Security Policy and Information Sharing in Distributed Event-Based Systems

Brian Shand, Peter Pietzuch, Ioannis Papagiannis, Ken Moody, Matteo Migliavacca, David M. Eyers, and Jean Bacon

Abstract. Linking security policy into event-based systems allows formal reasoning about information security. In the applications we address, highly confidential data must be shared both dynamically and for historical analysis. Principals with rights to access the data may be widely distributed, existing in a federation of independent administrative domains. Domain managers are responsible for the data held within domains and transmitted from them; security policy must be specified and enforced in order to meet these obligations. We motivate the event-driven paradigm and take healthcare as a running example, because the confidentiality of healthcare data must be guaranteed over many years. We first consider how to enforce authorisation policy at the client level through parametrised role-based access control (RBAC), taking context into account. We then discuss the additional requirements for secure information flow through the infrastructure components that contribute to communication within and between distributed domains. Finally, we show how this approach supports reasoning about event security in large-scale distributed systems.

Brian Shand
CBCU / Eastern Cancer Registry and Information Centre, National Health Service, Unit C – Magog Court, Shelford Bottom, Hinton Way, Cambridge CB22 3AD, UK
e-mail: `Brian.Shand@cbcu.nhs.uk`

Peter Pietzuch · Ioannis Papagiannis · Matteo Migliavacca
Department of Computing, Imperial College London, 180 Queen's Gate, London SW7 2AZ, UK
e-mail: `{prp,ip108,migliava}@doc.ic.ac.uk`

Ken Moody · David M. Eyers · Jean Bacon
Computer Laboratory, University of Cambridge, JJ Thomson Avenue, Cambridge CB3 0FD, UK
e-mail: `first.last@cl.cam.ac.uk`

1 Introduction

Large event-based computer systems must protect the confidentiality and integrity of data as it passes through them. This task is too large and dynamic to be done in an ad hoc or centralised way: instead, access must be controlled through explicit, distributed security policy. At the same time, policy enables formal reasoning about the security of information within the event system.

In this chapter, we show how security policy support and enforcement can be embedded into event-based systems. By adding fine-grained Distributed Information Flow Control restrictions, we enhance end-to-end security; this allows security analysis to extend beyond the boundaries of the event-based middleware. Reasoning about event security can then support system-wide security audit and information governance, and protect sensitive data in large-scale distributed systems.

Many large-scale distributed applications are best modelled as a federation of domains. A domain is defined as an independently administered unit in which a domain manager has, or may delegate, responsibility for naming and policy specification. The naming of individual users, groups and roles forms the basis of authentication and authorisation.

For example, a national health service comprises many independently administered hospitals, clinics, primary-care practices, etc. The care of a patient may move between domains, from primary care to treatment in hospital. Specialists may be associated with more than one domain, such as hospitals and clinics. Researchers may need to perform statistical analysis on health data. Records may need to be transmitted to auditors and universally accessible Electronic Health Record services. Patient treatment may necessitate the ordering of medication or services from separate domains. Communication within and between domains must therefore be supported, but because of the sensitivity of the data must be strictly controlled.

In such applications the need to communicate is driven by the actions of people (administration of treatment, taking medication), observations (derived automatically from sensors, laboratory test results), and changes in patient state or context (that might indicate emergency situations). Communication is naturally *event-driven*, requiring the asynchronous transmission of data that captures the nature of some occurrence according to application-specific event-naming, specification and management. In some domains patient treatment records are held in databases and the entry of a record may trigger communication. We shall not focus on this particular source of communication; details of how to integrate databases with event-based communication are given in [1, 2, 3].

Security models must reflect the structure and requirements of the application environment. When security policy is enforced within a federation of domains it must be clear where responsibility for each data item lies, and domain managers must be accountable for the information they generate, hold and are obliged to share. General rules may hold about the data that can be sent from the domain responsible for it to other specific domains. Complete data may be sent to a patient's primary care practice and current hospital clinician, but certain attributes may be withheld from other domains, such as pharmacy and audit. Medical data to be used for research purposes must be de-identified, but it must be possible to relate the different records of a given person.

Traditional access control mechanisms tend to focus on authentication and authorisation of individual clients. In highly dynamic, distributed applications, context becomes increasingly important; and the circumstances in which access is appropriate may also need to be captured. Context may include absolute and relative times, prerequisite actions and procedures, credential checks, the relationship between the principal accessing the data and the principal to whom the data relates, any individual exclusions that may have been placed on the data and whether an emergency situation exists.

