Multi-Level Security for Service-Oriented Architectures

HariGovind V. Ramasamy and Matthias Schunter
IBM Zürich Research Laboratory
Rüschlikon, Switzerland
{hvr,mts}@zurich.ibm.com

June 19, 2006

Abstract

Multi-level security (MLS) is a well-established and thoroughly studied approach towards security. Service-oriented architectures (SOA) are emerging in the commercial world and promise increased flexibility and better interoperability. While both concepts have substantial merit, there is no well-established approach for combining both. In this paper, we describe how to provide multi-level security in a service-oriented architecture. First, we propose a conceptual design for multi-level security in a service-oriented architecture. We then describe how this model can be realized in today's defense networks that are structured into mutually isolated network zones with different confidentiality classifications.

1 Introduction

There is an increasing interest to move network-centric defense applications to service oriented architectures (SOA) [8]. While this move has the potential to yield improved flexibility and inter-operability within forces and coalitions, a sound approach towards security in such architectures is still missing. In the context of military applications, security policies have been dominated by the concept of multi-level security (MLS) [5][7][6][16]. MLS policies aim to enable free information flow between authorized subjects and prevent disclosure to unauthorized subjects. Enforcement of these policies is based on classification and labeling of subjects (such as users, machines, network zones, etc.) and (data) objects. Classification labels (such as unclassified, restricted, confidential, secret, and top secret) identify the sensitivity level of the data and the clearance level of the subjects.

We address the problem of providing multi-level security in a service-oriented architecture. Today's military security architectures are structured into physically isolated "stovepipes" of different classifications and suffer from substantial inflexibility. Multi-level services that can act on multiple classification levels are hard to build with sufficient assurance and are therefore uncommon. Furthermore, there are no well-defined automated mechanisms for altering the assigned classification level of the data. For example, down-classification (or de-classification) of data is mostly done manually through trusted operators and is error-prone.

Our design decomposes MLS flow-control enforcement into a small number of individual high-assurance services. Our design enforces a clean separation between integrity classification and confidentiality classification of both subjects and objects. This increases the flexibility of handling data. For example, whereas high-confidentiality data can be traditionally handled only on high-integrity systems, in our design, any data after appropriate transformations (such as encryption and authentication) can be handled on a system of any integrity classification.

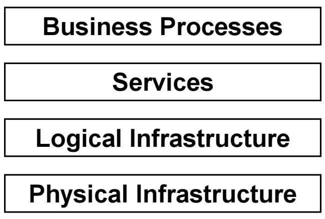

Figure 1: Layered System Model

The rest of the paper is structured as follows. In Section 2, we present the abstraction layers of the high-level system model in which our design was carried out. In Section 3, we define our conceptual model for multi-level security in a service-oriented architecture and discuss the MLS-enablement of the main abstraction layer we focus — the services layer. Section 4 describes details of the architectural entities that are needed to implement the conceptual model. Section 5 describes how the conceptual model is realized by the describing the interaction between the services proposed in Section 4. The resulting design enforces a clean separation between a small number of high-assurance services and the other components of the service "brokering" infrastructure. As a result, the security-criticality[1] of the bulk of the service brokering infrastructure can be significantly reduced. Section 6 presents a discussion on how some special cases and challenges can be handled. Finally, Section 7 summarizes our key contribution and concludes the paper.

2 High-Level System Model

Figure 1 depicts the simplified overall system model in which we operate. Higher layers provide a higher level of abstraction: The *physical infrastructure layer* represents the physical infrastructure, i.e., all physical physical machines (servers, satellites, cell phones, firewalls, VPN appliances, etc.), physical zones, and physical networks. The *logical infrastructure layer* comprises logical systems and logical networks and includes technologies like virtual machines and VLANs. The *services layer* provides services using business applications. These web-services based on the SOAP [1] form the core of the service oriented architecture. Services handle data and associated metadata. On top of the data layer is the *business processes layer* which provides configurable business processes that orchestrate services by flexibly composing individual services and decision functions into business flows.

Previous work has explored how to apply MLS up to the logical infrastructure layer [10]. In this paper, we take the next step of MLS-enabling the services layer. This requires introduction of metadata that assigns security classifications to all architectural entities (such as services, platforms, machines, etc.) across all layers. The classification is done in two dimensions: confidentiality (or sensitivity) and integrity (or trust in the pedigree).

