information

Article

Using Proven Reference Monitor Patterns for Security Evaluation

Mark R. Heckman [1,*] and Roger R. Schell [2,*]

[1] Center for Cyber Security Engineering and Technology, University of San Diego, San Diego, CA 92110, USA
[2] Aesec Corporation, Palo Alto, CA 94301, USA
* Correspondence: mheckman@sandiego.edu (M.R.H.); schellr@ieee.org (R.R.S.); Tel.: +1-619-260-2947 (M.R.H.)

Academic Editor: Willy Susilo
Received: 12 February 2016; Accepted: 14 April 2016; Published: 26 April 2016

Abstract: The most effective approach to evaluating the security of complex systems is to deliberately construct the systems using security patterns specifically designed to make them evaluable. Just such an integrated set of security patterns was created decades ago based on the Reference Monitor abstraction. An associated systematic security engineering and evaluation methodology was codified as an engineering standard in the Trusted Computer System Evaluation Criteria (TCSEC). This paper explains how the TCSEC and its Trusted Network Interpretation (TNI) constitute a set of security patterns for large, complex and distributed systems and how those patterns have been repeatedly and successfully used to create and evaluate some of the most secure government and commercial systems ever developed.

Keywords: security pattern; Reference Monitor; security kernel; TCSEC; Orange Book; TNI; Mandatory Access Control (MAC)

1. Introduction

The conventional wisdom today seems to be that it is impractical to design and develop a highly secure complex system, much less evaluate its security, but this quite widely-spread assertion has been repeatedly disproven by counterexamples from both real systems and research prototypes. As the counterexamples have shown, the most effective approach known for engineering and evaluating the security of complex systems is to deliberately construct them using security patterns specifically designed to make the systems evaluable.

While much effort has been expended in creating security patterns—structured, repeatable, solutions to software design problems—progress in demonstrating the usefulness of security patterns for the development and evaluation of large, complex, distributed, and secure systems has been slow [1]. One author noted:

> The very first reference to design patterns in a security context was a presentation by Deborah Frincke at NISSC 1996. The presentation discussed the security ramifications of some of the original 23 Design Patterns. Her basic idea was that one could start with a security requirement on a pattern, and decompose it into constituent object requirements. In this way, if a proof were developed at the pattern level, it could be reused in tandem with the pattern. This was a powerful idea, but limited to patterns that were not originally intended to address security requirements. Unfortunately, this work was not continued, and was never published.
>
> —Darrell M. Kienzle, *et al.* [2]

Despite this gloomy viewpoint, security patterns based on the Reference Monitor concept [3] were successfully used to evaluate secure, distributed systems beginning in the late 1970s, although most of these patterns were not recognized as such. Formal patterns written in a pattern language have been created for multilevel security and the Reference Monitor itself [4], other important patterns, such as the systematic composition evaluation methodology described in the Trusted Network Interpretation (TNI) of the Trusted Computer Security Evaluation Criteria (TCSEC) [5], have not been expressed in a pattern language, even though that formalized composition strategy is clearly a pattern in the sense that it is a structured, repeatable, solution to a software design problem. The composition strategy of the TNI is, in fact, a fully realized instance of Frincke's pattern decomposition idea in that it starts with an overall set of system security requirements and decomposes them into constituent object requirements.

The goal of this paper is to identify these previously unrecognized security patterns, which have been shown to be so effective for systematic, repeatable, systems-oriented security evaluation of large, distributed, complex systems, and to encourage future systems developers to confidently apply the proven security patterns to the evaluation of new systems.

2. Definitions and Context

A primary technical approach in these patterns is to use a high assurance Reference Monitor [6] (Sections 4.1 and 6.1) to enforce well-understood Mandatory Access Control (MAC) properties of the security policy of the global system, based on a formally-defined and proven Formal Security Policy Model [6] (Section 6.2). The model abstractly defines a secure state and the security-preserving operations on that state. A complex integrated system is composed, in conformance with the security patterns, from logically distinct components in multiple, diverse security domains. Those distinct components may have various degrees of security assurance or no assurance at all; the MAC properties precisely define the constraints between the various security domains of the system. The security of this composition of components, *i.e.*, the system, is evaluated by systematically verifying that it is a "valid interpretation" of the model, that is, by showing that the system has only valid operations defined by the model.

2.1. The Reference Monitor Subjects and Objects Foundation

Security for cyber systems built without a trustworthy Operating System (OS) is simply a scientific impossibility. The proven scientific principle of the "Reference Monitor", whose implementation is called a "security kernel", enables engineering a verifiably secure OS. The Reference Monitor [3] is an abstraction that allows active entities called subjects to make reference to passive entities called objects, based on a set of current access authorizations. A security kernel (hardware and software) ensures that every reference by a subject to information in an object (e.g., by a processor to primary memory) or change of authorization must go through the Reference Monitor. This mediation by the Reference Monitor is depicted in Figure 1.

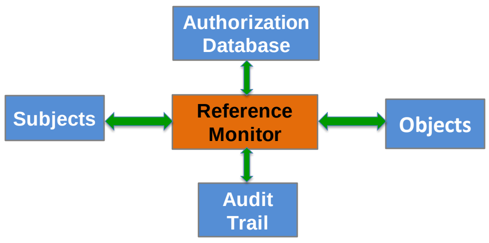

Figure 1. Reference Monitor concept.

Implementing the Reference Monitor security pattern in the form of a security kernel requires enforcing rigorous distinguishing properties for subjects and objects in all accesses performed by a computer system. A subject is an active entity, generally in the form of a process or device that causes information to flow among objects or changes the system state. An object is a passive entity that contains or receives information. Access to an object potentially implies access to the information it contains.

To successfully implement a security kernel, one must adhere to three engineering principles: (1) *completeness*, in that all access to information must be mediated by the kernel; (2) *isolation*, in that the kernel must be protected from tampering; and (3) *verifiability*, in that some correspondence must be shown between the security policy and the actual implementation of the kernel. Unfortunately, today's engineers developing the OSs used in most systems have largely failed to apply the Reference Monitor principles, perpetuating the security problem. The unending stream of OS security patches is evidence that every widely-used OS is untrustworthy. Such an OS is not able to protect a sensitive domain from unauthorized domains when facing a witted adversary, who may use techniques, such as software subversion.

The current widespread security "best practices", which are based on frequent patching, monitoring and surveillance, are part of an arms race we cannot win. In contrast, the transformative nature of Reference Monitor cyber security is evident in both the underlying science and in practical experience. Although rarely applied in today's systems, the security kernel is not a new idea. A leading security scientist as early as the 1970s understood the scientific theory and said, "The only way we know ... to build highly secure software systems of any practical interest is the kernel approach [7] (p. 23)." At least half a dozen security kernel-based operating systems [8] (Slide 4) ran for years (even decades) in the face of nation-state adversaries without requiring a single reported security patch. Furthermore, as we will discuss below, the means by which the Reference Monitor leverages its distinguishing abstractions of active subjects and passive objects provides a unifying theory foundation for powerful security patterns that can be used for the evaluation of the security of systems.

2.2. A Secure System Must Mitigate Adversarial Attacks, Including Subversion

Customers in many cases would not consider a system to be secure if it did not substantially address attacks likely to be used by a witted adversary, such as subversion. Several cyber security professionals have concluded that subversion "is the attack of choice for the professional attacker [9]." Subversion of a computer system's security mechanism involves the covert and methodical undermining of internal system controls to allow unauthorized and undetected access to information within the computer system. Such subversion is not limited to on-site operations. It includes opportunities that spread over the entire life cycle of a computer system, including: (1) design; (2) implementation; (3) distribution; (4) installation; and (5) use.

A subversive artifice will typically include the capability of activation and deactivation. It was recognized long ago, based on experience with information warfare demonstrations, that "deliberate flaws are dormant until activated by an attacker. These errors can be placed virtually anywhere and are carefully designed to escape detection [9]". The artifice activation mechanism waits for the presence of some unique trigger. Subversion of this type is illustrated in contemporary, coordinated cyber-attacks, including Stuxnet [10]. There are few, if any, approaches besides the Reference Monitor for effectively addressing subversion.

The threat of subversion in a system that implements a Reference Monitor is two-pronged: an attacker can attempt to subvert (1) applications that run on the security kernel or (2) the kernel itself. In the first case, the "completeness" principle of the Reference Monitor ensures that an application cannot violate the system access control security policy and access any information not permitted by the policy (although a subverted application may attack the assets of the application itself, even this can be mitigated by integrity policies). In the second case, the isolation principle of the Reference Monitor ensures that the protection mechanisms in the security kernel are themselves protected from tampering. In order to reliably implement all three principles of the Reference Monitor, a security kernel must be a high-assurance system built from the ground up using a rigorous system development and evaluation methodology.

2.3. Encompassing Codification for Evaluation

In most areas of information technology, testing is usually sufficient for determining compliance with system specifications. Testing alone, however, is insufficient for demonstrating the security of a platform. It is a fundamental scientific limitation that testing can only demonstrate the lack of security by exposing specific flaws, but can never demonstrate the absence of flaws (this is a refinement of Dijkstra's general observation that "Program testing can be used to show the presence of bugs, but never to show their absence [11]!"). Simply offering a platform with an absence of *known* flaws does not satisfy the need for measurable trust.

