of the NSA's "Trusted Computer System Evaluation Criteria" (TCSEC, also known as the "Orange Book") [16], as part of the evaluation of the Gemini Trusted Network Processor (GTNP) [15].

A goal of the AVG is to make it evaluable at the EAL7 level of the "Common Criteria for Information Technology Security Evaluation" [17] with a suitably strict protection profile that gives assurance equivalent to TCSEC Class A1. This goal requires the ability to formally reason about the security enforced by the system. Reasoning about an entire system as a monolithic entity, however, can be quite difficult. To make this effort feasible, the system must be decomposable into smaller parts that can be evaluated separately and then composed together into an overall evaluated system.

Accepted for presentation at the Layered Assurance Workshop (LAW) 2012, December 3-4, 2012, Orlando, FL

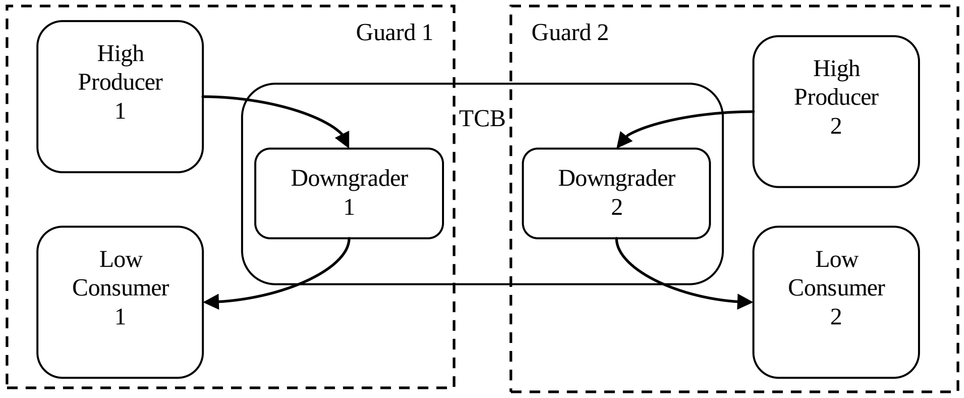

Figure 1 – Multiple, Independent Guards on one TCB

Ideally, we'd like to be able to separately certify (and accredit) transfer guards in the context of the overall system, with the ability to add or modify guards but requiring evaluation only of the new guards and without the need to reevaluate the entire system.

It is an intrinsic property of high-to-low transfer that the multilevel security (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. The use of trusted subjects complicates the evaluation of transfer guard systems. For example, while the Bell-LaPadula access control model, frequently used for evaluating the properties of MLS TCBs, includes the notion of a trusted subject as "those subjects not constrained by the *-property" [20] (the *-property says that a subject at a given security level must not write to any object at a lower security level), the model cannot address the information transfer function of the trusted subject itself, saying only "Outside the model, a subject, to be designated 'trusted,' must be shown not to consummate the undesirable transfer of high level information that *-property constraints prevent untrusted subjects from making."

Well-developed techniques exist for composing TCB systems where the systems share a homogeneous policy (e.g., "TCB Partitions" [19]) or where one system's policy is a refinement of the policy implemented by the other ("TCB Subsets" [18]). The application of these techniques, however, is complicated when trusted subjects are part of the component systems. For example, the technique of TCB Subsets, described in the "Trusted Database" interpretation (TDI) of the TCSEC [18], generally requires that each TCB subset must include all of the subjects that are trusted with respect to its technical policies. If this restriction holds, we would be unable to evaluate the transfer guard function separately from the TCB.

An additional problem arises when attempting to compose a collection of interconnected components when each has a different, heterogeneous security property, a problem sometimes called the "unconstrained composition" problem [2]. The downgrade function implemented by a transfer guard appears to enforce a quite different sort of policy than the access control policy enforced by the underlying TCB. And the downgrade function itself may consist of a series of multiple, different functions.