Often data is highly sensitive, and must be protected, yet must also be delivered in a timely manner to those parties that have a need and right to know, for example, clinicians carrying out emergency procedures. Security policy enforcement must not unduly delay the transmission of data. Heavyweight security procedures tend to be bypassed by human users when there is an option.

Data must be protected, not only from inappropriate clients end-to-end, but also during transmission or processing by intermediate infrastructure components. Data may remain confidential for as long as a human lifetime, or longer.

In summary, policy specification and enforcement must therefore capture: (1) Data is communicated asynchronously in the form of messages that embody events. (2) Security policy must ensure that event type names and specifications are unique within a system and that their evolution is controlled. (3) When events are structured, separate components of events may need different protection. (4) Security policy must specify the principals that are authorised to access data. In an event-based system, control is enforced in terms of who can send and receive events. (5) Parametrised RBAC has the required properties of simple administration and subtle expression of policy. In a multi-domain system, inter-domain negotiation establishes access rights in terms of roles within domains. (6) It must be possible to qualify access rights by context. (7) Domain managers must be supported in their obligations to protect data. General domain-level rules must be established and enforced on transmitted data. (8) Mechanisms must be in place to secure the flow of data through infrastructure components. (9) Security policy enforcement must not delay communication unduly. Real-time communication may be required by some clients. (10) Security policy should support analysis and reasoning about event security.

Section 2 of this chapter presents our model of event-based communication, showing how access control policy can be incorporated. In Section 3, we address the challenges of secure event type management for Internet-scale environments. Section 4 extends event access controls, to give end-to-end event security protection, even inside event broker nodes and event recipients. Section 5 demonstrates how security policy can allow formal reasoning about event security, and shows how end-to-end security enhances this. We conclude in Section 6.

This chapter focuses on policy expression and enforcement. As we introduce topics, we outline briefly the assumptions we make about the system architecture and implementation within which policy must be enforced. Other papers on the architecture and engineering of distributed event-based systems give further detail [4, 5, 6].

2 Integrating Access Control into Event-Based Systems

In this section we discuss both client-level and communication security. First we cover how the event data, that is to be communicated and accessed, is defined, named uniquely, and its evolution controlled.

2.1 Event Type Specification and Ownership in a Single Domain

Before events can be advertised, published or subscribed-to they must be defined and their specification made available system-wide. We assume that each event type has an owner who registers the type definition with the local domain's secure server. We assume a secure server per domain capable of supporting a Public Key Infrastructure (PKI). When a type is registered, a public/private key-pair is associated with it and the public key is bound into its definition; the type owner holds the private key. The management of the type, for example any changes that need to be made to its specification, are controlled within this PKI.

We model events as structures consisting of multiple named parts (or attributes). This fine-grained structure is useful for characterising event filters, and for restricting which subscribers may receive which parts of an event.

In Section 3.1 we discuss how events may be made known system-wide in a multi-domain system. We can then assume that event type specifications are available to all application domains, statically through a registration service, and dynamically when publishers advertise that they are live and ready to publish. First, we describe the basic communication mechanism and access control policy and mechanisms.

2.2 Event Communication: Advertise, Publish and Subscribe

Having defined event types, communication can then take place. We assume a publish/subscribe paradigm for event-based communication, [7]. Much of our research has been based on a large-scale pub/sub architecture Hermes, [8, 9]. The assumption is that event transmission is carried out by a shared event broker network, independent of, and shared by, the client domains that use it. An overview of the approach to secure transmission in Hermes is given in [10].

Here, we modify our assumptions about the broker network for applications with long-term confidentiality requirements as described in Section 1. We do not envisage a large-scale broker network independent of the application domains; each broker belongs to an application domain, from which its security properties, including trust, are inherited. A client-hosting broker acts on behalf of publishers and subscribers in its domain and enforces the security policy of the domain. For load balancing, more than one broker per domain can be used.

An event publisher, that becomes ready to publish, multicasts an advertisement system-wide via its domain-local event broker. The broker checks and enforces authorisation policy relating to the advertisement, see Section 2.5. Initial authentication of publishers, subscribers and brokers is done using public key pairs bound to identity certificates (e.g. X.509) [5]. There can be any number of publishers per event type and their advertisements make them known to the event-brokers.

