

Building Multilevel Secure Web Services-Based Components for the Global Information Grid

Dylan McNamee , Galois Connections, Inc.
CDR Scott Heller , Program Executive Office C4I and Space
Dave Huff , Fleet Numerical Meteorology and Oceanographic Center

A consensus is growing that the Department of Defense's vision of a future Global Information Grid will be built using architecture that takes advantage of Web services and uses standard Internet protocols, interchangeable components, and commercially available hardware and software wherever possible. This article describes the features and architecture of two systems: the Trusted Services Engine and the Multilevel Document Collaboration Server, including their use of a separation kernel with multiple independent levels of security, the design and assurance architecture of the cross-domain block-access controller, and the composition architecture that extends the inter-level isolation property from the block access controller outward through complex services.

The Global Information Grid (GIG) is the overall architecture intended to replace current stovepipe information systems. A consensus is growing that the Department of Defense's vision of this future GIG will use an architecture that takes advantage of Web services and uses standard Internet protocols, interchangeable components, and commercially available hardware and software wherever possible. By adopting modern standards-based protocols, the GIG will enhance current capability by enabling people and components to work together dynamically with integrated data.

Protocols such as Hypertext Transfer Protocol, eXtensible Markup Language (XML), Web-based Distributed Authoring and Versioning (WebDAV), Really Simple Syndication, and Lightweight Directory Access Protocol allow the GIG to be made of off-the-shelf components where appropriate. Where custom components are required, pervasive use of these protocols preserves the component-based architecture of the GIG, thus protecting the architecture from developing into a stovepipe system.

Many of these components and protocols are mature and well understood, but they were not designed with security as the paramount consideration. Securing the GIG is therefore a significant challenge. Particularly critical is securing its cross-domain services. For these, the GIG itself must somehow enforce separate levels of security.

Today, physical isolation enforces separation, though other technologies such as cryptography may someday be used. Such separation allows the use of commercial components as single-level components not responsible for cross-domain security concerns. However, for the GIG to realize its potential, some components must enable secure *cross-domain* data access. Clearly such components, while they must conform to commercial protocols, must be developed to *higher than* commercial standards.

This article, which describes such a component, has three main parts:

1. We describe the security and assurance attributes required of a cross-domain component of the GIG.
2. We describe the architecture and technologies we are using to achieve these attributes in the Trusted Services Engine (TSE), a network-enabled file store with integrated read-down across security domains.
3. We conclude by describing a system built on the TSE, the Multilevel Document Collaboration Server, to enable cross-domain collaboration *with* documents – an example of using simple cross-domain components to build more complex cross-domain systems using only standard protocols and APIs.

This article describes the features and architecture of both systems:

- The design and assurance architecture of the cross-domain *block access controller* (BAC).
- The use of a Multiple Independent Levels of Security (MILS) separation kernel.
- The composition architecture that extends the cross-domain isolation property from the MILS separation kernel to the BAC and outward through complex services.

This article is focused toward a technical audience familiar with Web services.

Assurance Requirements for Cross-Domain GIG Components

The nature and mission of the GIG makes it a prime target for trained, well-funded, and resourceful adversaries. The threats posed by such adversaries, coupled with the value of the information on the GIG, require us to show that the GIG components are robust in the face of these threats. In particular, the greater security risks associated with cross-domain components – as compared to single-level, commercial solutions – require a correspondingly higher level of trust. The process of generating and evaluating evidence of trustworthiness is known as *assurance*, the most difficult aspect of security engineering.

Two processes in the defense and intelligence communities support each other to generate assurance evidence for a GIG component: evaluation and certification. *Evaluation* is the process of validating security claims for a particular component. For example, the Common Criteria is an international standard for specifying claims of system security functionality and generating assurance that these claims are satisfied. We have determined that the cross-domain components we are building will need to meet the requirements for Common Criteria's Evaluation Assurance Level 6 or 7 [1].

Certification focuses on verifying that a component can be securely deployed at a particular site. Certification is best represented by such processes as Secret and Below Interoperability, and Top Secret and Below Interoperability. What these processes have in common is a way to tailor requirements for evaluation or certification of the following:

- Sensitivity of the data that the component handles.
- Severity of the threats it must withstand.

For example, under Director of Central Intelligence Directive 6/3, a cross-domain component that needs to demonstrate high assurance with respect to confidentiality must satisfy Protection Level 4 or 5 assurance requirements. Evaluating or certifying a component to one of those standards requires an extensive investment in time and resources. But given the responsibilities of a cross-domain component of the GIG, high assurance is a must.