In this paper, we explore how the proven composition techniques of Trusted Subjects and TCB Partitions can be used to support compositional evaluation of a transfer guard implemented as a trusted subject, or subjects, running on a high-assurance MLS TCB. Our focus is on how these techniques can be used to evaluate the infrastructure that surrounds the downgrade function, but not necessarily how to evaluate the correctness of unconstrained composition in the downgrade function itself. First, we present an abstract architecture for a transfer guard and describe the AVG architecture and its use of trusted subjects. We then describe how different compositional techniques can be used to support an incremental evaluation of a high-assurance transfer guard system built using the AVG architecture.

2. ABSTRACT ARCHITECTURE

Figure 1 is an abstract depiction of a transfer guard system. One or more independent downgraders are connected to high "producers" that produce the information to be downgraded, and to low "consumers" that consume the downgraded data. The combination of high producer, downgrader, and low consumer constitutes a "guard". The underlying TCB on which the downgraders run isolates the guards from one another. Given a TCB of sufficiently high assurance, each guard may have a different downgrade range.

The key formal argument questions that need to be made about a guard system at this level of abstraction are

I-1. Can the producer, downgrader, and consumer of each guard be composed into a guard

I-2. Can the guards be shown to be isolated from one another

I-3. Can each downgrader be evaluated separately from the underlying TCB

Figure 2 depicts how a downgrader may consist of one or more "stages", each of which implements a different downgrade policy on the data. The stages are organized into a pipeline. One stage processes the data and then passes the data to the next stage. The last stage releases the data. Key formal arguments that need to be made at this level of abstraction are

II-1. Can the code in each stage be shown to enforce the downgrade policy that it implements?

II-2. Can the stages be composed into a single guard downgrade policy?

II-3. Can the isolation and pipeline ordering properties be proven?

Formal arguments I-1 through I-3, and II-3 are "infrastructure" arguments and can be shown using techniques of TCB Partitions and TCB Subsets. For II-1 and II-2, however, which deal with policies specific to the downgrade function of the guard, other techniques must be used.

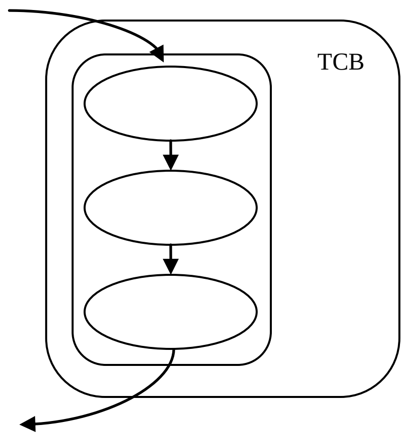

Figure 2 – Pipelined guard stages

Before discussing how the evaluation may proceed, we first describe an implementation of the abstract guard architecture on a high-assurance TCB. Several features of the implementation and the underlying TCB are fundamental to our composition arguments.

3. THE AVG ARCHITECTURE

We describe here an implementation of the abstract guard architecture described in section 2. The AVG architecture is a high-assurance transfer guard system based on the Class A1 GEMSOS TCB. We originally presented this description of the AVG in an earlier paper [12], but repeat it here for the convenience of the reader.

The AVG architecture is a design for a high-assurance transfer guard system implemented through GEMSOS-provided process isolation, GEMSOS-supported multilevel subjects, and assured pipelines created using GEMSOS mandatory access class labels. Network clients – the producers and consumers – communicate with the AVG through standard protocols such as Network File System (NFS). 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 3.

3.1 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 [4]. 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 [8][10].

Assured pipelines in the AVG are implemented using integrity categories, which are part of the integrity component of the mandatory security labels assigned to every subject and object managed by the GEMSOS TCB [14]. Each process in a guard downgrader shares a unique integrity category, called the *guard identifier*. The guard identifier protects the downgrader 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. Only the processes in a downgrader, however, have the guard identifier integrity category assigned to that guard.