Principals with an interest in receiving events of an advertised type (subscribers) issue a subscription, together with a subscription filter, via their domain-local event broker. Again the broker checks that the subscriber is authorised to make the subscription, see Section 2.5.

Subscriptions are routed to the relevant publishers' brokers. When an event is published, it is matched against the subscription filters and sent to those subscribers whose filters match the event attributes.

For example, a pathology laboratory would advertise its intention to publish pathology reports to its event broker, and would then publish each pathology report as an event. Authorised doctors and clinical researchers would subscribe to pathology events, with subscription filters for their patients or research studies.

We now describe how this procedure is qualified by security policy enforcement, controlling who can publish and subscribe and the data that can leave a domain and be sent to each destination domain.

2.3 Role-Based Access Control

Role-Based Access Control (RBAC) [11] is a standard technique for simplifying scalable security administration by introducing *roles* as a linking concept between *principals* (i.e. users and their agents) and *privileges*. Privileges, such as the right to use a service or to access an object managed by a service, are assigned to roles. Separately, principals are associated with roles. This separates the administration of people, and their association with roles, from the control of privileges for the use of services (including service-managed data). The motivation is that users join, leave and change roles in an organization frequently, and the policy of services is independent of such changes. Service developers need only be concerned with specifying access policy in terms of roles, and not with individual users.

RBAC is particularly suitable for securing event-based systems because the process of agreeing the notions of role between decoupled participants within an event-based system closely parallels the process by which those same decoupled participants must agree on how to interpret events. Both are well suited to operation in widely distributed, multi-domain systems. Here we focus on securing access to the communication service using RBAC.

Authentication into roles must be securely enforced to control the use of all protected services and access to the data they manage [12]. Domain managers, or their delegates, specify communication policy in terms of message types and roles; that is, which roles may create, advertise, send and receive which types of message, see Section 2.2.

Inter-domain communication is achieved through negotiated agreements, expressed as access control policy, on which roles of one domain may receive (which attributes of) which types of message of another. This negotiation must also take into account any general domain-level policies regarding data transfer, see Section 3.2.

Standard RBAC causes principals to be anonymous (i.e. the privileges available to role holders do not depend on their identity), whereas parametrised RBAC gives the option of anonymity or identification, for example, treating-doctor(*hospital-ID*, *doctor-ID*, *patient-ID*). By means of parametrised RBAC it is possible to capture relationships and patient-specified exclusions, as may be required by law. The use of parametrised roles can also help to avoid an explosion in the number of roles required when RBAC is used in large systems. For the communication service, RBAC policy indicates the visibility (to roles, intra- and inter-domain) of specified attributes of message types.

The fact that advertisement is required before messages can be published, and that both are RBAC-controlled, prevents the spam that pervades email communication between humans. Without such control, denial-of-service through publication or subscription flooding could degrade large-scale inter-software communication in the same way that it consumes resources in email management. With our approach, a spammer could only be an authorised, authenticated member of a role and therefore could be held accountable.

2.4 OASIS Role-Based Access Control

In this section we provide a brief introduction to the Open Architecture for Secure Interworking Services (OASIS) [12], developed at Cambridge and used as the basis for our research. OASIS provides a comprehensive rule-based means to check that users can only acquire the privileges that authorise them to use services by activating appropriate roles. Although we use OASIS policy in subsequent sections, we aim to highlight fundamental principles that must guide any policy implementation.

A role activation policy comprises a set of rules, where a role activation rule for a (target) role r_t takes the form

$$r_1, .., r_n, a_1, .., a_m, e_1, .., e_l \vdash r_t$$

where r_i are prerequisite roles, a_i are appointment certificates (most often persistent credentials) and e_i are environmental constraints. The latter allow restrictions to be imposed on when and where roles can be activated (and privileges exercised), for example at restricted times or from a restricted set of computers. Any predicate that must remain true for the principal to remain active in the role is tagged as a *role membership condition*. Such predicates are monitored, and their violation triggers revocation of the role and related privileges from the principal.

For example, a role activation rule could support patient referral between consultants, by allowing dynamic assignment of a treating-doctor target role, given an appropriate consultant-referral appointment certificate.