Architecture for a High-Assurance GIG Component

The TSE, a government off-the-shelf software development project funded by the Space and Naval Warfare Systems Command (SPAWAR) and National Security Agency, is a network-enabled file store with integrated read-down across security domains. The TSE provides the file store using the standard WebDAV protocol. It has a separate hardware network interface for each network security level and a separate file store for data at each level.

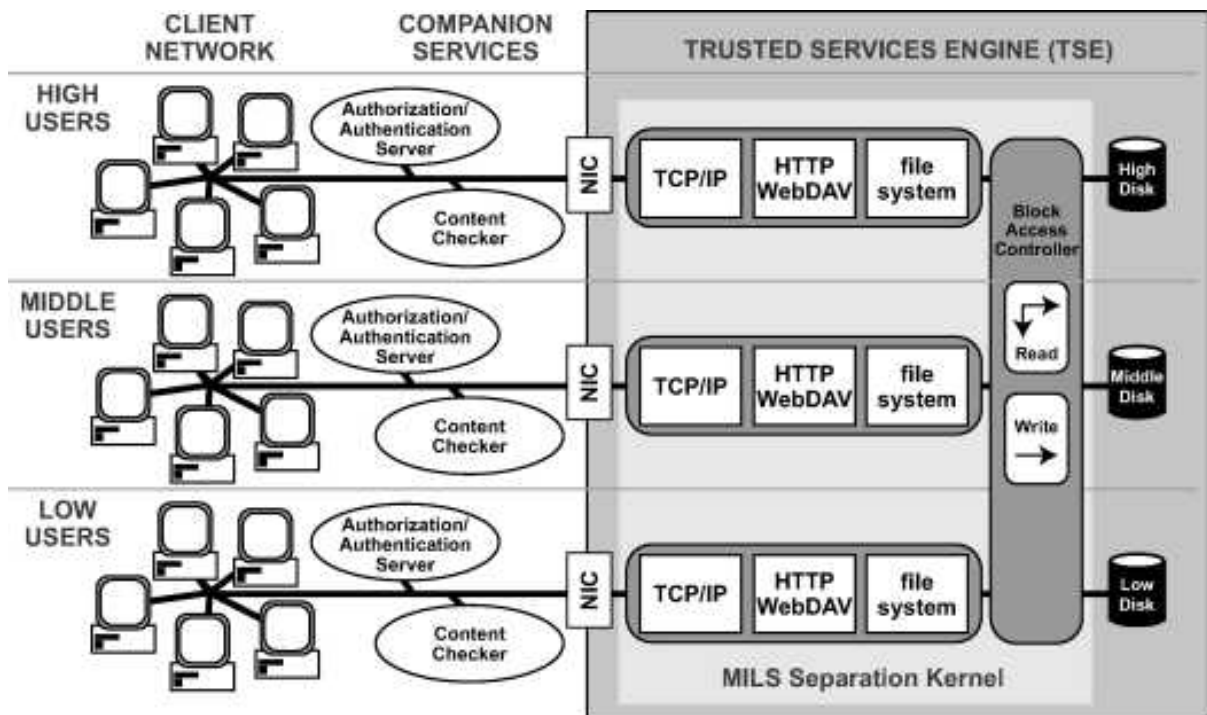
The TSE enforces the Bell-LaPadula policy of information flow [2], in which users on each network can read from their own level and below, but can write only to their own level. For example, when one security level dominates another (for example, TOP SECRET dominates SECRET), the TSE allows *read-down* – the ability for users at a higher level to access data from a lower level, but not vice-versa. All levels share a single name space, but views of that name space differ according to the network security level accessing the TSE.

Read-down eliminates the need for low-security data to be explicitly copied for users at high security. The single name space combined with read-down makes a wide range of applications and user workflows easier, more dynamic, and less error-prone than existing solutions.

Developing, certifying, and evaluating a high assurance cross-domain component such as the TSE at acceptable cost requires a fundamentally different architecture from that of typical, single-level

components. Our approach is the following: Use as few high-assurance components as possible, each with a single purpose, to keep it small and simple, allowing it to be analyzed formally. But security is a property of a whole system, not just a component. Appropriate composition techniques can extend the security properties of the trusted computing base outward to the rest of the system.