Additional integrity categories are used to keep the pipelines of the downgrader ordered and separate. For example, in figure 3, the Output Queue Manager is trusted within a very specific integrity range so that it 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". This is an example of an assured pipeline. Integrity category "ic1" is the guard identifier, while "ic2" and "ic3" are used to implement the

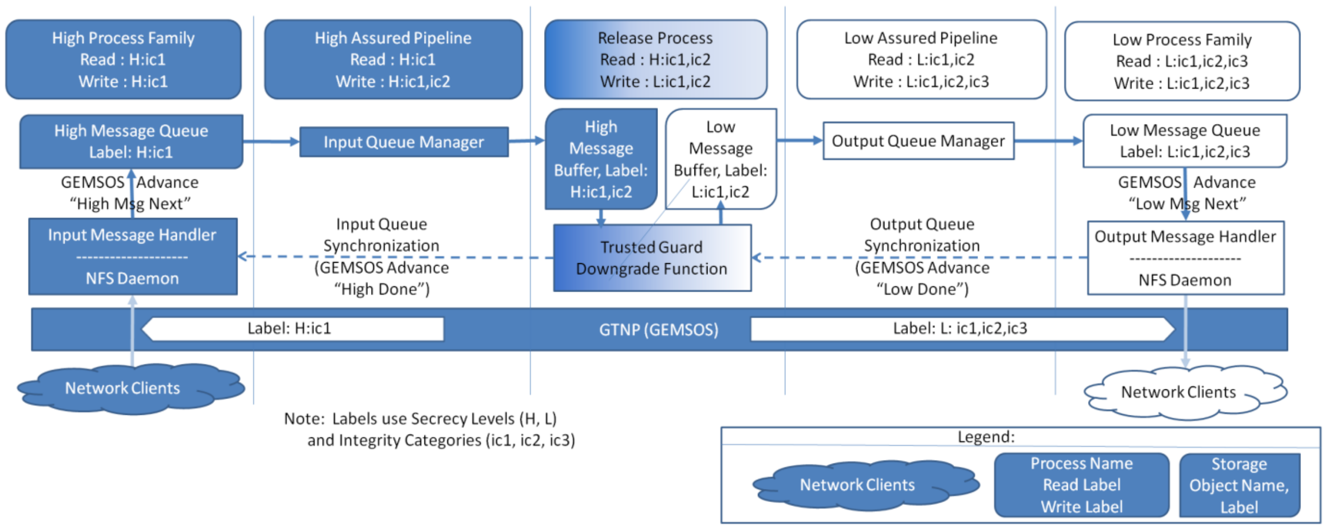

Figure 3 - Aesec Virtual Guard Architecture

assured pipeline. Other processes in the downgrader lack the “ic3” integrity category, so they cannot bypass the Release Process and write directly to the Low Message Queue. There is a second assured pipeline on the input side of the guard.

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

3.2 Trusted Subjects

The AVG architecture uses the multilevel, 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 [3], low-integrity data cannot flow to a higher-integrity domain. Assured pipeline trusted 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 bypass.

3.3 AVG Design

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

3.3.1 Guard Input

In figure 3, 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 a directory on the AVG system mounted by the client using the Network File System (NFS) protocol.

The directory (and the files it contains) 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 processes the file, writes the message into the High Message Queue, and signals the Input Queue Manager that a message is in the High Message Queue by incrementing an eventcount – a type of secure synchronization object implemented in GEMSOS [13]. The operation to increment an eventcount is called “advance”. The eventcount has the same access class as the High 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 High 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. Another eventcount is used to signal the Input Message Handler when space is available in the High Message Queue. This type of synchronization using eventcounts is performed for each of the queues and buffers in the downgrader. We will omit further details about buffer synchronization in order to simplify the description of the downgrader.

The Input Queue Manager process transfers each message from the High Message Queue to a High Message Buffer that it shares with the Release Process. The Input Queue Manager is an assured pipeline process that is 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. The additional category “ic2” means that the High Message Buffer has higher integrity.

3.3.2 Guard Release

The Release Process implements the downgrade function (a single stage function, in this example) and is trusted within a secrecy range “High” to “Low”. Note that the “High” and “Low” labels in this example are not necessarily system high and system low, but are the configured high and low secrecy levels of the range to which the downgrader has been constrained. 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 Release Process reads an input message from the High Message Buffer, performs downgrade processing on the message, and writes a downgraded message to the Low Message Buffer, which has secrecy level “Low” and the same two integrity categories (“ic1” and “ic2”) as the High Message Buffer. From this point on, all processing in the guard is done at the “Low” secrecy level.