An authorisation rule for some privilege p takes the form

$$r, e_1, .., e_l \vdash p$$

An authorisation policy comprises a set of such rules. OASIS has no negative rules, and satisfying any one rule indicates success.

In our example, role authorisation rules could allow treating doctors and clinical researchers respectively the privilege priv-clinical of reading clinical data for the appropriate patients.

OASIS roles and rules are parametrised. This allows fine-grained policy requirements to be expressed and enforced, such as exclusion of individuals and relationships between principals, for example treating-doctor(*hospital-ID*, *doctor-ID*, *patient-ID*), as outlined in Section 2.3.

2.5 Access Control Policy for Publish/Subscribe Clients

The most general access control requirement in pub/sub relates to how clients connect to the pub/sub service (e.g. a local broker of a distributed broker network), and make requests using its API. This implements security at the pub/sub network edge. Many pub/sub systems [7, 13] include the following service methods in one form or another:

define(*message-type*)
advertise(*message-type*)
publish(*message-type*, *attribute-values*)
subscribe(*message-type*, *filter-expression-on-attributes*)

Some policy languages will only be able to provide a coarse specification of the client privileges required to use the API. In OASIS RBAC, the authorisation policy for any service specifies how it can be used in terms of roles and environmental constraints, and parametrised roles can be used to effect fine-grained control.

OASIS policy indicates, for each method, the role credentials, each with associated environmental constraints, that authorise invocation. OASIS role parameters can be used to limit privileges to particular *message-type*s. The define method is used to register a message type with the service and specify its security requirements at the granularity of attributes. On advertise, publish and subscribe, these requirements are enforced. We can therefore support secure publish/subscribe within a domain in which roles are named, activated and administered.

3 Multi-domain Security Architecture

Multi-domain systems can be structured as a federation of autonomous domains. This distributed approach requires a model for secure event type definition and management, and an approach to inter-domain communication control. Before covering these issues, we first clarify our domain model.

A domain-structured OASIS system is engineered with a per-domain, secure OASIS server, see [4] for details. Also, each domain has a policy store containing all the role activation and service-specific authorisation policies. This avoids the need for every service to independently perform authentication and secure role activation, allowing simple service implementation. The domain's OASIS server carries out all per-domain role activation and monitors the role membership rule conditions while the roles are active. This effectively concentrates role dependency maintenance within a single server and provides a single, per-domain, secure service for managing inter-domain authorisation policy specification and enforcement. For robustness, this would typically be a replicated service, with fail-over to a hot standby server.

3.1 Management of Event Names, Types and Policies

As introduced in Section 2.1, a mechanism is needed to agree on the naming of event types, when constructing policy-secured, pub/sub systems. Extending for multiple domains, we assume that domains are allocated unique names within the system as a whole and that roles are named and managed within a domain. Each domain provides a management interface through which role activation policies and service authorization policies can be specified and maintained.

A group of domains may have a parent domain from which an initial set of role names and policies is obtained. For example, health service domains may start from an initial national role-set. The domain management

Table 1 Contents of a secure event type definition

Name tuple:	1 Type issuer's public key
	2 User-friendly name
	3 Version number
Body:	4 Attributes
Digital signature:	5 Delegation certificates
	6 Digital signature

interface allows local additions and updates, for example when national government policy needs to be customised for implementation regionally. As mentioned above, parametrised roles allow domain-specific parameters, for example treating-doctor(*hospital-ID*, *doctor-ID*, *patient-ID*). This allows relationships and exclusions to be captured as well as avoiding excessive numbers of roles in large-scale systems.

For a multi-domain pub/sub system, we introduce a format of event type definition that binds the type name and definition together in a secure manner. Public key cryptography is used to guarantee the authenticity and integrity of this type information. To achieve this we require that all participating brokers in the pub/sub system have a key-pair. We can thus require that event-type issuers incorporate this public key into the type name. This facilitates an event naming scope for each particular type issuer. Since event type names include a public key, it is intuitive that the event type definitions should be signed by the corresponding private key. This binds the type definition to the type name, and facilitates verification of event type integrity and issuer authenticity. Thus we protect the system against forged or tampered event type definitions. We also reduce the chance of accidental name collisions, and provide a unique handle through which policy can refer to the names of types and attributes. An outline is given below and further details are in [14, 15, 16, 10].