The evaluation conundrum is how to verify the negative assertion that the system has no flaws. The solution is a model-based method to evaluate security. The linchpin in providing the desired verifiable protection is a Formal Security Policy Model that describes the core access control security policy to be enforced by the implementation of the Reference Monitor, *i.e.*, the security kernel. The formal security policy abstractly models all of the functions of the security kernel (its application program interface, or API) as a set of rules that change its internal state. All of the possible states are mathematically proven to be sound and correct with respect to the desired security policy. The system interface operations of the security kernel are shown to each correspond to one or more rules that represent the security policy model. All elements of the system must be demonstrated to be both necessary and sufficient for enforcement of the policy. Using the model, the implementation is systematically traced to the policy, an effort that includes showing that all of the code that is in the kernel is required for policy enforcement and system self-protection. This process is commonly termed "code correspondence" and is specifically intended to mitigate subversion. The model enforced by most security kernels is derived from what is commonly referred to as the "Bell and LaPadula Model" [12]. This model provides rules that are provably sufficient for preventing unauthorized observation and modification of information.

During the decade following the invention of the Reference Monitor, substantial additional research produced working prototypes, demonstrations and systematic engineering processes for implementation and evaluation. Through extensive collaboration with government and industry stakeholders, these processes were carefully codified into a technical standard specifically designed to be used to evaluate the security of systems. The resulting TCSEC [6] specifies in detail a set of patterns (informally, in the sense of structured, repeatable processes) for such evaluation. This standard explicitly identified the Reference Monitor concept as part of its rationale. To support evaluation, it required that "documentation shall describe how the TCB implements the Reference Monitor concept and give an explanation why it is tamper resistant, cannot be bypassed, and is correctly implemented".

The TCSEC had well-defined levels of security, called classes. The seven classes are divided into four divisions of increasing security assurance, termed D, C, B and A. We have above emphasized the importance of high assurance to address the clear and present danger from attacks likely to be used by a witted adversary, such as subversion. In terms of the TCSEC evaluations, it was recognized that "Division A is distinguished by additional requirements substantially dealing with the problems of subversion of security mechanism. To gain increased assurance beyond division B, one needs more structured verification support tools [13]". Since only Division A substantially deals with subversion, and since this division has only one class, Class A1, only Class A1 systems have the high assurance necessary to address subversion.

Class A1 [6] (Section 4.1) includes comprehensive requirements unique to this class that specifically focus on the problem of subversion of the security kernel itself. These requirements include the following development processes and artifacts, in the sections shown:

- Formal Top-Level Specification (FTLS) [6] (Section 4.1.4.4). The FTLS precisely defines the security kernel API as the basis for a high degree of assurance that the implementation is correct.
- Show that FTLS is consistent with the model [6] (Section 4.1.3.2.2). This is evidence in the form of a mathematical proof that both the model and kernel interface are correct representations of the security policy.
- Formal covert channel analysis [6] (Section 4.1.3.1.3). Formal analysis techniques must be used to identify and analyze covert channels.
- Penetration testing based on the FTLS [6] (Section 4.1.3.2.1). Penetration is based on mapping of the FTLS to source code.
- Hardware and firmware included in FTLS [6] (Section 4.1.3.2.2). This provides a complete definition of the execution environment for applications running on the kernel.
- Code correspondence [6] (Section 4.1.3.2.1). Mapping of the FTLS to kernel source code to provide evidence of correct implementation and the absence of artifices from subversion.
- Strict configuration management [6] (Section 4.1.3.2.3). Strict configuration control to address subversion of tools and design documentation, as well as of source and object code.
- Trusted distribution [6] (Section 4.1.3.2.4). This addresses supply chain subversion by confirming that each customer's kernel software, firmware and hardware are exactly what were evaluated.

Even for lower assurance systems, for which this level of effort is not required, the TCSEC methodology provides a major benefit by helping users understand the level of assurance the system provides.

It is worth emphasizing that the raison d'être for the TCSEC is to provide a method to evaluate the security of an entire *system*, not just a component, as is the case for some other approaches like the Common Criteria [14]. The TCSEC evaluation method successfully exploits the system design pattern for Reference Monitor subjects and objects. A distinguishing TCSEC pattern is the concept of a Trusted Computing Base (TCB), which it defined as follows:

> The totality of protection mechanisms within a computer system—including hardware, firmware, and software—the combination of which is responsible for enforcing a security policy. A TCB consists of one or more components that together enforce a unified security policy over a product or system. The ability of a trusted computing base to correctly enforce a security policy depends solely on the mechanisms within the TCB and on the correct input by system administrative personnel of parameters (e.g., a user's clearance) related to the security policy.
>
> —National Computer Security Center [6]

2.4. Well-Defined Security Software Engineering Process

After a number of years where the TCSEC was successfully used to design, build and evaluate systems, there are a number of working examples that can serve to inform the application of its implicit patterns. We will not reiterate all of the details, but highlight a couple of particularly significant considerations.

The TCSEC discussion above has noted the important role of mapping from the delivered software in a security kernel all the way up to an abstract access control security policy. The essential elements of the TCSEC secure system engineering mappings are illustrated in Figure 2. This engineering process is especially important to security evaluation, because it provides a critical part of a proof sketch to demonstrate that all of the software in the completed product is a valid interpretation of the kernel API specification derived directly from the "Formal Security Policy Model" discussed earlier.

From the TCSEC, there is a pattern for each of the mappings shown in Figure 2. One especially important mapping that is a specific application of traditional software engineering is the mapping labeled "Source code layering and info hiding". The "System Architecture" requirements of [6]

(Section 4.1.3.1.1) of the TCSEC provides much of the basis of the mapping. In fact, meeting its requirement for "layering, abstraction and data hiding" is essentially a litmus test for whether (or not) what is developed has effectively applied this Reference Monitor-based security pattern. A well-written paper on engineering a commercial security kernel cites this requirement and describes the pattern for layering as follows:

> Each layer of the design implements some abstraction in part by making calls on lower layers. In no case does a lower layer invoke or depend upon higher layer abstractions. By making lower layers unaware of higher abstractions, we reduced the total number of interactions in the system and thereby reduce the overall complexity. Furthermore, each layer can be tested in isolation from all higher layers, allowing debugging to proceed in an orderly fashion, rather than haphazardly throughout the system.
>
> —Paul A. Karger, *et al.* [15] (p. 1154)

The same paper also points also out the importance of another software engineering decision, the choice of programming language: "Perhaps the most critical software engineering issue in the Security Kernel design was the choice of a programming language. . . . We wanted as strongly typed a language as possible".

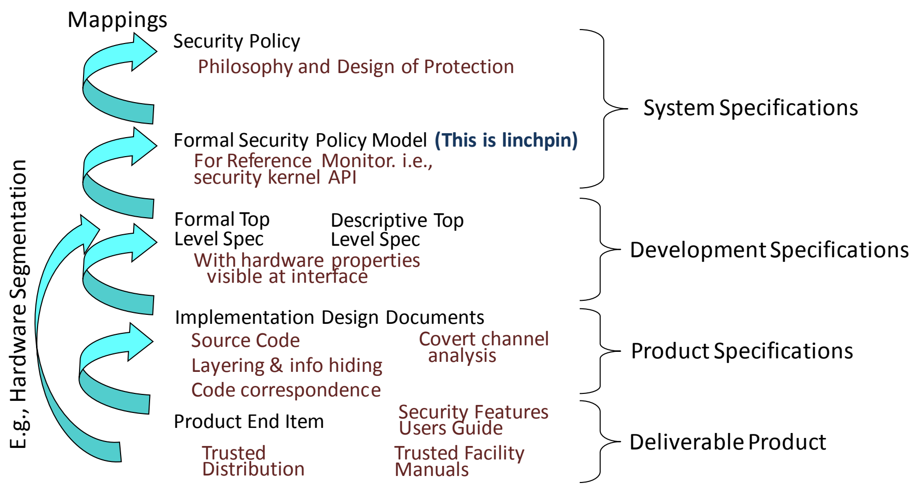
Mappings
Security Policy
Philosophy and Design of Protection
System Specifications
Formal Security Policy Model (This is linchpin)
For Reference Monitor. i.e., security kernel API
E.g., Hardware Segmentation
Formal Top Level Spec
Descriptive Top Level Spec
With hardware properties visible at interface
Development Specifications
Implementation Design Documents
Source Code
Layering & info hiding
Code correspondence
Covert channel analysis
Product Specifications
Product End Item
Security Features Users Guide
Trusted Distribution
Trusted Facility Manuals
Deliverable Product

Figure 2. Systematic security kernel engineering process.

2.5. *MAC Policy Is Key to Security Architecture of Large Systems*

The Reference Monitor concept is not defined by the security policy, nor does it define the security policy. The Reference Monitor concept is compatible with a broad range of security policies. All access control policies fall into two classes: Discretionary Access Control (DAC) and MAC. It was recognized early on that "protection against malicious software is offered only by an implementation of the Reference Monitor concept enforcing Mandatory Access Control policies, though the Reference Monitor paradigm . . . is also used for discretionary access control [16]." Since a major goal of this paper is to include evaluation for security capabilities to mitigate adversarial attacks, including subversion, our examples and discussion will for didactic reasons in many cases specifically mention MAC.

A MAC policy provides an overriding constraint on the access of subjects to objects, with high assurance of protection possible, even in the face of Trojan horses and other forms of malicious software. MAC policies can provide protection against unauthorized modification of information (integrity), as well as protection against unauthorized disclosure (confidentiality). In terms of the Reference Monitor concept, the idea is that we can affix a label to objects to reflect the access class of the information they hold. We can correspondingly affix a label to subjects to reflect the equivalence class of object sensitivity that the subject can access. The Reference Monitor compares the labels on subjects and objects and grants a subject access, per the requested access mode, to an object only if the result of the comparison indicates that the access satisfies the MAC policy.

The distinguishing qualities of MAC policies are that they are global and persistent within some universe of discourse; these qualities enable verifiability of the Reference Monitor implementations that enforce them. In this context, "global" means that particular information has the same sensitivity wherever it is; "persistent" means that particular information has the same sensitivity at all times. These properties are key to applying the Reference Monitor-based security patterns to large systems. The quintessential example of need for a MAC policy is the pervasive government requirement for multilevel security (MLS), where information is at multiple levels, such as confidential, secret and top secret, and where the universe of discourse is the entire U.S. government. (Note that MAC policies are not limited to government. A commercial enterprise might also be large and geographically dispersed and have a similar need to manage information at multiple levels.)