3.3.3 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 into the Low Message Queue. The Output Message Handler copies each message from the Low Message Queue into a file in a Low directory in the file system. Clients on the Low network can retrieve the “Low” files containing sanitized data by mounting the Low directory using the NFS protocol.

3.4 Multiple Pipeline Stages

An important advantage of the assured pipeline feature of the AVG architecture is the ability to easily add additional stages. The figure shows a single-stage downgrader, but additional assured pipelines can be created using additional integrity categories (“ic4”, etc.). Each of these pipelines can be used to connect to an additional stage.

3.5 Performance

We implemented a prototype of the AVG and ran it on a single 550MHz Pentium III processor system. The downgrade function consisted of a single-stage "dirty word search". The system processed approximately 2500 4KB messages/second. This processing rate was consistent, up to the test maximum of 3 simultaneous guard instances.

4. FORMAL ARGUMENTS

We divide formal arguments into two parts: "infrastructure" and "downgrade policy". Infrastructure arguments are concerned with, for example, the isolation properties enforced by the underlying TCB. Downgrade policy arguments are concerned with the correctness of the unconstrained composition of the downgrade functions.

4.1 Infrastructure Arguments

Infrastructure arguments can be divided into four parts:

1. That the downgrader can be evaluated separately from the underlying TCB
2. That the composition of the producer, downgrader, and consumer implement a complete guard policy
3. That the guards are isolated from one another
4. That the isolation and pipeline ordering properties required by the downgrade policy components are correct

4.1.1 Process Isolation and Pipeline Ordering

The GEMSOS TCB enforces process separation, in general. The use of guard identifier integrity categories in the AVG ensures that all components of a downgrader are isolated and protected, even from trusted subjects in other downgraders.

The AVG uses TCB integrity categories to implement non-bypassable assured pipelines. The ordering provided by the assured pipelines in the AVG, however, is not complete: processes farther along the pipeline have higher integrity (viz., more integrity categories) and so can write data into earlier stages of the pipeline. There is no functional reason for them to do so, however, and there is no security risk from the point of view of information flow, because high-level data cannot bypass the downgrader.

4.1.2 Composition of Guard System

A guard consists of a producer, downgrader, and consumer, which communicate through a network. Each downgrader, whose isolation from other downgraders is enforced by the underlying TCB, is implemented in the AVG as a virtual machine.

The Trusted Network Interpretation (TNI) of the TCSEC proposes a virtual machine as an example of a network component [19]. A downgrader running on a virtual machine implemented using guard identifier integrity categories on a high-assurance TCB meets the TNI requirements for a NTCB partition in that there is a clearly distinguished TCB with a definitive protection domain boundary. Additional guards, which are kept separate by their unique guard identifiers and the enforcement of the underlying system, are also NTCP partitions. The evaluation of the entire system is a composition of the partitions. Consistent with this TNI approach 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" [16]. The AVG leverages this feature.

Having established that each of the downgraders running independently of one another on the TCB are network components, it is a relatively simple matter to compose a downgrader together with its producer and consumer, which are also network components, using the TNI's technique of "TCB Partitions" [19].

4.1.3 Separation of Downgrader from the TCB

This is a critical argument for the AVG in that, if no such argument can be made, it would be impossible to add or modify guards without the need to reevaluate the entire system.

In a transfer guard system, there are two, policy-enforcing entities: the downgrade policy implemented by the transfer guard and the access control policy enforced by the underlying TCB that protects and limits the transfer guard. The TCB definition includes the notion of trusted subjects while the guard downgrader refines the definition of trusted subject to sharply limit what those trusted subjects are allowed to do.