The six items that make up a secure event type definition are shown in Table 1. Items 1–3 identify the name of the type. The Attributes item 4 indicates the core event type definition, and items 5 and 6 contain a digital signature of the event type. Table 2 illustrates this for a pathology report type.

Adding a public key to the type name eliminates event name conflicts. While this is desirable, a user-friendly name is also maintained. The user-friendly name is able to encode useful aspects, such as hierarchical naming, e.g. `uk.nhs.path_report`. This enables administrative grouping of type definitions across multiple event type owners. Orthogonal to both of those concerns is type evolution, hence the provision of a version number. Releasing a new version of an event type definition will not conflict with previous instances still in use within the publish/subscribe system. Indeed, to avoid

Table 2 Example of a secure event type definition for pathology reports

1 Type issuer's public key
```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Comment: Public key for NHS Information Authority ...
```
2 User-friendly name
```
uk.nhs.path_report
```
3 Version number
```
31ab47dc-4e52-4509-92ec-39f6e603d6e6
```
4 Attributes

`patient-ID`	`0640d3d2-0b2b-45db-b66b-0f4200cd8358`	`String`
`patient-name`	`de312646-e940-466d-a71f-fa9122891d5a`	`String`
`hospital-ID`	`e4ab7b23-87dc-44ca-bf7c-cba5cc5ce6f7`	`String`
`lab-number`	`1dd5f593-1bc6-4f53-89ef-5f2a38b1d4ec`	`String`
`sample-receipt-date`	`a273e069-7cb6-48ca-b827-9bb11d449dbf`	`Date`
`path-report-text`	`b831c325-4d63-4d98-b7c0-782a1d4a1c44`	`Binary`
`SNOMED-CT-codes`	`3514a33d-5173-453c-8b5a-11e8ead71761`	`String`

5 Delegation certificates
```
[Delegation certificate from NHS root to NHS Information Authority]
```
6 Digital signature
```
-----BEGIN PGP SIGNATURE----- ...
```

race conditions for version numbers when multiple type managers are releasing updated event definitions, a UUID scheme is used for version numbers. UUIDs are 128-bit values, generated by a collision-avoiding algorithm.

Item 4 in the event type definition describes the event type structure. Each attribute definition itself consists of a user-friendly name, a unique identifier (UUID), and an attribute type identifier. The set of types supported depends on the subscription filter language used. Friendly names for attributes are intended to be used by clients of the publish/subscribe system, e.g. *path-report-text*, whereas the UUID is used by intermediate brokers during distributed event routing. The friendly names only need to be unique within the context of one particular version of an event type definition. When a publisher-hosting broker receives an event to route, it looks up the friendly names used by its client using the publisher's event type definition. This allows the UUID fields to be correctly populated. The reverse of this process occurs at subscriber-hosting brokers. The UUIDs allow multiple versions of an event type to exist within the publish/subscribe system at a point in time.

The digital signature of an event type provides a guarantee of the authenticity and integrity of the type definition. The signature is calculated over items 1–4 in Table 1. It thus binds the type definition and the name tuple.

The delegation certificates (item 5) facilitate Internet-scale management of event types. Since key-pairs are involved in signing event types, without delegation certificates the type owner would need to re-sign all updates to the type. Delegation certificates facilitate a digitally-signed path of trust from the original event type owner to type managers: parties that are allowed to update event types on their behalf. In Table 2, the UK National Health Service (NHS) authorises the NHS Information Authority to securely manage certain event types. The delivery of delegation certificates to type managers can be performed out-of-band, e.g. by secure email. The delegation certificates also provide the means to specify fine-grained access rights. Our prototype implementations have typically supported rights such as addAttribute, removeAttribute, editAttributeName and editAttributeType.

3.2 Inter-domain Communication Control

When security policy is enforced within a federation of domains, the manager of each domain is responsible, and accountable, for the information held within it and propagated to other domains. Encryption can be used to protect data during communication but the confidentiality of healthcare data persists for as long as a human lifetime, or longer. On that timescale, encryption keys may be compromised by being exposed or becoming insufficiently secure to withstand attacks from the increased computing power available.

Under these requirements, some of the much-researched optimisations advocated for publish/subscribe systems become problematic. Use of a shared external broker network, with content-based routing, is ill-advised and direct communication between domains is preferable. We envisage that a given message will be sent to a handful of domains, unlike in global-scale pub/sub information dissemination.