2.6. Systematic Composition Is Key to Distributed Systems

One reason why evaluating the security of real-world systems is a complex endeavor is that systems are distributed over many diverse components that are part of a network. Methods for evaluating the security of a network system need a systematic way of representing a secure network. This issue of evaluating the security of a network was the specific purpose of the Trusted Network Interpretation (TNI) of the TCSEC. Section I.4.2 of the TNI described its approach as follows:

> Like a stand-alone system, the network as a whole possesses a single TCB, referred to as the NTCB, consisting of the totality of security-relevant portions of the network. But, unlike a stand-alone system, the design and evaluation of the network rests on an understanding of how the security mechanisms are distributed and allocated to various components, in such a way that the security policy is supported reliably in spite of (1) the vulnerability of the communication paths and (2) the concurrent, asynchronous operation of the network components.
>
> —National Computer Security Center [5] (Section I.4.2)

Such a network system is simply a composition of components. However, doing composition "in such a way that the security policy is supported reliably" (in the words of the TNI) is the nub of the evaluation problem: What are the security properties of a system that consists of a composition of interconnected components? No general, closed-form, engineering-free solution to this problem has been found, and one may not be possible [17]. Today, only two security composition methods have been proven to work: "TCB subsets" and "partitioned TCB" [18]. The system functionality available by employing these patterns is more a matter of innovative engineering and art than science. The good news is that as a matter of practical experience, few, if any, common Internet-based capabilities seem blocked by being limited to these two composition methods.

2.6.1. TCB Subsets

It is often beneficial to develop a TCB and to evaluate its security in parts, rather than as a monolith. A TCB subset is an extension of the Reference Monitor concept that enforces some access control policy on subjects that may attempt to gain access to a set of objects under the subset's control. The access control policy enforced by a TCB subset is a subset of the overall access control policy of the TCB. Access control policies of distinct TCB subsets are enforced by distinct, hierarchically-ordered protection domains within the TCB [18].

Because the Reference Monitor implementation (defined as a security kernel) controls the physical hardware, it is in the most privileged domain. The Reference Monitor typically implements the least restrictive MAC policy in a TCB. Other TCB subsets within that TCB may implement other, more restrictive access control policies, including more restrictive MAC policies and DAC policies. These TCB subsets are within the TCB but outside the Reference Monitor.

A couple of evaluation considerations are worth mentioning. Subsets require the rigorous imposition of the constraints of the pattern for a "subject". This is reflected in the TCSEC glossary definition of a subject as "Technically, a process/domain pair". That means that a process with TCB subsets can have multiple subjects in the same process. The most direct and common way to implement the required hierarchical protection domains is using hardware that supports classic "protection rings". Fortunately, support for rings is included in the ubiquitous Intel x86 instruction set architecture.

Another important security pattern enabled by TCB subsets is what has been called "balanced assurance". Supporting TCB subsets means the designer has the ability to have a lower level of assurance for subsets in less privileged domains. The layered TCB satisfies the following key requirements: The TCB is organized into two or more layered protection domains, with lower layers tamper-proof with respect to higher layers (including both applications and outer TCB layers). Each TCB layer has a clearly expressed security policy that it alone enforces. This enables designers and evaluators to balance the organizational benefits of a policy category with the costs of implementing assurance techniques for that subset. A common use is to place lower assurance DAC enforcement in a higher layer than the more important MAC enforcement.

2.6.2. Partitioned TCB

The overall network system security policy is the security policy of the NTCB. It may be decomposed into security policy categories that are allocated to appropriate components and used as the basis for the security policies for those components. This technique is referred to as partitioning a network's security policy into component security policies, or NTCB partitions. In the TNI, a component is a collection of hardware, firmware and software that performs a specific security function in a network and that contains an NTCB partition. The distinguishing quality of a component is that it has a security policy partition allocated to it, induced on it by the overall network security policy.

Abstractly, the Reference Monitor for a partitioned NTCB is realized as a collection of security kernels for individual components. Morrie Gasser describes this [19] (p. 274) as a "technique to keep the kernels as disjoint as possible, relying on each kernel to enforce subject-to-object access within its own computer system and minimizing the amount of trusted control information that must be exchanged between kernels". To obtain the required levels of assurance that each security kernel enforces its security policy, a formal security policy model must be formulated for each component. However, it would be too restrictive to require that the formal security policy model for each security kernel be the same, or that an overall formal security policy model be formulated for the network. Instead, each formal security policy model is shown by convincing arguments to correctly represent the security policy allocated to the component.

A pivotal step in the partitioned TCB composition method is the security pattern for allocating portions of the overall system security to each of the components that are composed to make up the overall networked system. The TNI in Section A.1.1 defines a taxonomy [5] for network components based on subordinate policy elements to be enforced as follows:

> In the context of a larger system, however, it may well be (and usually is) the case that the set of policy-related features to be supported by the component need not be the complete set required for a stand-alone system: features not supplied by one component for the system are supplied by another ... In order to limit the number of component types we break the "maximal" set of policy-related features, defined by the TCSEC for A1 systems, into four relatively independent categories which can be characterized as supporting Mandatory Access Controls (MAC), Discretionary Access Controls (DAC), Audit, and Identification and Authentication ... these categories will be given the one-letter designations M, D, A, and I, respectively.
>
> —National Computer Security Center [5] (Section A.1.1)

To provide a helpful mnemonic, this is often referred to as the "MAID" taxonomy. Although there is nothing fundamentally scientific about this particular taxonomy choice of four categories, they were carefully chosen to be easily related to the components of the Reference Monitor abstraction. Remember that, although the patterns described in this paper could generally be applied to any of these four categories, our emphasis is on MAC.

A powerful pattern for evaluating the MAC security of a network system is applied to the connections between components. Section 4.1.1.3.2.1 of the TNI defines this pattern when saying [5] that, "The components, including hosts, of a network shall be interconnected over 'multilevel communication channels', ... The protocol for associating the sensitivity label and the exported information shall provide the only information needed to correctly associate a sensitivity level with the exported information transferred over the multilevel channel between the NTCB partitions in individual components". Morrie Gasser points out the essence of this multilevel channel pattern:

> It turns out that, for a multilevel security policy, the only trusted control information that needs to be exchanged is the access class of the network objects; the transmitting kernel inserts an access class label on a message based on the access class of the subject that created the message, and the receiving kernel uses that access class to determine who on its own system may read the message. If the trusted network service operates at the transport layer, a label is required only when the virtual circuit is established, and not on each message that is exchanged over that circuit. The kernels must still trust each other to insert and obey the access class labels according to their common policy, but no information other than the labels need be managed or exchanged between the kernels.
>
> —Morrie Gasser [19] (Section 13.5)

Just as with TCB subsets, a partitioned TCB also enables "balanced assurance". Providing this taxonomy for a partitioned TCB provides designers and evaluators the pattern to use to balance the organizational benefits of a policy category with the costs to implement assurance techniques for the physical component that support that policy. Appendix A of the TNI [5] provides substantial detail for answering the key question "How can the composition of rated components be evaluated?" It does this by providing specific patterns for composing components of possibly different levels of assurance "in such a way that evaluation of individual components will support eventual evaluation of the entire network".

3. Results

The purpose of our work is to identify security patterns that have been shown to be effective as the basis for a systematic, repeatable, systems-oriented methodology for security evaluation of large, distributed, complex systems for both government and commercial use. Our results are from the application of these security patterns to several quite different systems. We have deliberately chosen diverse and unrelated systems as examples in order to have a reasonably representative sample of the diversity and complexity of the systems to which the patterns could be applied. These were not experiments conducted by the authors in some sterile laboratory, but were in the context of the design and evaluation of complex, real-world systems. The subsections below, each of which present the results from a different example, make extensive use of openly published results.

3.1. Commercial Secure Oracle DBMS System

Oracle Corporation has for many years offered a market-leading commercial Database Management System (DBMS). In the years through the 1980s, security customers, including the government, noted that no available DMBS, including Oracle, provided high assurance (e.g., TCSEC Class A1) enforcement of a MAC policy, including MLS. Linda Vetter, a senior security professional at Oracle, summarized the problem as follows:

In building multilevel database systems, providing such assurance is especially challenging because large, complex mechanisms may be involved in policy enforcement. Achieving Class A1 assurance for a multilevel database system is possible only if the portion of the system enforcing mandatory security is small and isolated and does not depend on the large database machinery for its correct enforcement.

—Linda Vetter, *et al.* [20]

After years of substantial research investment, the accepted conventional wisdom at that time was that a highly secure MLS DBMS was an oxymoron. Then, in the mid-1980s, a research project called SeaView collaborated with Oracle to explore a novel approach to the MLS DBMS problem, applying Reference Monitor based patterns discussed above. The resulting SeaView architecture was successful beyond most expectations. The SeaView architecture is illustrated in Figure 3. The "Trusted Operating System" in the SeaView design was the Class A1 Gemini Multiprocessing Secure Operating System (GEMSOS) product [21]. The abstract from one of the SeaView project's published papers summarized the approach:

We describe our approach to multilevel database security and show that we can support element-level labeling in a Class A1 database system without the need to verify the entire database system, or even most of it. We achieve both the high degree of assurance required for Class A1 and the flexibility of element-level labeling by layering the TCB, where the lowest TCB layer is a Reference Monitor enforcing mandatory security; and by decomposing multilevel relations into single-level relations that are managed by the Reference Monitor.