Entities at the higher abstraction layers rely upon entities at the lower layers to correctly provide their functionalities. A service depends on the logical platforms it is deployed in, the platforms in turn rely on the physical infrastructure hosting them, and so on. The classification of an entity at a higher layer can only be as high as the classification of the lower layer entities that it relies on.

We distinguish between single-level and multi-level entities. Legacy entities that are not classification aware are called *single level*. Single-level entities are entitled to handle and generate data with the same confidentiality and same or lower integrity classification as the entities themselves. Data that is modified by a single-level entity inherits the entity's classification. Data with a higher or lower confidentiality than

[1]Mouratidis and Giorgini define security criticality [12] as the "measure of how the security of the system will be affected if the security constraint is not achieved."

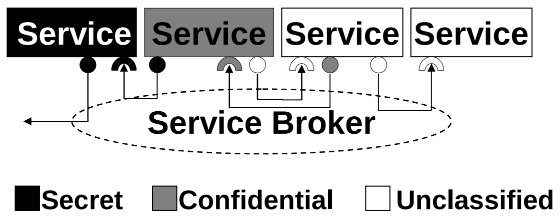

Figure 2: Conceptual Design of Multi-Level Service Interaction

the entity must undergo appropriate transformation before being handled by the entity. For example, data with a higher-confidentiality classification than the entity can only be stored at the entity only after down-classification or appropriate encryption. Similarly, data with a lower-confidentiality classification than the entity can be handled by the entity after being properly authenticated to prevent undetectable modification of the data.

Like a single-level entity, a multi-level or MLS entity has an associated classification. However, unlike a single-level entity, a MLS entities can handle and generate data with confidentiality level that is equal to or lower than their own and can assign integrity levels (to data) up to their own. MLS entities are *classification-aware*, i.e., they can process classification-related metadata attached to data (as explained later in Section 3) and they attach classification-related metadata to all data they generate.

An important issue in multi-level entities is providing sufficient isolation between the various classification levels. We believe that the complexity of isolating various classification levels with reasonably high assurance gets harder as we move up in the stack of layers. Today's state-of-the-art virtualization technologies [4][13][14] (virtual machines monitors, VLANs, remote VPNs) can be used to provide a certain level of isolation among classification levels at the logical infrastructure level while sharing physical resources. Leveraging this fact and notwithstanding the presence of legacy (single-level) services, the focus of this paper is to enable multi-level security at the services layer using a very small number of high-assurance MLS services.

3 A Model for Multi-level Secure Services

Figure 2 shows our conceptual design of multi-level secure service interaction. The different shades (white, grey, and black) are used to indicate different confidentiality levels. Shaded circles and semi-rings are used to depict output and input ports respectively. A service broker helps in the discovery of services and acts as a mediatory between service requesters and service providers, which may belong to the same or different classification levels. The service broker must implement mechanisms for enforcing flow control policies that allow for appropriate protection of sensitive information. For example, in Figure 2, the flow from unclassified service to the confidential service and the flow from the confidential service to the secret service must be allowed only after the data from the originator has been *up-classified*. Similarly, the flow from the confidential service to the unclassified service must be allowed only after the data from the originator has been *down-classified*. On the other hand, there may be free information flow between two services belonging to the same classification level (e.g., the two unclassified services in Figure 2).

There are two main steps involved in MLS-enabling the services layer. One is definition and management of MLS-related metadata and the other is the specification of security policies. In this section, we discuss these two steps. Later in Section 4, we describe the building blocks for the concrete realization of an MLS service broker, and in Section 5, we describe how the services interact with the components of the service broker to realize a MLS service-oriented architecture.

		Entity A’s Policies			
		must	may	should not	must not
Entity B’s Policies	must	must	must	must	conflict
	may	must	may	should not	must not
	should not	must	should not	should not	must not
	must not	conflict	must not	must not	must not

Table 1: Example of DAC Conflict Resolution Using Well-defined Priorities

3.1 Metadata Management

Data is the input and output of services. Metadata gives information about the data. We use metadata to indicate information about the associated data that is needed for enforcing security policies. Metadata can provide information about the data classification (through confidentiality and integrity labels explained below), the data access policy (e.g., rule-based or lattice-based), authorization (read, write, or execute) among other things. In addition, metadata may indicate special policies, e.g., whether the data is readily available to a role, obtainable to the role after explicit on-line authorization, or not obtainable except through special offline delegation.