Some data may have a short lifetime, such as that transmitted in battlefield scenarios and emergency situations. In these situations it may be appropriate to use an external broker network, perhaps assembled in an ad hoc fashion from mobile components. Messages in transit are assumed to be protected by a standard, link-level scheme such as TLS [17]. At the client level, the attributes of messages may be encrypted separately and decrypted by brokers on behalf of their authorised clients. Some brokers may be less trustworthy than others. Here, client-level encryption is appropriate, and essential, so that untrusted brokers cannot decrypt highly sensitive attributes. Details of how this can be achieved are summarised in [16].

Sensitive data should not be communicated to principals in other domains unless authorised by policy. This can be established when RBAC-based authorisation is set up by negotiation between domains. Rather than sending entire events, containing attributes that are sensitive but protected by encryption, it may be preferable to remove or to lower the precision of some of the attributes. Similarly, brokers receiving events from other domains should enforce client access rights according to RBAC policy. These issues have been explored in detail in [2, 3].

4 End-to-End Security with Information Flow Control

Thus far, we have outlined a security model for event-based systems, in which multi-domain RBAC protects access to both individual events and the event type system. This approach of integrating access control into event-based systems provides strong, consistent security at the boundaries between publishers, event broker nodes and subscribers, and between federated domains. However, it has limitations too:

- A large base of event broker code must be trusted not to leak information, e.g. accidentally disclosing events to unauthorised subscribers, or disclosing one subscriber's subscriptions to another subscriber.
- If services such as network monitoring or complex event detection are added to the event brokers, they need to be trusted too.
- Once data is released to a subscriber, information about its security is lost, limiting the potential for end-to-end security, e.g. in follow-on messages.

In this section, we show how these limitations can be overcome, using Distributed Information Flow Control (DIFC) to guard all event processing within the event broker nodes. This approach also enhances long-term data security, because unauthorised data cannot be accessed or stored within a DIFC framework, even in encrypted form.

Figures 1a and 1b illustrate this distinction, between security controls at the boundaries of the event system, and continuous, end-to-end security tracking with DIFC. In both figures, publishers and subscribers communicate via trusted event brokers in two domains. Two publishers in Domain 1 send Event A and Event B respectively. The subscriber in Domain 2 has a subscription which matches both events, but Event A is blocked for security reasons. On receiving Event B, the subscriber attempts to save it to a file, and republish it as Event C – both of which actions should be blocked for security reasons. At the same time, a monitor component Σ in Domain 2 emits an event count, which is triggered by Event A, but not Event B. In the figures, the large circles represent the boundaries of the event brokers, and a shaded background marks trusted code. Event flows with the same line style are protected with the same security restrictions. Tap icons show potential data leaks. Solid small circles are security checks.

In Figure 1a, security is enforced only at the boundaries of the event brokers. This correctly blocks Event A from the subscriber, and allows Event B through. However, the subscriber can then republish or leak to a file the confidential contents of Event B. Furthermore, there is no straightforward way to ensure that the event count contains no confidential data.

In Figure 1b, DIFC tags allow continuous end-to-end tracking of data security. These not only correctly block and allow Events A and B respectively, but also ensure that subsequent uses of Event B retain the same security properties (dashed lines), preventing republishing or saving to a file. Furthermore, the event count can be shielded from confidential data, by using a small,