—Teresa F. Lunt, *et al.* [22]

Oracle successfully implemented a commercial-grade system based on the SeaView architecture [23] and identified other expected benefits, including portability to mandatory TCBs other than GEMSOS, time to market and ease of evaluation [20]. These benefits were due to the TCB subsets approach, because a large portion of the system had already been subjected to evaluation, even if a different TCB were substituted. The SeaView design approach used by Oracle also benefitted from the balanced assurance security pattern available with TCB subsets, where the MAC enforcement of the trusted operating system typically has higher assurance than the DAC enforcement dependent on the large database machinery.

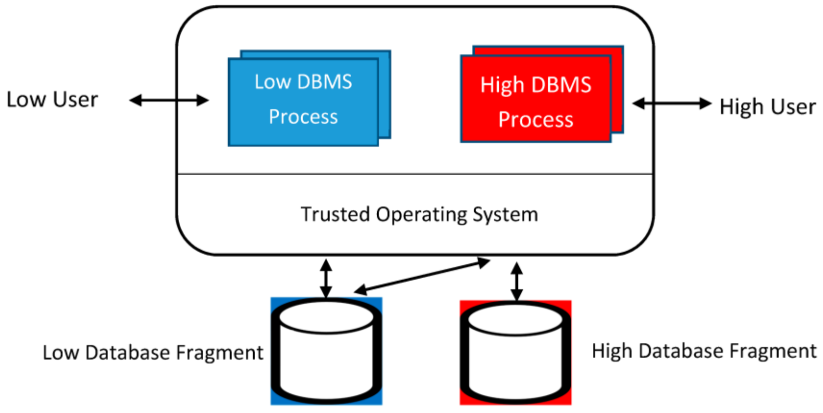

Figure 3. SeaView subset architecture.

Again, this was not just a theoretical conclusion. Oracle went on to have the National Security Agency (NSA) complete a formal security evaluation. In the NSA published Evaluated Product List (EPL), the specific commercial product was identified as "Oracle Corporation Trusted Oracle 7, 7.0.13.1 with the Procedural Option, Database Management System" [23]. The Oracle commercial product referred to their implementation of the SeaView model as the "OS MAC" mode. The NSA EPL confirmed this, noting "In OS MAC mode, only the underlying operating system enforces MAC security policies on single-level storage objects at the file level. Each database is single level; however, users can read information from lower level databases."

3.2. Honeywell Multics Commercial Mainframe

Multics (Multiplexed Information and Computing Service) was a mainframe timesharing operating system begun in 1965 and used until 2000. Honeywell, a major computer vendor, offered Multics as a commercial product and sold dozens of systems. Honeywell had the system formally evaluated by NSA. As the result of its successful evaluation, NSA awarded Multics a TCSEC Class B2 as reflected in the Formal Evaluation Report (FER) [24].

Multics was designed to be secure from the beginning, but the initial product did not support a MAC security policy. In the 1970s, Honeywell included an enhancement, called the Access Isolation Mechanism (AIM), to support MAC. This facility was a part of every Multics system shipped. It enforced label-based classification of information and authorization of users. Each object in the system had a classification level and categories, and each user of the system had an authorization level and categories. The AIM augmented the Multics Access Control List (ACL)-based DAC. The mechanism for both the DAC and MAC policy were shown to be consistent with the Bell–LaPadula formal security policy model [12].

A key issue in the design of trusted systems is the ability of a system to support practical applications. We look here at two real-world uses of Multics, configured to run with very different MAC policies.

The first example is a system to enforce the classical government multilevel MAC policy that has been summarized [25] as follows: "The Air Force Data Service Center (AFDSC) in the Pentagon had articulated a requirement for a multilevel time-sharing system that would allow programmers with Secret clearances to develop models that would support defense analysts who would apply the models to Top Secret data."

This installation of Multics was first installed in November 1973, and the final shutdown was June 1992. The system lifetime of nearly twenty years provided ample opportunity to observe the experience of many thousands of users and the response of management to this experience. They found little, if any, evidence that the capabilities were significantly different than expected from any time-sharing system of that era. In fact, the Multics-based AFDSC grew to become the largest data center in the Pentagon. It is reported [26] that "This site had 6 systems: four were multilevel secure, two were unclassified." Many of these were powerful multiprocessor systems.

The second example is a system that relied on the Multics AIM to provide controlled sharing of open discussions while simultaneously supporting and protecting the highly sensitive proprietary design information of competing vendors. Although the NCSC was a government organization, the Multics MAC policy primarily represented distinct security domains for commercial computer vendors and was unrelated to the Government's own multilevel policy. Multics was used as a timesharing system to support its team of evaluators and to provide a secure shared computing and communications environment for evaluators, vendors and consultants. This Multics was installed in July 1984, and its final shutdown was March 1998. The system was routinely used to provide services accessed over the Internet by users. We are not aware of any widespread adverse reaction about the limitations of its services based on its rigorous imposition of the Reference Monitor principles. In fact, a published description of its services sounds like other typical Internet services:

> DOCKMASTER is a (Multics-based) subscription service of the National Computer Security Center (NCSC), that they consider an "Information Security Showcase." Its large repertoire of available services (its user's manual is over one hundred pages) includes E-mail, electronic bulletin boards, and allows hands-on software evaluation. Its Evaluated Products List rates computers and computer security products. Users can access online documents

> (such as the Orange Book), participate in online discussions, and learn about computer security conferences. Users can connect to DOCKMASTER through MILNET (part of the Internet), TYMNET (a packet switching service), and local dial-in.
>
> —Multicians.org [27]

We have emphasized the importance of high assurance to address the serious danger from subversion, yet with a TCSEC Class B2 level of security, the commercial Multics product was not high assurance. As noted earlier, it is only a Class A1 system that can be expected to substantially deal with the problems of subversion of security mechanism. Being "just Class B2" does not mean there was little security value; Multics offered considerably stronger security than most systems commercially available at that time. However, there is still the question as to whether (or not) the Reference Monitor patterns could be applied to build a Class A1 Multics that preserved the powerful capabilities illustrated above. A joint project by Honeywell, MIT, the MITRE Corporation and the Air Force was started to build a security version of Multics intended to satisfy the requirements for Class A1 [28]. The project completed and formally specified a detailed "Design and Abstract Specification of a Multics Security Kernel" [29]. A conclusion of the project was that "the specification contains sufficient detail to allow decisions to be made about the ability of the kernel to support efficiently and compatibly the current Multics user interface".

The Reference Monitor-based security patterns proved highly effective as a method for both constructing and evaluating the security of the complex, general-purpose Multics system, which was offered as a commercial mainframe by a major international computer vendor. Furthermore, systematic security engineering could produce and methodically evaluate a highly compatible Class A1 version. Some of the applicable security patterns (in the informal "structured and repeatable methods" sense) included: (1) being deliberately and explicitly designed to be a valid interpretation a formal security policy model; (2) hardware protection rings enabling TCB subsets; (3) hardware segmentation for controlled sharing of reference monitor objects; and (4) configurable MAC policy support.

3.3. Government BLACKER Secure Network Infrastructure

BLACKER was an NSA-produced and deployed encryption system that served as essentially a MLS virtual private network for government TCP/IP networks for several years. The BLACKER system was designed to provide host-to-host secure communications across a packet-switched network based on Internet technology, called the Defense Data Network (DDN).

According to Clark Weissman [30], "BLACKER is the name given by the DoD (United States Department of Defense) to a long term project to build an integrated suite of devices to secure the Defense Data Network (DDN). There are four devices, which together provide a secure system appliqué to DDN designed to achieve Al certification per the [TCSEC]." This network system is illustrated in Figure 4. Notice that the security of the entire system had to be evaluated, not just individual pieces. It was a large, complex, distributed network system whose architecture and design were informed by the technology represented by the patterns of the TNI, especially those associated with a partitioned TCB. Notably, the TNI explicitly (p. 190) mentions BLACKER:

> Assurance is concerned with guaranteeing or providing confidence that features to address Data Confidentiality threats have been implemented correctly and that the control objectives of each feature have been actually and accurately achieved. Blacker is an example of such an application of a TCB for high assurance of data confidentiality.
>
> —National Computer Security Center [5] (p.190)

Each of the four BLACKER devices are a TNI component in the partitioned TCB. The BLACKER Front End (BFE) is a smart encryption box interposed between the network and a host computer. The packet switch hosts using attached BFE devices can communicate with each other over an unsecured packet-switched network using data paths secured by the encryption services of the

BFEs. The BFE device has its own security kernel, as is required for TNI composition. A BLACKER Initialization-parameters Carrier (BIC) is a removable device that gives a BFE its host-specific personality. The Access Control Center (ACC), with its own security kernel, is the "brains" of the system that determines and controls permissions for hosts to exchange messages. The BLACKER Key Distribution Center (KDC), with its own security kernel, distributes encryption keys to the BFEs under direction of the ACC.