The TSE's trusted computing base consists of the minimum number of components: one. TSE functionality is decomposed into a set of single-level components and only one cross-domain component. The underlying MILS separation kernel separates components at different security levels. Each network security level has a set of clients, an authentication service, and an integrity checker (see Figure 1). Within the TSE, each network level has its own network interface card, hard drive, and software stack implementing the TSE's networking, WebDAV, and file system services.



- Mediate all disk block access.
- Connect single-level disks and partitions.
- Write blocks to the same level.
- Read blocks from the same or lower levels.

The keys to BAC security are that it has a well-defined job and is constructed from very few lines of code. The current version of the BAC is 780 lines of C code. To ensure that the BAC implements the required attributes, we do the following:

1. Develop a formal model of the code.
2. Verify that the model corresponds to the code.
3. Develop a formal model of the policy.
4. Use model-based testing to check that the code implements the policy.
5. Formally verify that the model implements the policy.

Our formal verification ensures that the TSE security policy maps directly to the model, and the model to the implementation. To map the policy to the model, we use the Isabelle Higher Order Logic (HOL) theorem prover [3]. The theorems we prove in this logic are the following:

- None of the error states are reachable.
- The noninterference property holds.

The noninterference property states that all system actions by high security-level components are invisible to low security-level components; that is, the final state of the low-level component is the same as it would be if no actions had occurred at the high-security level.

To map the model to the implementation, a code-to-spec review team of at least two people performs a line-by-line inspection of the HOL code and the C implementation.

The example in Table 1 – a single step of the BAC – shows how closely the model matches the implementation. Our model-based testing approach uses the QuickCheck tool [4]. Based on a formal statement of the security policy, QuickCheck generates test cases that check whether or not the implementation violates that policy. The policies we have verified using this method are the following:

- **Read-across:** Reads fetch the data written at that same level.
- **Read-down:**
 - Valid reads succeed.
 - Invalid reads (that is, read-up) fail.
 - Read-downs do not affect the lower level being read (noninterference).

HOL Model	C Code
<pre> bacStep :: "config => (unit, store) m" "bacStep conf == let n = numLevels conf in processQueuedLevels (requestsPerLevel conf) n >> queueLevels conf n" </pre>	<pre> void bacStep (config conf) { nat n = conf->numLevels; processQueuedLevels (conf->requestsPerLevel, n); queueLevels(conf, n); } </pre>

Table 1: A Single Step of the Block Access Controller

Other Key Components

MILS Separation Kernel

The BAC, when hosted by the MILS separation kernel [5, 6], is an instantiation of the *reference monitor* concept [7]. Unlike a traditional operating system that provides many services and abstractions, a separation kernel provides only data isolation among separate partitions and controlled communication between *partitions*. Porting an application to MILS also requires choosing a runtime or operating system to run within each partition that provides the higher-level system services the application requires, or porting one of your own choosing.

It is not enough simply to port a single-level application to a MILS separation kernel, however. The system needs to be thoughtfully decomposed and mapped to MILS partitions. Further, some key components (such as the file system) may need to be radically restructured to function in a multilevel environment.

While the TSE project aims to be portable across separation kernels, the initial target is Green Hills Software's INTEGRITY Server. This platform allows us to deploy software components from different security levels on the same hardware, thus reducing space, weight, and power requirements while retaining isolation properties equal to those provided by networks on physically separate hardware.

The WebDAV Server

The single-level components of the TSE are the WebDAV server, the file system, the network stack, and the secure sockets layer/transport-layer security (SSL/TLS). To provide the security aspects of WebDAV with high assurance, we implemented the WebDAV server using Haskell, a type-safe functional language [8]. We ported the Haskell runtime system to INTEGRITY server. The Haskell runtime system encapsulates services such as networking, threading, and memory management.

The Wait-Free File System

As Figure 1 shows, the TSE file system is a single-level component. We were surprised to find that no existing single-level file system met our requirements. The problem is caused by read-down – a user on a high-level network can read files from a lower level while a user on the low-level network changes those files. Ordinarily, locks could be used to solve this problem, but cross-domain locks violate non-interference and are unacceptable in this case. How can the TSE present consistent data without introducing a proscribed communication channel, overt or covert?

Designers of algorithms for shared-memory multiprocessors face a similar problem that they solve using a method called *wait-free synchronization* [9]. Wait-free synchronization guarantees that interactions with concurrent objects take a finite number of steps instead of using *critical sections*, which block competing processes for an indeterminate time. The *wait-free file system* adapts this idea for its own synchronization method. This preserves the isolation property by the following:

- Writers are oblivious to readers.
- Readers can proceed independently of writers.