Confidentiality labels and integrity labels are two important types of metadata that are associated with all data items input and output to services. The confidentiality label attached to a data item indicates the sensitivity of the data item to unauthorized disclosure, i.e., it indicates the seriousness of the potential fallout resulting from the unauthorized disclosure of the data item. Without loss of generality, we consider three kinds of confidentiality labels: secret, confidential, and unclassified. Integrity labels give an indication of the assurance provided by the entities that have generated or handled the data item in the past. Again, we consider three kinds of integrity labels: high, medium, and low.

As mentioned before, all entities in the layered architecture of Figure 1 are also classified, and hence have associated confidentiality and integrity labels. The classification determines the maximum confidentiality level of the data that an entity can contain/handle (e.g., a confidential service can contain/handle confidential and unclassified data, but not secret data) and identifies the *default* confidentiality and integrity level that is assigned to data created by the entity. The integrity classification assigned to an entity indicates the trustworthiness of the entity (determined, for example, through assurance assessment and from the protection level of the entity). Normally, an entity (e.g., a service or platform) that is higher classified confidentiality-wise would also be higher classified integrity-wise. That is because, an entity handling data of higher confidentiality level must have sufficient level of assurance and protection. Thus, in Figure 2, the services with confidentiality levels secret, confidential, and unclassified would have integrity levels high, medium, and low, respectively.

Classification-enablement of data can be done at varying levels of granularity, and determining the right granularity is a major challenge. At one extreme, every byte of data generated by every entity can have associated metadata indicating the classification; however, that is likely to have significant negative effects of performance. A more reasonable granularity can be obtained by implicitly interpreting the classification of data to be the same as that of its originator (*implicit labeling*) and explicitly attaching the classification metadata (*explicit labeling*) only when the data has to move across classification zones or when the classification is meant to differ from this implicit label. For example, explicit labeling is used if the recipient has a different classification than the originator or if the data’s originator wants to consciously decrease the integrity level of the data. Explicit labeling is also useful when different parts of the same document have to be classified differently. For example, consider an XML document in which certain paragraphs are to be marked secret, whereas others are confidential or unclassified. XML allows structuring data hierarchically as an XML tree and different subtrees may have differing explicit labels depending on the desired confidentiality classification.

It is important to maintain the integrity of metadata. In particular, undetectable modification of metadata should not be possible. One way of ensuring that undetected metadata modification cannot occur is through cryptographic authentication by using a message authentication code (MAC) or a digital signature.

There are multiple ways to associated data and metadata. Conceptually, we assume the data and associated metadata are transported and stored together. For example, in a service-oriented architecture, it is straightforward to define an XML data exchange format that carries payload data along with its metadata in a single XML structure (e.g., <envelope><payload/><metadata/></envelope>). In practice, this assumption can be done away with using today's advanced SOA metadata management solutions (e.g., using a dedicated tagging service).

Metadata has to be maintained throughout the life-cycle of the associated data, i.e., from the time of the data's creation until its deletion, and through all operations (such as read, write, and copy) performed on the data in between. By default, the data inherits the same classification of the entity creating it. If the creating entity wants to label the data in any other way, then it has to invoke special high-assurance services that we explain later in Sections 4 and 5. Even though read operations do not change the metadata, those operations may be logged, especially when extensive auditing is desirable. For write operations (modification of the data or metadata), the classification of the data has to be replaced by the classification of the writer and previous classification metadata has to be automatically invalidated (e.g., using digital signatures and cryptographic hashes). Copy operations must copy both the data and the associated metadata. Similarly, delete operations must delete both the data and metadata.