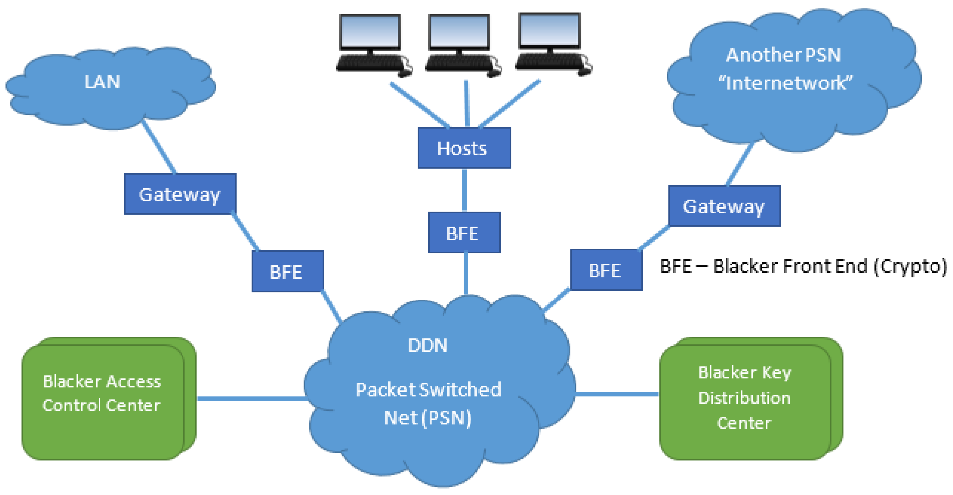
LAN
Hosts
Another PSN "Internetwork"
Gateway
BFE
BFE
Gateway
BFE
BFE – Blacker Front End (Crypto)
DDN
Packet Switched Net (PSN)
Blacker Access Control Center
Blacker Key Distribution Center

Figure 4. BLACKER-Defense Data Network (DDN) architecture.

The BLACKER system innovatively used centralized key management and access control to implement the first secure system with Trusted End-to-End Encryption using government encryption. BLACKER security is a unique challenge, because it must concurrently satisfy both Communications Security (COMSEC), primarily related to cryptography, and Computer Security (COMPUSEC) requirements. COMSEC requirements are satisfied by a single logical processor managing the cryptographic component. This was an unprecedented and revolutionary approach, because the security of one of NSA's most secure (called Type 1) cryptographic systems was deeply dependent on the assurance of the security kernels. The BLACKER Program selected "GEMSOS, an off-the-shelf kernel from Gemini Computers, for the ACC/KDC" [30] (p. 290). As required by the TNI-partitioned TCB security pattern, for the BLACKER overall system to be evaluated as Class A1, the security kernels for these individual networks components must themselves meet Class A1. The NSA Final Evaluation Report of their Class A1 evaluation of GEMSOS highlighted that pattern when reporting the significance of that selection:

> One of the first applications of the GEMSOS security kernel was for an MLS wide-area network called BLACKER, which was successfully evaluated against the Class Al under the Trusted Computer System Evaluation Criteria (TCSEC). In the BLACKER system, the GEMSOS security kernel along with other trusted components, is used as the basis for trust for critical cryptographic and key distribution functions that maintain communications separation by cryptographic means.
>
> —National Computer Security Center [31] (p. 5)

The production program was awarded to Unisys (merged SDC and Burroughs) in the early 1980s, with devices fielded in the late 1980s. Cisco added support for encryption through the BFE, and the Defense Communications Agency (DCA) certified Cisco Systems' DDN X.25 standard service implementation for attachment to the DDN. BLACKER was formally evaluated as TCSEC Class A1 and operationally deployed for several years internationally to protect the most sensitive information of the U.S. and its allies.

The Class A1 BLACKER system, a complex and large (up to 50,000 hosts) distributed system based on Internet technology, is a rather dramatic demonstration of the power of the TNI security patterns. In particular, the U.S. Government formally evaluated this large complex system for one of the Government's most critical applications as meeting the highest level of security (*i.e.*, Class A1) while incorporating an off-the-shelf product (*i.e.*, GEMSOS security kernel) that was not built for this particular system, but offered by a commercial high assurance computer vendor.

3.4. Government Pentagon Enterprise Management Information System

As discussed in Section 3.2, an installation of several Honeywell Multics commercial mainframe computers was the largest MLS data center in the Pentagon. When Honeywell went out of the computer business and Multics was no longer a supported commercial product, the Air Force needed to find an MLS replacement. The program to quickly and affordably purchase a new MLS system was called the Headquarters Systems Replacement Program (HSRP). The challenge was that, by that time, no major commercial vendor offered a MLS computer that was designed to meet even the medium level of assurance (TCSEC Class B2) of Multics.

It was recognized by the Air Force customer and vendors alike that it could be expected to take on the order of a decade to build from scratch even a modestly high assurance system to enforce a MAC policy, as would be required to meet the MLS need. Furthermore, developing such a new system could be expected to take many tens or even hundreds of millions of dollars. This was clearly not an even remotely possible approach to meet the requirements.

Fortunately, the TNI with its composition approach based on the partitioned TCB was already available. This was the only viable approach identified. Grumman Data Systems (GDS), a large-scale system integrator, was selected as the prime contractor for HSRP and received a firm-fixed price contract award in July 1988. Daniel Gambel was the lead security architect for their solution. He described the project in a 1989 conference paper [32] entitled "HSRP—A1'ing a large-scale management information system". For much of our summary in this example, we draw heavily from what is in that paper. Gambel introduced the project strategy in the paper's abstract:

> This paper discusses the development of a large-scale Management Information System (MIS) at the A1 certification level, as discussed in the Trusted Network Interpretation (TNI) of the Trusted Computer System Evaluation Criteria known as the TNI. The system design uses three commercial off-the-shelf (COTS) products as a basis for a distributed Trusted Computing Base (TCB) using a component architecture as discussed in the TNI. Several extensions to these COTS products that are necessary to enable the system to meet specific requirements are discussed.
>
> —Daniel Gambel, *et al*. [32]

GDS had to meet demanding requirement for both capabilities and security. Bound by a firm-fixed price contract, they needed to pursue a relatively low-risk approach for which they could be confident of success. The whole system had to work together as an integrated MIS, much like what today might be termed an enterprise cloud. The demanding needs included [32]: "An A1 implementation as defined in NCSCTG-005, Version 1, of the TNI 1. Ultimately, the system must support processing of data from Unclassified to Top Secret with 1000 classified terminals supporting over 3000 classified system users in a system population of over 10,000 users."

Using the "MAID" policy taxonomy of the TNI discussed earlier, different categories of the overall security policy were allocated to different components. The balanced assurance security pattern was applied so that not all of the components had to have Class A1 assurance, even though the overall system had to be designed to meet Class A1. The discretionary policy ("D" in the taxonomy) was allocated to three separate Class C2 IBM 3090 large mainframe hosts, one for each processing level of Unclassified, Secret and Top Secret. Each of these single-level hosts is connected via a TNI "secure channel" to a multilevel Secure Communications Processor Environment (SCPE) developed by GDS.

Gambel emphasizes that this is practical because, "The SCPE is based on the Gemini Multiprocessing Secure Operating System (GEMSOS) security kernel." A Class A1 evaluation of GEMSOS means that it "serves as the COTS portion of the A1 TCB for the HSRP system". In fact:

> The SCPE consists of two separate types of components configured on two different Gemini models. The first is the Access Control Module (ACM), which provides most of the I/A functions for the system and the I/A associated mandatory supporting policies. The ACM validates the user's logonid and password, determines the user's acceptable Mandatory Access Control (MAC) session domain/range from the user's clearance and the terminal classification range . . .
>
> The second type of SCPE component is the Communications Control Module (CCM). The CCM's major role is to enforce the MAC policy. The CCMs are also responsible for communication and communication management services between the HSRP user community and the hosts.
>
> —Daniel Gambel, *et al*. [32]

Gambel points out that "The initial configuration of the SCPE for the HSRP system consists of two ACMs, one primary and one secondary, and 16 CCMs." He also notes that

> The use of a COTS TCB has several positive aspects. The foremost of these is simply that in building the system, we do not have to start from scratch but can build upon the foundation provided by the A1 TCB. The advantages of this are far reaching in cost and time . . .
>
> Another advantage of using a COTS TCB concerns the Assurance Evidence. Assurance Evidence consists of the documentation required . . . for A1 certification. This includes the Formal Top Level Specification (FTLS), the Descriptive Top Level Specification (DTLS), the Trusted Distribution Plan (TDP), the Covert Channel Analysis (CCA), the Policy Model (PM), and the Code Correspondence (CC) as well as user's guide and testing documentation. In the A1 development process, the development of this documentation is more time consuming than the development of the software. By using a COTS TCB, we can take advantage of the vendor-developed Assurance Evidence.
>
> —Daniel Gambel, *et al*. [32]

As with the BLACKER project, Reference Monitor-based security patterns proved highly effective as a method for both constructing and evaluating the security of a complex, distributed HSRP system. It would have been impossible to deliver this high assurance MLS system without the TNI pattern approach. In particular, this example shows the power of incorporating an off-the-shelf OEM product (*i.e.*, the GEMSOS security kernel) offered by a commercial high assurance computer vendor rather than trying to build new high assurance components from scratch.

3.5. Research Prototypes Hosted on the GEMSOS™ Security Kernel

In addition to the deployed commercial products and government systems described above, there are several other research prototypes with published results that serve to demonstrate the use of Reference Monitor-based security patterns. In contrast to all of the previous examples, these prototypes were not operationally deployed to protect sensitive live data, and an evaluation was not completed for the contemplated system represented by the prototype. One or both of the authors of this paper actively participated in the design and development of each of the following examples. These prototypes were all hosted on the GEMSOS security kernel [21].

NSA previously evaluated [33] the GEMSOS™ security kernel in the Class A1 Gemini Trusted Network Processor (GTNP), a commercial product of Gemini Computers, Inc. The GEMSOS security kernel implements a Reference Monitor that verifiably enforces MAC policies. This kernel leverages the support for protection ring and segmentation security patterns provided by the Intel IA-32 (32-bit version of the x86 instruction set) processor architecture.

Each of the following examples was created to demonstrate the feasibility and characteristics of its architecture for a large, evaluable system, but because each was only a prototype, the contemplated large system was not actually evaluated. However, a primary result of each project was a method for such an evaluation. The architecture for each of the prototypes was constructed based on composition patterns for TCB subsets and TCB partitions. Each of these reflects a particularization of the general composition architecture illustrated in Figure 5. Each of these systems was designed for incremental evaluation, so the resulting system could be evaluated without needing to reevaluate the security kernel. This reflects that GEMSOS is a commercial product available under the typical OEM business model.