A formal argument about the distinctness between the guard and the underlying TCB can be made using the technique of TCB Subsets, applied in the "Trusted Database" interpretation (TDI) of the TCSEC [18]. Although the TDI says that each TCB subset must include all of the subjects that are trusted with respect to its technical policies, the TDI in section TC-6.4 also provides an example of a special case, where "a previously evaluated TCB or TCB subset is modified to accommodate the policy-enforcing elements of a new application layer" and "the alteration takes the form of a less primitive subset which is implemented, at least in part, with trusted subjects." In this example, the TDI says "the local analysis [of the more-primitive subset on which the trusted subject runs, i.e., the TCB] represents a reasonable candidate for analysis that need not be redone." The justification in the TDI is two-fold:

1. The policy of the more primitive subset (the TCB, in this case) includes the strict enforcement of a mandatory access control policy that allows trusted subjects to execute in the less-primitive subset domains that compose the downgrader. This condition is met by the multilevel subject feature of GEMSOS.
2. Analysis of the security properties of the TCB subset is unaffected by changes to the downgrader.

The TDI is careful to point out that "an assessment of the impact of [the less-primitive subset] on the behavior of [the TCB] cannot be made strictly by an examination of the trusted subjects and the definition of [the TCB's] interface. A global assessment … is required [18]." The important point, however, is that the downgrader can be changed without requiring re-evaluation of the underlying TCB.

Previous work on secure databases, moreover, has introduced the concept of "balanced assurance", concluding "We do not believe that it is necessary to require all of the Class Al assurance techniques for the extended TCB, because it is constrained by the underlying general-purpose TCB . . . because the risk associated with its incorrect operation is correspondingly less [9]." Similarly, a downgrader, although trusted, is constrained by the underlying TCB, so it does not necessarily require all of the Class A1 assurance techniques.

The AVG is an instance of the example in the TDI, where the downgrader is a set of trusted subjects running on the previously evaluated TCB. The process separation and access control enforced by the underlying TCB (through use of the guard identifier integrity category) isolates the guard subsystem from the rest of the system. All guard processing is constrained by the guard identifier and the administratively configured downgrade range. In particular, only information in objects that are created with the guard identifier can be downgraded. On a properly configured system, the guard subjects can't write information outside of the downgrader because the guard identifier prevents them, and nothing outside the downgrader has the guard identifier so nothing outside the downgrader can write data to it. The guard identifier is an example of a "tamperproof attribute" that implements "virtual partitioning", a form of isolated protection domain [25].

4.2 Downgrade policy Arguments

The downgrade policy arguments are concerned with the proving that the downgrader correctly implements the downgrade policies. Formal techniques for downgrade policy arguments are not well developed, however, and although our current work did not extend to this area, we briefly describe here some published techniques for reasoning about the correctness of a downgrade stage and for composing downgrade stages into an overall downgrade policy, and explain how the isolation provided by the mandatory security enforcement of the underlying TCB satisfies pre-requisites for applying these techniques.

4.2.1 Single Stages

An early attempt at formal verification of a transfer guard was NASA's Restricted Access Processor (RAP). The RAP was intended to prevent messages containing classified data from reaching a system with uncleared users and to prevent messages sent by uncleared users from "affecting" classified data – i.e., simultaneously enforcing secrecy and integrity policies. The verification effort included a model designed specifically for the RAP and a formal top-level specification (FTLS). A formal proof was performed to show the correspondence between the model and the FTLS, and an informal argument showed the correspondence between the FTLS and the code [21].

A more recent example of a formal technique for reasoning about the implementation of a downgrade policy is the Guardol programming language. Guardol is specifically intended to aide in the creation and formal verification of transfer guards [22]. An unstated caveat of Guardol is that the properties of programs that are proven using Guardol cannot be enforced unless the integrity of the code and data is protected, as it would be in a high-assurance TCB. This is an example of how infrastructure arguments are not only separable from downgrade policy arguments, but are also supportive of them.

Class A1 requirements, including strict configuration management and special safeguards, along with GEMSOS classification labels, including integrity labels, can be used to protect code and data from tampering by other subjects in the TCB.