3.2 Security Policies

In this section, we sketch ways of developing a security policy model for a MLS service-oriented architecture. At a high level, the security policy model can be a a combination of mandatory access control (MAC) and discretionary access control (DAC). MAC policies can be used to specify under what conditions a service belonging to a given classification is allowed to handle incoming data belonging to the various classifications and to generate outgoing data belonging to the various classifications. Thus, there will be a set of policies, each of the form $\{s, f, d, c\}$, where s denotes the service classification, $f \in \{in, out\}$ denotes the data flow direction, d denotes the data classification level, and c is a set of conditions. Finer-grained policies can be obtained by combining the above MAC policies with DAC policies specified at the service instance[2] level. For example, a service generating a data object may specify what other services should be allowed access to that object. The main benefit of such DAC policies is increased security within a classification level. They will usually be formalized using a role-based policy model [15] specifying the required privilege levels for invoking certain service functions. If executing a service function results in high-risk information flows, then only a high-privileged user may invoke that function.

On the logical infrastructure layer, two types of policies can be specified: storage policies and flow control policies. Storage policies define the minimum level of integrity required for a platform to be allowed to store a given data class. An example policy may specify that confidential data can only be stored on a medium integrity platform, which has sufficient protection due to security zones, firewall capabilities, and tamper protection. Flow control policies specify the conditions to be satisfied before data can be transferred from one platform to another platform. For example, a MAC policy may specify that any data transfer from confidential platforms to unclassified platforms can take place only after down-classification or encryption; a further refinement of the policy may, e.g., be obtained through another policy specifying more conditions to be satisfied before data transfer from a Airforce Class 3 confidential platform to the Marine

[2]For a given service classification, there may be several service instances.

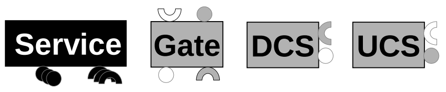

Figure 3: Components for Realizing a Multi-level Secure Broker Infrastructure

Corps Class 3 unclassified platform. For a given source platform p_1 with classification level l_1 and destination platform p_2 with classification level l_2, the combined MAC policy can be expressed formally as a tuple $\{p_1, l_1, p_2, l_2, t, c\}$, where t denotes the required transformations and c denotes the constraints to be satisfied.

3.2.1 Policy Conflicts

When combining multiple policies, there is potential for conflict. One way to resolve conflicts is by creating a hierarchy of policies that govern service usage, in which the child nodes in the hierarchy inherit and refine the policies specified for their parents' nodes. The higher in the hierarchy, the coarser-grained the policies. For example, policies specified at the service instance level may inherit and refine the polices specified at the service classification level.

We now give an example of conflict resolution using precedence relationships similar to [3]. At each level in the hierarchy, there are both MAC and DAC policies. Let us consider a particular service instance. Suppose that, MAC1 and MAC2 denote the MAC policies specified at the service classification and service instance level, respectively. Further, suppose that DAC1 and DAC2 denote the DAC policies specified at the service classification and service instance level, respectively. Then, the overall policy for a given service instance may be a free combination of the MAC1 and MAC2 policies and an ordered composition of the DAC1 and DAC2 policies with the following precedence ordering used to resolve conflicts: (MAC1+MAC2) $>$ DAC2 $>$ DAC1. MAC policies, by definition, are mandatorily enforced. Therefore, there must not be a conflict among the corresponding MAC1 and MAC2 policies for a given service type; if there is one, then there is no alternative but to resolve the conflict by making policy changes. For conflicting DAC policies for a given service instance, the DAC2 (instance) policy is defined to get precedence over the DAC2 (service classification) policy.

Table 1 shows how conflicting policies specified by two different entities can be resolved by establishing well-defined priorities — in this example, by the rule that "the stronger denial policy takes precedence." Conflicting policies may occur, for example, if two coalition partners have conflicting requirements for the same data classification.

4 Building Blocks for Realizing a Multi-level Secure Broker Infrastructure

We now introduce a set of components (Figure 3), which are architectural abstractions for realizing the MLS service brokering infrastructure shown in Figure 2. In our architecture, two types of MLS services, namely, gateways and data classification services, are used for flow control. While classification services reside on a single level, gateways interconnect two levels.