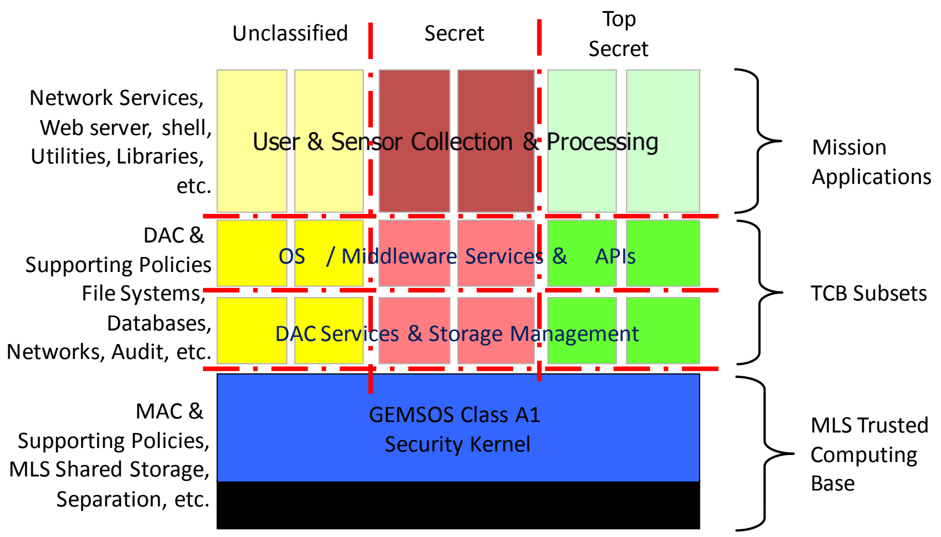
Unclassified
Secret
Top Secret
Network Services, Web server, shell, Utilities, Libraries, etc.
User & Sensor Collection & Processing
Mission Applications
DAC & Supporting Policies
OS / Middleware Services & APIs
File Systems, Databases, Networks, Audit, etc.
DAC Services & Storage Management
TCB Subsets
MAC & Supporting Policies, MLS Shared Storage, Separation, etc.
GEMSOS Class A1 Security Kernel
MLS Trusted Computing Base

Figure 5. Gemini Multiprocessing Secure Operating System (GEMSOS) generalized composition architecture.

3.5.1. Source-Code Compatible Secure Linux

This prototype demonstrates a highly secure Linux execution environment using a high assurance security kernel as the Linux hardware abstraction layer and supports source code compatibility for applications that run on Linux. This is a true MLS Linux, not just a vaguely "security enhanced" Linux. The prototype quite closely follows the GEMSOS generalized composition architecture illustrated in Figure 5, directly applying the TCB subset security pattern.

The Linux operating system services are based on the Aesec Gemini Application Resource and Network Support (GARNETS) file system [34] that was designed to run on the GEMSOS security kernel. GARNETS runs as an untrusted application subject in a layer above the TCB, creating a file management capability using TCB objects that are themselves built using kernel-managed segments. The GARNETS file system operates like standard file storage; but the data are securely labeled with mandatory access classes, and the TCB reliably isolates security domains, permitting only the sharing allowed by the MLS policy.

The GARNETS file system creates a logical name space that spans domains, leveraging the MLS property that higher domains can read lower domain data (*i.e.*, the directory and file structure of lower domains), but not modify it in any way. Thus, secure "low-to-high" sharing does not require trusted code outside of the TCB or copying of unused file data to higher domains, because higher domains have direct access to lower-domain data. Linux applications call the network stack and file system through a system call interface that is intended to support Linux application source code compatibility.

3.5.2. GemSeal™ Guard Secure Network Interface Proof of Concept

In a multilevel environment, users may, for example, need to access unclassified information not available on their system high networks. If their environment does not include Internet or other unclassified infrastructure, the connection demands a controlled interface. Highly sensitive networks must take extraordinary care to protect their data from attack and unauthorized data disclosure when delivering unclassified (or other lower sensitivity) connections to users. Protection against patient, professional adversaries requires the systematic and verifiable protection of information flow controls provided by systems implementing a Reference Monitor that satisfy the Class A1 certification criteria.

Any connection between networks with different sensitivity levels is multilevel. The nature of network protocols and their ability to find routes between connected devices makes such connections both necessary and multilevel by their nature. These connections must have an assurance level appropriate for the value of the information that they protect. The security patterns for a partitioned TCB with components connected by secure channel were used to create a solution architecture for what was called GemSeal. GemSeal Guards rely on the GEMSOS security kernel for MLS enforcement and use its label integrity and distributed key management mechanisms. This architecture is illustrated in Figure 6.

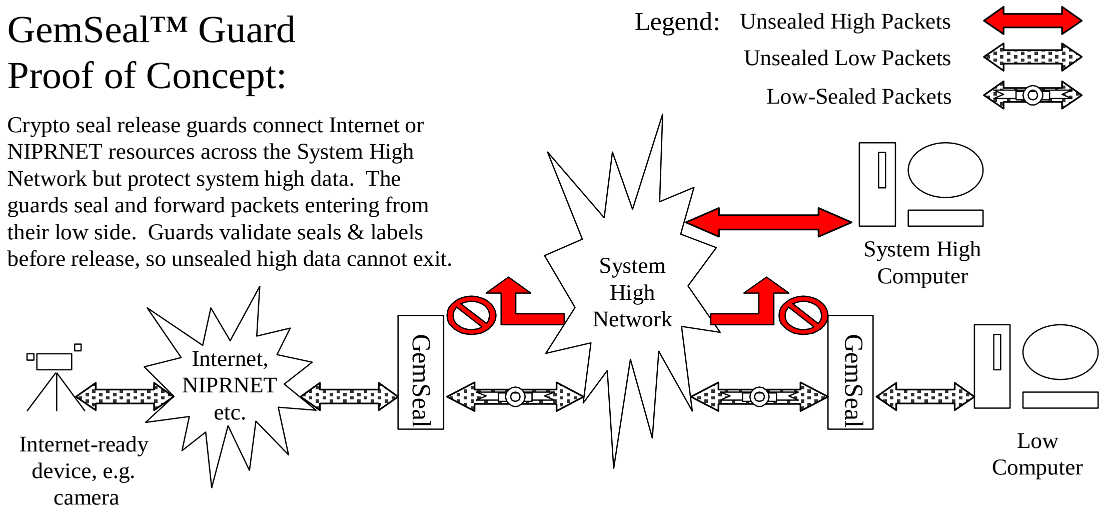
GemSeal™ Guard
Proof of Concept:
Crypto seal release guards connect Internet or NIPRNET resources across the System High Network but protect system high data. The guards seal and forward packets entering from their low side. Guards validate seals & labels before release, so unsealed high data cannot exit.
Legend:
Unsealed High Packets
Unsealed Low Packets
Low-Sealed Packets
System High Network
System High Computer
Internet, NIPRNET etc.
GemSeal
GemSeal
Internet-ready device, e.g. camera
Low Computer

Figure 6. GemSeal guard architecture.

GemSeal Guards use a "crypto seal" to cryptographically bind packets entering the system with the label for their low sensitivity source. The guards forward each labeled packet across the system high network to a guard at its low sensitivity destination. Destination guards validate the label of each packet against the destination sensitivity label before releasing it. A system high packet cannot exit the system because it will not have a crypto seal binding a label to a matching low sensitivity destination label. A paper by the authors [35] discusses this architecture in the context of a more general guard architecture that was created using Reference Monitor-based patterns.

3.5.3. Secure File Service for Cloud Prototype

This prototype demonstrates a standards-compliant interface to a highly secure Network File Service (NFS). In this prototype, the NFS application is a typical Linux application that runs on top of the high assurance secure Linux summarized above. It serves as a compelling demonstration of its Linux application source code compatibility. The demonstrated file services supporting a MAC policy are created by integrating an open source Linux NFS implementation with this GEMSOS-based operating system. The architecture for the combined NFS and MAC-enforcing Linux is illustrated in Figure 7.

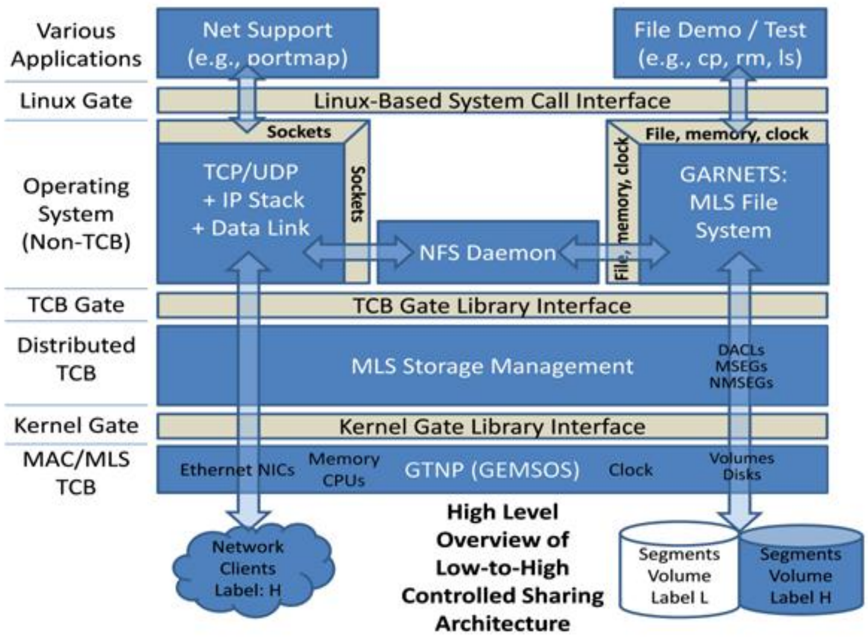

Figure 7. MAC policy-enforcing NFS architecture. GTNP, Gemini Trusted Network Processor; NIC, Network Interface Card.