4.2.2 Composition of Stages

The composition of the different stages in a downgrader must be shown to correctly implement the overall guard downgrade policy. An example of a technique that can be used for this purpose is the composition principle of Abadi and Lamport [23][24]. A key requirement of this composition method is to partition "agents" (the entities that perform state changes) into environment agents and system agents. When composing two systems, the agents of one system are part of the environment of the other. The process separation and classification labels of the underlying TCB can be used to validate that the agents are completely identified and distinct, satisfying the prerequisites of the composition principle.

5. CONCLUSION

In this paper we have sketched out some of the difficulties involved with formally evaluating a high-assurance transfer guard and how a combination of techniques can be used to evaluate a transfer guard system in parts that can later be composed. There are two major advantages to this approach:

1. The compositional evaluation means that components can be evaluated separately, so if a component later changes, the entire evaluation does not have to be redone.
2. The compositional techniques for which there is a great deal of confidence can be separated from less-well-developed techniques that must be used for other components, thereby increasing confidence in the correctness of the overall evaluation.

6. REFERENCES

[1] Bailey, M. 2008. "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

[2] Bell, D. 2005. "Looking back at the Bell-La Padula model". In *Proc. 21st Annual Computer Security Applications Conference* (ACSAC 05), Pages 337-351. Available: http://www.acsac.org/2005/papers/Bell.pdf

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

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

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

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

[8] Fletcher, B., Roberts, C., and Risser, K. 2007. "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

[9] Lunt, T.F., Schell, R.R., Shockley, W.R., Heckman M., and Warren, D. "A near-term design for the SeaView multilevel database system". In Proc. 1988 Symposium on Security and Privacy. 1988. Available: http://www.cs.washington.edu/research/projects/poirot3/Oakland/sp/PAPERS/00044435.PDF

[10] MacMillan, K., Shimko, S., Sellers, C., Mayer, F., and Wilson, A. 2006. *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

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

[12] Heckman, M.R., Schell, R R., Reed, E.E. 2012. "A high-assurance virtual guard architecture," to be published in *Proc. IEEE Military Communications Conference (MILCOM) 2012*.

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

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

[15] *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

[16] *Department of Defense Trusted Computer System Evaluation Criteria*, 5200.28-STD, United States National Computer Security Center, December 1985. Available: http://csrc.nist.gov/publications/history/dod85.pdf

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

[18] *Trusted Database Management System Interpretation of the TCSEC (TDI)*, April 1991, NCSC-TG-021. Available: https://www.fas.org/irp/nsa/rainbow/tg021.htm

[19] *Trusted Network Interpretation of the Trusted Computer System Evaluation Criteria*, DoD 5200.28–STD, 31 July 1987, NCSC–TG–005. Available: http://csrc.nist.gov/publications/secpubs/rainbow/tg005.txt

[20] Bell, D. E. and LaPadula, L. J. 1976. "Secure Computer System: Unified Exposition and Multics Interpretation," Technical Report ESD-TR-75-306, MITRE Corp., March 1976. Available: http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA023588

[21] Proctor, N., The Restricted Access Processor: an example of formal verification. *Proc. 1985 IEEE Symposium on Security and Privacy*, p. 49-53. Available: http://www.cs.washington.edu/research/projects/poirot3/Oakland/sp/PAPERS/00044558.PDF

[22] Hardin, D., Slind, K., and Whalen, M. 2011. "Introduction to the Guardol programming language and verification system," 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-hardin.pdf

[23] Abadi, M. and Lamport, L. 1993. Composing specifications. *ACM Trans. Program. Lang. Syst*. 15, 1 (January 1993), 73-132. DOI= http://doi.acm.org/10.1145/151646.151649

[24] Heckman, M.R. and Levitt, K.N. 1998. "Applying the composition principle to verify a hierarchy of security servers." *Proc. of 31st Hawaii International Conference on System Sciences*, Vol. 3, p.338-347. Available: http://seclab.cs.ucdavis.edu/papers/pdfs/mh-kl-98.pdf

[25] Shockley, W.R. and Schell, R.R. 1987. TCB subsets for incremental evaluation". *Proc. AIAA/ASIS/IEEE 3rd Aerospace Computer Security Conference*, 1987, pp 131-139. Available: http://www.acsac.org/secshelf/papers/tcbsubsets.pdf