4.1 Multi-level Gateways

A gateway service is a machine that connects two networks belonging to different classification levels. Gateways need to provide high assurance since they are important for enforcing the flow policy between these levels and blocking unauthorized SOAP messages. The unauthorized messages include confidentiality breaching down-flows (i.e., information flow from a higher-confidentiality level to a lower-confidentiality level) and up-flows (i.e., information flow from a lower-confidentiality level to a higher-confidentiality level) that flood the higher-confidentiality level. For high-assurance, gateways must be simple. Hence, only very simple checks are performed directly by the gateways. More complex operations are delegated to other services, such as data classification services, encryption services, authentication services, and authentication verification service. There are two kinds of gateways:

1. *Up-Gateway:* An up-gateway receives a message from an entity in a lower-classified network and transfers the message to a higher-classified network. Before the transfer, the up-gateway attaches metadata indicating the lower-classification (both integrity and confidentiality labels) of the message and asks a higher-classified authentication service to attach an authentication token, which authenticates both the message and metadata. An example of a similar gateway is the information pump proposed in [9].

2. *Down-Gateway:* A down-gateway receives a message from an entity in a higher-classified network and transfer the message to a lower-classified network if and only if the message had been previously down-classified to have the same confidentiality classification[3] as that of the lower-classified network. Prior to the transfer, the gateway asks an authentication verification service to verify the authentication token attached to the message. The gateway blocks all messages that do not have the proper classification or authentication.

4.2 Data Classification Services

Data classification services are high-assurance services that can alter the privileges associated with input data items. The services achieve these alterations to the data by modifying the associated metadata, removing any previously attached authentication token, and re-attaching a new token.

There are two types of data classification services:

1. *Up-classification Services:* An up-classification service decreases the privileges associated with an input data item, by either increasing the confidentiality level or by decreasing the integrity level of the data item. The service can also be used to tighten the access restrictions on the data, e.g., adding a condition that certain data can only be read by a certain role. Like any other MLS service, an up-classification service will have an associated default classification. The service can increase the confidentiality level of input data up to its own and can decrease the integrity level of the data without limits. For example, consider an up-classification service with default confidentiality classification confidential and integrity classification medium. The service can accept data with confidentiality classification unclassified or confidential and output data with confidentiality classification confidential. The service can also accept data of any integrity classification and output data with integrity medium or low.

2. *Down-classification Services:* A down-classification service increases the privileges associated with an input data item, by either decreasing the confidentiality level or by increasing the integrity level of the data item. The service can decrease the confidentiality level of input data without limits and can increase the integrity level of the data up to its own. For example, consider a down-classification service with default confidentiality classification confidential and integrity classification medium. The service can change the confidentiality classification of input data from confidential to unclassified. The service can also change the integrity classification of data from low to medium.

Down-classification services are high-assurance services and a consequence of an important design decision to not allow arbitrary services (with perhaps unverified assurances) to increase the privileges associated with data themselves. They are a step towards automated down-classification, which is essential for minimizing error-prone interventions by human operators while preserving sufficient security. The criteria for allowing such a down-classification need to be restrictive in order to preserve security. Down-classification can only be triggered by services with sufficient assurance where service and its user are authorized for down-classification. Moreover, it is essential to verify the data format to minimize the capacity of covert

[3]Note that the integrity classification of the message may be the same as that of the higher-classified entity.

channels [11][17]. Syntax analysis can often be done automatically in a service-independent way (e.g., using XML-Schema [2]). However, for semantic checking of complex messages, service-specific plug-ins will be required to minimize covert channels. This automated down-classification can cover service messages with well-defined formats. If the messages include larger pieces of free-form data, a human may be involved to check these portions in the context of the service. It is also important that the operations of the down-classification service are properly logged and audited.

The usually restrictive criteria to be satisfied by a service invoking a down-classification service may be relaxed if the data has been encrypted with a high-assurance encryption algorithm that cannot be broken by lower-classified entities. This special case is useful for down-classification of documents with mixed classification. Suppose that only parts of a confidential document need to be classified as confidential, while others can be unclassified. Then, the document as a whole can be down-classified to unclassified if the confidential parts of the document are either encrypted or removed. The down-classification service attaches metadata indicating the confidentiality label to be unclassified and may invoke an authentication service to add an authentication token, which authenticates both the data (including its encrypted portions) and the new metadata. The down-classified and authenticated document may then be transmitted to services with confidentiality classification unclassified through a down-gateway.