Outside Services

To minimize the trusted base and avoid duplication of function, the TSE will use, or uses outside services wherever possible. Key services are authentication and integrity-checking; so far we have evaluated Navy enterprise single sign-on for authentication and one-way file transfer for integrity-checking, but final decisions will be driven by the demands of specific installations at customer sites.

Though it is conservative and efficient to draw on outside services, it also means that we must build a chain of trust from our base to the outside service. We use several methods to help us do so:

- Outside services are all single-level, which minimizes their trustworthiness requirements.
- We choose services specified and trusted by our customers that have been vetted in similar deployment scenarios.
- The TSE and companion services use the standard cryptographic protocols SSL/TLS and digital certificates to manage communication between them.

The sum of the TSE and a specific set of external services is submitted for the certification prerequisite to multilevel deployment.

Building Complex Multilevel Services on the TSE

The TSE can be used as a building block for more complex cross-domain services, as demonstrated by another current Galois project, the Multilevel Document Collaboration Server (DocServer). Its architecture reuses the decomposition structure of the TSE to provide multilevel secure document-based collaboration.

The DocServer allows a user at a high network level to make private modifications to an XML-based document stored at a lower level. The DocServer supports ongoing modifications at multiple network levels; modifications from the high network are visible only to users on the high network, while modifications from the low network are visible to users at that level and above.

The DocServer also supports publishing regraded documents from high network levels to low, using XML filtering and integration with an outside regrading system such as Radiant Mercury or ISSE Guard. These systems enable transfer of documents from high security to low security by enabling a human reviewer to reliably review all of a document's contents (including possibly hidden content), and, upon successful review, write it to the low network.

In the case of the DocServer, a high-level user marks up the document according to a new set of security levels, and submits it for regrading. The DocServer filters the document and sends the filtered version to the regrading system. After human review, the filtered version of the document is written to the DocServer's low-level file system.

Figure 2 shows the *publish, edit, merge* workflow of the DocServer. At left, a user on the Secret network publishes the document to the Unclassified network. The DocServer filters the Secret content and submits the resulting unclassified document to the regrader. After regrading, users on both network levels make modifications to the document. Modifications made at Secret are not visible below, but Unclassified modifications are visible to users at Secret using the DocServer's *merge* each time the document is read.

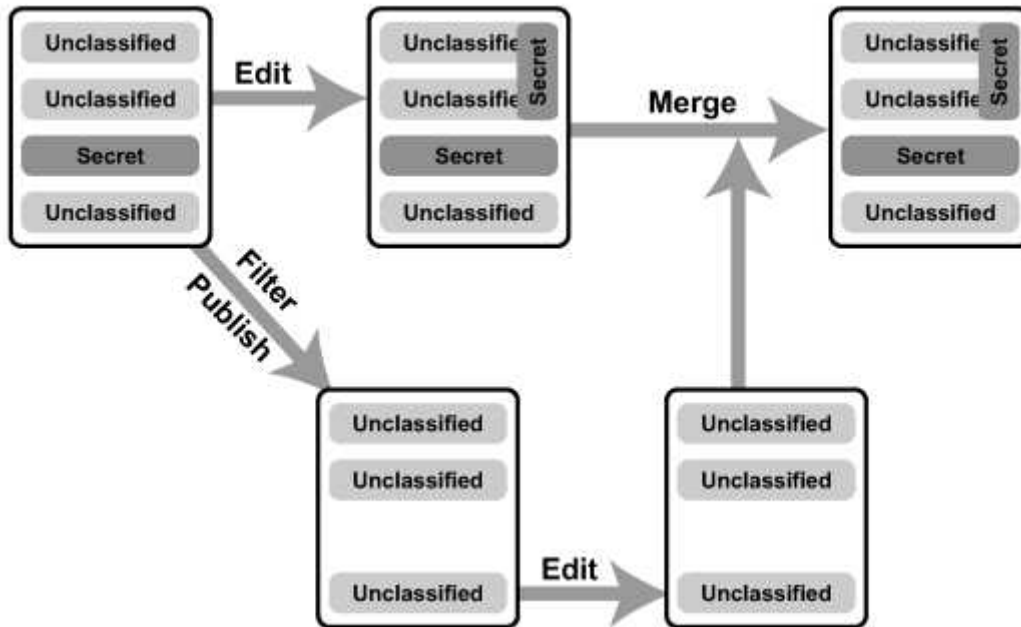


Figure 2: *DocServer Merge Operations*

The DocServer is a Phase 1 Small Business Innovative Research project funded by SPAWAR.