Without the benefit of a security kernel, the current approach to maintaining the security of multi-domain environments is to duplicate the hardware for each domain, which is expensive and which complicates sharing in many applications that depend on it. In contrast, the high assurance file server in this prototype enforces a configurable MAC policy in its support of a controlled access and controlled cross-domain file sharing solution, which can be used for cloud storage [36]. The MLS file server is designed to support the Network File Service (NFS) file protocol where workstations, servers and applications process information of a single classification level while allowing data sharing between the various levels. The prototype server has separate Network Interface Cards (NICs) for each security domain. No trusted code outside the GEMSOS TCB is required in order to provide low-to-high controlled sharing via MLS read-down from higher domains to lower domain files and directories.

4. Discussion

We have examined the application of what we call Reference Monitor security patterns, though most have not previously been recognized as security patterns, from the perspective of previously reported studies carried out across several decades. The success of these patterns at creating evaluable secure systems has been repeatedly demonstrated in a diverse range of large, distributed, complex systems. We have focused on systems-oriented methods of applying these patterns to evaluate system security. Our examples include interpreting our identified patterns in the deployed system not only for commercial products, but also unique government projects where security was a major system objective. We have also included results from some research prototypes in which the authors were directly involved, in order to draw from personal insights that might not otherwise be noticed.

4.1. Recap of Key Elements of Reference Monitor-Based Security Patterns

Rather than speculate about hypothetical approaches for evaluating the security of complex systems, our results are based exclusively on working examples that included a major requirement or emphasis on objective, third-party, systematic evaluation of security. Our results are based on the integrated set of security patterns that were created decades ago based on the Reference Monitor abstraction and their codification in the TCSEC and TNI. These are essentially a tool bag; no specific system is likely to need all of the tools, so we provide here a brief description of the major Reference Monitor patterns identified above. For the sake of precision, some of the descriptions use computer science "terms of art".

4.1.1. Trusted Computing Base

The TCB is the small, simple and well-structured implementation of the Reference Monitor abstraction, which has four components. The Reference Monitor enforces the system security policy, mediating every attempt by a *subject* to gain access to an *object*, based on a tamperproof *authorization database*. Its *audit trail* maintains a record of security-related events, without unauthorized observation or interference. When this security pattern is applied, these four components are distinguishable elements of the system.

4.1.2. Subjects

Subjects are the Reference Monitor active entities, such as processes or tasks that gain access to information on a user's behalf, causing information to flow between objects or changing the system state. There are two closely-related sub-patterns:

Hierarchically-Ordered Protection Domains: Protection Rings

Within a processes or task, there may be a totally ordered set of protection domains. In a processor instruction set, the degenerate case of two domains is commonly referred to as the user domain and the kernel domain, the most privileged domain. The more general case is referred to as protection rings. Technically, a subject is a process/domain pair, *i.e.*, there are multiple subjects in a single process. This is a particularly important pattern because it enables the composition of security components using TCB subsets.

Security Kernel

The most privileged domain, also known as "Ring 0", is the hardware and software that create the execution environment for subjects. That means the security kernel itself is not a subject, although it can support the most privileged TCB subset.

4.1.3. Objects: Segments

Objects are the passive data repositories for information, which subjects can access via the reference monitor subject to the policy authorizations in the Reference Monitor authorization database. Objects include both metadata, such as directories, and organizational information, such as files or segments. It is essential that there be separate and distinct repositories for objects with different attributes (e.g., classifications) accessible by untrusted subjects with different authorizations (e.g., for MAC with different clearances). For systems with shared objects, there is strong evidence that the most practical objects for effective security evaluation are individual hardware-supported segments.

4.1.4. Policy Taxonomy: "MAID" Map to Reference Monitor

A system is only "secure" with respect to a precisely-defined policy. This pattern enables evaluation to be methodically conducted. The TNI taxonomy of Mandatory, Audit, Identification and Discretionary (MAID) is helpful because it is designed to directly leverage the Reference Monitor. "M" and "D" access control is from the authorization database; "I" is for interpretation of subjects for individuals; "A" is for interpretation of audit file. With four policy elements, there are sixteen possible policy combinations.

4.1.5. Composition of TCB Subsets

The TCB subset pattern enables the composition of complex systems as multiple subjects in hierarchically-ordered privilege domains in a single process supported by an underlying security kernel. This is effectively supported in a segmented address space by hardware protection rings, which are a complete, conceptually simple protection mechanism with precisely-defined semantics.

4.1.6. Composition of TCB Partitions

The partitioned TCB pattern defined by the TNI provides a precise and systematic pattern for creating the architecture of large, distributed, complex, and secure systems as a composition of distinct, interconnected TCB partitions on distinct network components.

Policy Allocation to Network Components: 16 Combinations

The TNI provides a precise pattern for how each of the sixteen MAID policy combinations can be allocated to a component and composed with others to support the total policy for a complete, complex distributed system.

Incremental Evaluation

This pattern enables a powerful method to incrementally evaluate the security of the entire system by individually evaluating the TCB partition of each component. Each component has its own reference monitor that is evaluated with respect to only the enforcement of the policy allocated to that component.

Secure Interconnecting Channels

This pattern provided by the TNI provides for a constrained composition of components in which authoritative security information is only passed between components via a "secure channel". For a MAC policy, an authoritative label is the only security information that is exchanged. The TNI calls this a "multilevel channel".

4.1.7. Balanced Assurance

The balanced assurance pattern balances the value of the policy with the cost of implementing and evaluating it. Using TCB subsets, there can be a hierarchy of policies that have different mechanism strengths. MAC is typically the most valuable policy, and most costly, and is typically enforced by the security kernel. This provides information flow confinement, even for Trojan horse subversion. The DAC policy may be enforced with lower assurance and less cost in a less privileged protection domain on top of the security kernel. The application policy may be allocated to an even lower assurance TCB subset, e.g., for the rich access policy enforced by most DBMS.

4.1.8. Security Software Engineering

There are a few software engineering techniques that are essential to an evaluable implementation of a reference monitor, whether in a monolithic system or a component in a partitioned TCB. There are well-developed patterns for three of these: (1) layering; (2) information hiding; and (3) a strongly-typed language, such as Pascal, for the source code of the implementation.

4.2. Delivery Architecture

It can be expected to take 10–15 years and tens of millions of dollars to build and evaluate a high assurance security kernel. Once completed, the systematic engineering process of the Reference Monitor-based security patterns greatly facilitates long-term maintenance of security assurance through technology refresh. In the TCSEC context, this is called the "ratings maintenance phase" (RAMP).

The RAMP has repeatedly delivered affordable updates to systems in a few months and delivered a new secure system based on the same reusable security kernel in a couple of years.

The major investment for a general purpose security kernel is highly reusable, as reflected in the example systems in the Results section above. This is demonstrated by OEM deployments of highly secure systems and products, ranging from enterprise "cloud technology" to general purpose data base management systems (DBMS) to secure authenticated Internet communications, by applying commercially available security kernel technology.

5. Conclusions

Based on the evidence of many successful evaluated or evaluable systems, a reasonable conclusion is that the most effective approach to evaluating the security of complex systems is to deliberately construct them using security patterns specifically designed to make the systems evaluable. A well-developed set of security patterns based on the Reference Monitor concept has been repeatedly and successfully used over many years for that very purpose, yet without recognition that they are, in fact, security patterns. The application of these patterns has been shown to be one of the most powerful and proven methods for building and evaluating large, distributed, complex and secure systems.

Acceptance of the Reference Monitor and associated TCSEC patterns for building and evaluating secure systems, however, is not universal. The rigorous development and evaluation methodology necessary to engineer systems that meet the requirements of TCSEC Class A1 is expensive and time-consuming. Moreover, the functionality of a system may be simplified or reduced in order to facilitate evaluation. As one author noted:

> While customers may want improved security, they usually have second thoughts when security features adversely affect other, "more important" features. Since few customers are willing to pay extra for security, vendors have had little incentive to invest in extensive security enhancements . . . demand is fairly weak and can easily evaporate if the features should have an adverse impact on cost or any other functions.
>
> —Morrie Gasser [19]

Yet, we have no proven method of implementing high-assurance secure systems other than what we have here referred to as Reference Monitor patterns. In critical applications where security is of the utmost concern there is currently no viable alternative, hence the need to express the implicit security patterns of this methodology in a structured way to facilitate their use.

A concern is that, even if the security patterns described in this paper are written using the existing method of formalizing security patterns—a structured set of sections describing the problem solved by the pattern, its context, the forces that affect the solution, *etc.*—the result will still not easily facilitate the application of the patterns to the composition of systems. In other words, the problem may not only be a lack of suitable patterns, but the lack of a suitably powerful pattern language to formally express in a useful way the decomposition idea originally described by Frincke. A future research area is how to extend the existing method of formalizing security patterns in order to support composition.

On the other hand, we are encouraged by the extensive real-world experience in successfully using composition for the design, implementation and evaluation of large, complex, distributed, and secure systems, as reflected in the results described above. These results include both real systems and research prototypes that have heavily leveraged the systematic security engineering and evaluation methodology that was codified as an engineering standard in the TCSEC and its TNI. We find that the broad contexts for our results—including mainframes, a DBMS, World Wide Web communications, and cloud computing—provide compelling evidence that there are no fundamental barriers to the formulation of suitable patterns.

Acknowledgments: Roger Schell, one of the authors, was a member of the executive management of organizations, including Gemini Computers, Inc., and Aesec Corporation, participating in the design and delivery of several of the systems discussed in the results.