While designing policies and services for down-classification, it is important to have mechanisms in place that prevent uncontrolled down-classification. For example, when secret information is down-classified to confidential, it should not be possible for a compromised entity in the confidential domain to automatically further down-classify the information to unclassified. Opting for direct down-classification instead of "cascaded" down-classification may help avoid such uncontrolled down-classification. For example, when secret information has to be down-classified to the unclassified level, it is better for a secret entity to directly down-classify the information rather than doing a cascaded down-classification (i.e., secret to confidential, and then to unclassified). In general, it is advisable to mandate at least one human-authorized down-classification between two automated down-classifications.[4]

5 Implementing Multi-level Secure Services using a Multi-level Secure Broker

We now describe how the conceptual model presented in Section 3 can be realized in today's defense networks, which are structured into mutually isolated network zones, each with their own confidentiality classification. Each service shown in Figure 2 would be deployed in the network zone of the corresponding classification. The monolithic MLS service broker assumed in Figure 2 is decomposed in Figure 4 to include the components described in Section 4 (viz., gateways and data classification services). The decomposed service broker is distributed across all the network zones, and uses the components to connect between zones of different classifications. As a result, the security-criticality of the rest of the service brokering infrastructure can be greatly reduced. Below, we describe a set of rules that can be used to achieve multi-level secure interaction among the services shown in Figure 2. The core idea (shown in Figure 4) is to implement inter-classification connections using the appropriate gateways and data classification services.

Communication between Services of the Same Classification: Communication between two services belonging to the same classification (e.g., the two unclassified services in Figure 4) is brokered just like any regular non-MLS service.

Communication from a Higher-Classified Service to a Lower-Classified Service: Suppose H denotes the classification of the higher-classified service A and L denotes the classification of the lower-classified

[4]Note that this mechanism only protects against subsequent automated down-classification by accident. If a service on a intermediate level is allowed to generate messages to be down-classified, existing data may be leaked into these messages.

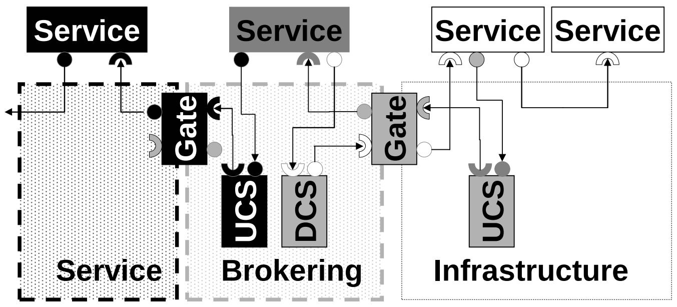

Figure 4: Decomposition of MLS Services

service B. For example, Figure 2 shows a down-flow from a confidential service to an unclassified service. In this case (as shown in Figure 4), the output port of A meant for the down-flow messages is classified at L. The down-flow first goes through a down-classification service (which down-classifies the messages from H to L) and then through a down-gateway. The output from the down-classification service is connected to the input of the down-gateway[5] and the output of the down-gateway is connected to the input port of the service B.

Communication from a Lower-Classified Service to a Higher-level Service: As before, suppose that H denotes the classification of the higher-classified service A and L denotes the classification of the lower-classified service B. For example, Figure 2 shows an up-flow from a confidential service to a secret service. In this case (as shown in Figure 4), the output port of B meant for the up-flow messages is classified at H. The up-flow first goes through an up-classification service (which up-classifies the messages from L to H) and then through an up-gateway. The output from the up-classification service is connected to the input of the up-gateway and the output of the up-gateway is connected to the input port of the service A.

6 Discussion

In this section, we discuss some important special cases and challenges.

6.1 Transporting Lower-Classified Data Through a Higher-Classified Network Zone

Data with a lower-confidentiality classification L can be transported through a network zone with a higher-confidentiality classification H to an entity in the network zone with lower-confidentiality classification L using message authentication codes (MACs). For example, confidential data can be transported (say, for routing efficiency reasons) from an entity in one confidential network through a secret network to another entity in another confidential network. The data goes through an up-gateway when it crosses over from the lower-confidentiality network to the higher-confidentiality network. As explained in Section 4.1, the up-gateway invokes an authentication service, which attaches an authentication token. On the way from the higher-confidentiality network to the lower-confidentiality network, the data goes through a down-gateway, which invokes an authentication verification service to detect any modification in the data or metadata while in transit through the higher-confidential network. If the authentication token is verified correctly, the down-gateway allows the data to re-enter the lower-confidentiality network. Note that, the data does not have

[5]In Figure 4, the "Gate" boxes contain both an up-gateway and a down-gateway. For example, the gray-colored "Gate" box contains an up-gateway controlling flows from the unclassified to the confidential network zone and a down-gateway controlling flows from the confidential to the unclassified zone. The "UCS" box represents an up-classification service and the "DCS" box represents a down-classification service.

through an up-classification service or a down-classification service, since its classification remains the same (lower-confidentiality) throughout.