Conclusion

The DocServer uses the TSE for file storage and its sole cross-domain component. Reusing the only high-assurance component gains us a great deal – the DocServer should be certifiable to the same level as the TSE with little additional work.

The DocServer's use of the TSE to achieve high assurance, cross-domain function mirrors the TSE's internal use of the BAC. By building the DocServer from this core component, we once again take advantage of the BAC, effectively extending its security policy through to increasingly complex systems.

The TSE's component architecture demonstrates a powerful technique for extending the security properties of a formally analyzed core component to a wide scope. In a similar manner, the DocServer uses MILS to extend the security properties of the TSE outward to provide complex multilevel functionality.

TSE Status

Development of Vers. 1.0 of the TSE will be complete in summer 2006, and will be followed by certification at a customer site. We expect to begin Common Criteria evaluation at evaluation Level 6+ the following year. Phase 1 of the DocServer is near completion. We hope to begin Phase II in spring 2006, and commercial transition sometime in 2007.

Acknowledgements

The authors would like to acknowledge contributions from the following people: David Burke with the evaluation and certification sections; John Matthews and Paul Graunke with the verification and validation sections; and Lauren Ruth Wiener with the clarity of thought and exposition.

References

1. Common Criteria <www.common-criteria-portal.org>.
 2. D. E. Bell and L. J. LaPadula. *Secure Computer Systems: Mathematical Foundations and Model*. The Mitre Corporation, 1976. <<http://csrc.nist.gov/publications/history/bell76.pdf>>
 3. Isabelle. "A Proof Assistant for Higher-Order Logic." University of Cambridge Computer Laboratory <[www.cl.cam.ac.uk/Research/HVG/ Isabelle](http://www.cl.cam.ac.uk/Research/HVG/Isabelle/)>.
 4. QuickCheck Automatic Specification-Based Testing <www.cs.chalmers.se/~rjmh/QuickCheck>.
 5. National Information Assurance Partnership. *U.S. Government Protection Profile for Separation Kernels in Environments Requiring High Robustness*. Vers. 0.621. Ft. Meade, MD: NIAP, July 2004 <http://niap.nist.gov/pp/draft_pps/pp_draft_skpp_hr_v0.621.html>.
 6. Vanfleet, Mark W., et al. "MILS: Architecture for High-Assurance Embedded Computing." Aug. 2005 <www.stsc.hill.af.mil/crosstalk/2005/08/0508vanfleet_etal.html>.
 7. Anderson, James P. "Computer Security Technology Planning Study." Fort Washington, PA: James Anderson & Co, Oct. 1972 <<http://csrc.nist.gov/publications/history/ande72.pdf>>.
 8. Haskell. Haskell: A General-Purpose Purely Functional Language <www.haskell.org>.
 9. Herlihy, Maurice. "Wait-Free Synchronization." *ACM Transactions on Programming Languages and Systems (TOPLAS)* 1: 124-149. New York: ACM Press, Jan. 1991 <<http://portal.acm.org/citation.cfm?id=102808>>.
-

About the Authors



Dylan McNamee, Ph.D., is the technical lead for cross domain projects at Galois Connections. He received his doctorate in computer science from the University of Washington.

Galois Connections
12725 SW Millikan WY
STE 290
Beaverton, OR 97005
Phone: (503) 626-6616 x137
E-mail: dylan@galois.com



CDR Scott Heller is currently the Cross Domain Solutions lead at PMW 160 within the Program Executive Office Command, Control, Communications, Computers, and Intelligence, and Space in San Diego, Calif. He has a master's degree in computer science with an emphasis in Multi-level Security from the Naval Post-Graduate School in Monterey, Calif.

Program Executive Office
C41 and Space
626 Orange AVE #303
Coronado, CA 92118
Phone: (619) 929-1451
E-mail: scott.heller@navy.mil



Dave Huff serves as the director, Exploratory Projects Division at the Fleet Numerical Meteorology and Oceanography Center. His team is focused on information assurance and Web-based techniques for establishing identity, authorization, and cross-domain information exchange.

Fleet Numerical Meteorology and Oceanographic Center
7 Grace Hopper AVE
Monterey, CA 93943
Phone: (831) 656-4569
E-mail: dave.huff@metnet.navy.mil