Author Contributions: Mark R. Heckman conceived of and designed the methods and approaches for formulating this paper in terms of security "patterns". Mark R. Heckman and Roger R. Schell conceived of and designed the methods and approaches for formulating the results from previous published experiments. Mark R. Heckman and Roger R. Schell wrote the paper. Both authors have read and approved the final manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

DBMS:	Database Management System
MAC:	Mandatory Access Control
MLS:	Multilevel Security
OEM:	Original Equipment Manufacturer
OS:	Operating System
TCB:	Trusted Computing Base
TCSEC:	Trusted Computer System Evaluation Criteria
TNI:	Trusted Network Interpretation (of the TCSEC)

References

1. Ito, Y.; Washizaki, H.; Yoshizawa, M.; Fukazawa, Y.; Okubo, T.; Kaiya, H.; Hazeyama, A.; Yoshioka, N.; Fernandez, E.B. Systematic Mapping of Security Patterns Research. In Proceedings of the 22nd Conference on Pattern Languages of Programs Conference 2015 (PLoP 2015), Pittsburgh, PA, USA, 24–26 October 2015.
2. Kienzle, D.M.; Elder, D.T.; Edwards-Hewitt, J. Security Patterns Template and Tutorial. Available online: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.131.2464&rep=rep1&type=pdf (accessed on 21 January 2016).
3. Anderson, J.P. *Computer Security Technology Planning Study*; USAF Electronics Systems Division: Bedford, MA, USA, 1972.
4. Schumacher, M.; Fernandez, E.; Hybertson, D.; Buschmann, F. *Security Patterns: Integrating Security and Systems Engineering*; John Wiley & Sons: Hoboken, NJ, USA, 2005; pp. 253–258.
5. NCSC-TG 005: Trusted Network Interpretation. National Computer Security Center: Fort Meade, FL, USA, 1987; Available online: http://ftp.mirrorservice.org/sites/ftp.wiretapped.net/pub/security/info/reference/ncsc-publications/rainbow-books/NCSC-TG-005.pdf (accessed on 27 January 2016).
6. Department of Defense Trusted Computer System Evaluation Criteria (DoD 5200.28-STD). United States National Computer Security Center: Fort Meade, FL, USA, 1985; Available online: http://ftp.mirrorservice.org/sites/ftp.wiretapped.net/pub/security/info/reference/ncsc-publications/rainbow-books/5200.28-STD.pdf (accessed on 27 January 2016).
7. USAF Electronics System Division. *Multilevel Security Issues and Answers: An Evaluation of the AFSC Program (MCI-75-8)*; Hanscom AFB: Bedford, MA, USA, 1975; Unpublished work.
8. Schell, R.R. A University Education Cyber Security Paradigm Shift. In Presented at the National Initiative for Cybersecurity Education (NICE), San Diego, CA, USA, 3–4 November 2015; Available online: https://www.fbcinc.com/e/nice/ncec/presentations/Schell.pdf (accessed on 29 January 2016).
9. Anderson, E.A.; Irvine, C.E.; Schell, R.R. Subversion as a Threat in Information Warfare. *J. Inf. Warf.* **2004**, *3*, 51–64.
10. Langner, R. Stuxnet: Dissecting a cyberwarfare weapon. *IEEE Secur. Priv.* **2011**, *9*, 49–51. [CrossRef]
11. Dijkstra, E.W. *Notes on Structured Programming*; Technical Report 70-WSK-03; Technische Hogeschool Enidhoven: Enidhoven, The Netherlands, 1970.
12. Bell, D.E.; LaPadula, L.J. *Computer Security Model: Unified Exposition and Multics Interpretation (ESD-TR-75-306)*; Hanscom AFB: Bedford, MA, USA, 1975.
13. Schell, R.R.; Brinkley, D.L. Evaluation Criteria for Trusted Systems. In *Information Security: An Integrated Collection of Essays*; Abrams, M.D., Jajodia, S., Podell, H.J., Eds.; IEEE Computer Society Press: Los Alamitos, CA, USA, 1995; pp. 137–159.

14. Common Criteria for Information Technology Security Evaluation, version 3.1 revision 4, 2012. Available online: https://www.commoncriteriaportal.org/files/ccfiles/CCPART1V3.1R4.pdf (accessed on 29 January 2016).
15. Karger, P.A.; Zurko, M.E.; Bonin, D.W.; Mason, A.H.; Kahn, C.E. A retrospective on the VAX VMM security kernel. *IEEE Trans. Softw. Eng.* **1991**, *17*, 1147–1165. [CrossRef]
16. Brinkley, D.L.; Schell, R.R. Concepts and Terminology for Computer Security. In *Information Security: An Integrated Collection of Essays*; IEEE Computer Society Press: Los Alamitos, CA, USA, 1995; pp. 40–97. Available online: http://www.acsa-admin.org/secshelf/book001/02.pdf (accessed on 21 January 2016).
17. Bell, D.E. Looking Back at the Bell-La Padula Model. In Proceedings of the 21st Annual Computer Security Applications Conference (ACSAC'05), Tucson, AZ, USA, 5–9 December 2005; pp. 337–351.
18. Shockley, W.R.; Schell, R.R. TCB subsets for incremental evaluation. In Proceedings of the Third Aerospace Computer Security Conference, Orlando, FL, USA, 7–11 December 1987; pp. 131–139.
19. Gasser, M. *Building a Secure Computer System*; Van Nostrand Reinhold Company: New York, NY, USA, 1988.
20. Vetter, L.; Smith, G.; Lunt, T.F. TCB subsets: The next step. In Proceedings of the Fifth Annual Computer Security Applications Conference, Tucson, AZ, USA, 4–8 December 1989; pp. 216–221.
21. Schell, R.; Tao, T.F.; Heckman, M. Designing the GEMSOS security kernel for security and performance. In Proceedings of the 8th National Computer Security Conference, Gaithersburg, MD, USA, 30 September–3 October 1985; Volume 30, pp. 108–119. Available online: http://www.mrheckman.com/yahoo_site_admin/assets/docs/DesigningTheGemsosSecurityKernel-OCR-120409-DRAFT.158131458.pdf (accessed on 29 January 2016).
22. Lunt, T.F.; Denning, D.E.; Schell, R.R.; Heckman, M.; Shockley, W.R. Element-level classification with A1 assurance. *Comput. Secur.* **1988**, *7*, 73–82. [CrossRef]
23. National Security Agency, Evaluated Products List, Trusted Oracle7, Class B1, United States National Computer Security Center, 5 April 1994. Available online: http://webapp1.dlib.indiana.edu/virtual_disk_library/index.cgi/1347159/FID1806/epl/entries/ csc-epl-94-004.html (accessed on 27 January 2016).
24. Final Evaluation Report, Honeywell Information Systems, Multics MR11.0, National Computer Security Center, CSC-EPL-85/003, 1 June 1986. Available online: http://www.multicians.org/multics-fer.pdf (accessed on 29 January 2016).
25. Lipner, S.B. Security assurance. *Commun. ACM* **2015**, *58*, 24–26. [CrossRef]
26. Site History: AFDSC. Available online: http://www.multicians.org/site-afdsc.html (accessed on 29 January 2016).
27. Site History: DOCKMASTER. Available online: http://www.multicians.org/site-dockmaster.html (accessed on 29 January 2016).
28. Karger, P.A.; Schell, R.R. Thirty years later: Lessons from the Multics security evaluation. In Proceedings of the 18th Annual Computer Security Applications Conference, Las Vegas, NV, USA, 9–13 December 2002.
29. Schiller, W.L. *Design and Abstract Specification of a Multics Security Kernel*; MITRE Corp.: Bedford, MA, USA, 1977; Volume 1.
30. Weissman, C. BLACKER: Security for the DDN examples of A1 security engineering trades. In Proceedings of the 1992 IEEE Computer Society Symposium on Research in Security and Privacy, Oakland, CA, USA, 4–6 May 1992; pp. 286–292.
31. Final Evaluation Report, Gemini Computers, Incorporated, Gemini Trusted Network Processor, Version 1.01, National Computer Security Center, NCSC-FER-94/008, 28 June 1995. Available online: http://webapp1.dlib.indiana.edu/virtual_disk_library/index.cgi/1347159/FID1806/library/fers/ncsc-fer-94-008.pdf (accessed on 29 January 2016).
32. Gambel, D.; Walter, S.; Fordham, M. HSRP—A1'ing a large-scale management information system. In Proceedings of the 7th Computers in Aerospace Conference, Monterey, CA, USA, 3–5 October 1989; American Institute of Aeronautics and Astronautics, Inc.: Reston, VA, USA, 1989; pp. 920–923.
33. National Security Agency, Evaluated Products List, GTNP Version 1.01, Class A1, United States National Computer Security Center, 6 September 1994. Available online: http://webapp1.dlib.indiana.edu/virtual_disk_library/index.cgi/1347159/FID1806/EPL/ENTRIES/CSC-EPL-94-008.HTML (accessed on 27 January 2016).

34. Irvine, C.E. A multilevel file system for high assurance. In Proceedings of the 1995 IEEE Symposium on Security and Privacy, Oakland, CA, USA, 8–10 May 1995; pp. 78–87.
35. Heckman, M.R.; Schell, R.R.; Reed, E.E. A high-assurance, virtual guard architecture. In Proceedings of the 2012 IEEE Military Communications Conference, MILCOM 2012, Orlando, FL, USA, 29 October–1 November 2012; pp. 1–9.
36. Heckman, M.R.; Schell, R.R.; Reed, E.E. A multi-level secure file sharing server and its application to a multi-level secure cloud. In Proceedings of the 2015 IEEE Military Communications Conference, MILCOM 2015, Tampa, FL, USA, 26–28 October 2015; pp. 1224–1229.