6.2 Tunneling Higher-Classified Data Through a Lower-Classified Network Zone

Data with a higher-confidentiality classification H can be "tunnelled" through a network zone with a lower-confidentiality classification L to an entity in the network zone with higer-confidentiality classification H using encryption. For example, secret data can be transported (say, for routing efficiency reasons) from an entity in one secret network through a confidential network to another entity in another secret network. The data goes through a down-gateway, which invokes an encryption service, when it crosses over from the higher-confidentiality network to the lower-confidentiality network. On the way from the lower-confidentiality network to the higher-confidentiality network, the data goes through a up-gateway, which invokes a decryption service. The encryption service ensures that the data and associated metadata can be decrypted only by the particular decryption service to be invoked by the up-gateway. Security policies may specify additional tunneling security requirements to be satisfied, such as introduction of dummy data to thwart traffic analysis attacks. The classification of the data remains the same (higher-confidentiality H) throughout the tunneling procedure.

6.3 Data Fusion – Combination and Partitioning

Multiple pieces of lower-classified data can be combined to create higher-classified data. The process that does the combination must itself be higher-classified and will be receiving the individual pieces of data from lower-classified entities (through up-gateways). For example, maps gathered from individual battlefields may be confidential data, but the combined information from multiple maps that gives the overall picture (and perhaps the overall strategy) may be secret.

It is very hard to determine whether an assembly of confidential information pieces into secret information has been done illegally on the confidential layer (unless the combining entity signs the combined data and the data classification of the entity is known). The only way to counter such information fusion attacks is by means of stringent access control (no access to information that is not needed) as well as intrusion detection to detect behaviors corresponding to such illegal information fusion.

Higher-classified data generated or stored at a higher-classified process may be partitioned into multiple pieces, each with a lower classification. As an inverse of the above example, a process with confidentiality classification secret may contain a secret map that gives the overall battle strategy. The process may then automatically partition the map into confidential sub-maps (through a down-classification service), one for each battlefield, and then convey the sub-maps through down-gateways to the respective battlefield lieutenants.

6.4 Time-Sensitive Labeling

An example of time-sensitive labeling is data that is higher-classified (e.g., secret) for a particular duration. Time-sensitive labeling may be achieved by having the data's default classification level correspond to the higher-classification (i.e., secret, in our example), and then automatically down-classifying the data once the duration has expired. Such a scheme is useful, for example, in creating some types of mission-critical data whose classification may be high during the course of the mission and then lowered once the mission is over.

Time-sensitive labeling may be viewed as a form of up-classification or down-classification, and hence, should undergo the same procedure as regular up-classification and down-classification respectively. Specifically, since any entity can up-classify, a lower-classified entity (say, a confidential service) can create time-sensitive higher-classified data (secret for 3 minutes) on its own. On the other hand, since down-classification is more controlled, a higher-classified entity (say, a secret service) must go through a down-classification service to generate time-sensitive data of the *same or lower-classification* (secret for 3 minutes). Note that a secret entity creating data that is labeled “secret for 3 minutes” is actually a form of down-classification and needs similar authorization.

6.5 Legacy Services

There are several challenges in metadata management. Attaching metadata to services is now feasible in service-oriented architectures. However, metadata-enablement of legacy applications that implement newly designed service interfaces is still a challenge, particularly, if the data flow inside the legacy applications is not easy to determine. Another challenge that has not been discussed in detail here is the standardization of metadata exchange formats for allowing different forces to exchange data and metadata.

Metadata attachment depends on the granularity with which services handle security classification. If a service does not use metadata (e.g., a legacy single-level service), then the metadata can be simply tunneled (using encryption) through the service. If the service has separate ports for input and output channels, each handling only a single classification level, then the ports themselves are classified at the same level as the data they handle and there is no need for metadata attachment by the service (as metadata attachment can be done at the ports). On the other hand, if a port outputs data belonging to different classification levels on a per-instance basis, then metadata indicating the data classification level needs to be part of the services output format.

In services that are not label-aware (i.e., do not provide labels for output data), an important challenge is to identify the metadata to be attached at the service ports. Following are two options to address that challenge: (1) attach the highest confidentiality level of any input data to all outputs, or (2) tunnel metadata.

7 Conclusion

In this paper, we have defined a conceptual model for multi-level security in a service-oriented architecture and described how this model can be realized in today’s defense networks structured into mutually isolated network zones, each with their own confidentiality classification. We discussed the MLS-enablement of the service layer through metadata management and specification of security policies. We have introduced MLS architectural abstractions, such as gateways and data classification services, which can help realize a MLS service brokering infrastructure. These abstractions, when implemented with high-assurance, can improve flexibility, enhance information flow control, and reduce the security-criticality of the rest of the service brokering infrastructure.

While this is a first step towards MLS-enabling business processes (at the top-most layer of the stack model shown in Figure 1), the high complexity of business processes and the lack of technologies to provide reliable MLS isolation at that layer leads us to believe that substantial research is required before that goal can be realized.

References

[1] SOAP Specifications. http://www.w3.org/TR/soap.

[2] XML Schema. http://www.w3.org/XML/Schema.

[3] M. Backes, B. Pfitzmann, and M. Schunter. A toolkit for managing enterprise privacy policies. In *Proceedings of the Eight European Symposium on Research in Computer Security (ESORICS)*, volume 2808 of *Lecture Notes on Computer Science (LNCS)*, pages 162–180. Springer-Verlag, Berlin, Oct. 2003.

[4] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. L. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the Art of Virtualization. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles*, pages 164–177, October 2003.

[5] D. Bell and L. La Padula. Secure computer systems: Unified exposition and multics interpretation. Technical Report MTR-2997, MITRE Corp., Bedford, MA, July 1976.

[6] D. E. Bell. Looking Back at the Bell-La Padula Model. In *ACSAC*, pages 337–351. IEEE Computer Society, 2005.

[7] K. Biba. Integrity considerations for secure computer systems. Technical Report TR-3153, Mitre, Bedford, MA, Apr. 1977.

[8] K. Birman, R. Hillman, and S. Pleisch. Building Network-Centric Military Applications over Service Oriented Architectures. In *Proc. of the SPIE Conference on Defense Transformation and Network-Centric Systems*, Orlando, FL, USA, March 2005.

[9] M. H. Kang, I. S. Moskowitz, and D. C. Lee. A network pump. *IEEE Transactions on Software Engineering*, 22(5):329–338, 1996.

[10] P. A. Karger. Multi-level security requirements for hypervisors. In *21st Annual Computer Security Applications Conference*, pages 267–275. IEEE Computer Society, 2005.

[11] I. Moskowitz and M. H. Kang. Covert channels — here to stay? In *COMPASS '94*, pages 235–243, Gaitersburg MD, June 1994. IEEE Press.

[12] H. Mouratidis and P. Giorgini. Analysing Security in Information Systems. In *Proceedings of The Second International Workshop on Security In Information Systems (WOSIS-2004)*, Porto, Portugal, April 2004.

[13] OpenTC Consortium. The OpenTC consortium home page. http://www.opentc.net.

[14] R. Sailer, T. Jaeger, E. Valdez, R. Perez, S. Berger, J. L. Griffin, and L. van Doorn. Building a MAC-based security architecture for the Xen open-source hypervisor. Research Report RC23629, IBM Research Division, June 2005.

[15] R. Sandhu. Role-based access control. In M. Zerkowitz, editor, *Advances in Computers*, volume 48. Academic Press, 1998.

[16] R. Smith. Multilevel Security. In H. Bidgoli, editor, *Handbook of Information Security: Threats, Vulnerabilities, Prevention, Detection and Management*, volume 3, chapter 205. John Wiley, 2005.

[17] B. Venkatraman and R. Newman-Wolfe. Capacity estimation and auditability of network covert channels. In *1995 IEEE Symposium on Security and Privacy*, page 186, 1995.