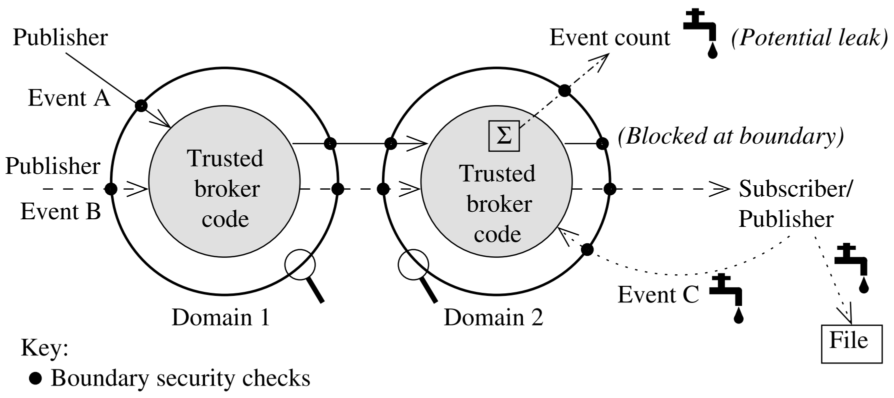

Fig. 1a Boundary security between event publishers, brokers and subscribers

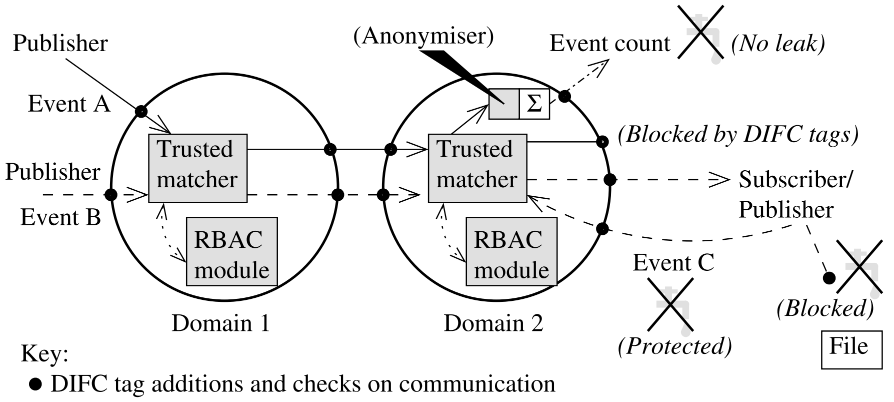

Fig. 1b End-to-end security with Distributed Information Flow Control

isolated anonymiser component before computing the sum; this reduces the need for trusted code in the event broker.

4.1 Distributed Information Flow Control

Information Flow Control [18] is a security technique that uses security labels to track and guard all information and processing. Information, that must have a security label, can flow only to processes with compatible security labels, and any resulting information is restricted by the labels of both the process and the original information – unless special privileges are exercised, such as declassification. For example, this could block confidential patient data from a process with insufficient confidentiality privileges.

Distributed Information Flow Control (DIFC) [19, 20] applies this technique to distributed systems, by allowing new security label constituents to be created dynamically by processes, instead of relying on a centrally predefined set.

We express each security label as a set of tags, each representing an atomic security concern. Each tag is used either for confidentiality or integrity purposes: data may ordinarily only increase in confidentiality, and decrease in integrity as it is processed, unless special privileges are exercised for *declassification* (lowering confidentiality) or *endorsement* (higher integrity).

Notationally, we write a DIFC security label as (S, I) with confidentiality component S and integrity component I, where S and I are each sets of tags. Information may flow from a to b with labels (S_a, I_a) and (S_b, I_b) only if $S_a \subseteq S_b$ and $I_a \supseteq I_b$. For example, this restriction would guard receipt of event data by a process. To gain access to data, the process may need to use privileges to increase its confidentiality (or secrecy), or lower its integrity – these privileges are considered weaker than their opposites, declassification and endorsement above.

To illustrate this practically, we use the following clinical example:

> At a surgical outpatient appointment at Addenbrooke's Hospital, the consultant Miss Ali takes a biopsy from a patient (Mr Patterson), and sends it to the pathology service for further analysis. When she receives the results two days later, she refers the patient to Dr Brown, a consultant oncologist, for chemotherapy, including the pathology report and her own comments. The pathology report is also released to a cancer researcher, Dr Chen, for analysis in an anonymised patient cohort.

The three events in this example are: e_1: the pathology request sent in conjunction with the physical biopsy, e_2: the pathology report, a `uk.nhs.path_report` event as in Table 2, and e_3: the pathology report with comments.

Here, a confidentiality tag $t_{\text{identifiable}}$ could be used to restrict access to the identifiable patient data in the pathology report event e_2, such as the *patient-name* or *patient-ID* attributes. Non-identifiable data, such as *SNOMED-CT-codes* and *path-report-text* could be restricted with other confidentiality tags. Finally, the pathology laboratory would add an integrity tag t_{pathlab} to all attributes which it added, to demonstrate the authenticity of the report.

The overhead of DIFC label checks is small [21]. Computationally, the complexity is linear in the number of labels; results can be cached for efficiency.

In a distributed environment, we treat tags as local entities, ensuring that network outages need not stall the event processing system. Tags are created within an event broker node, and the creator of a tag can delegate privileges to other processes, for adding the tag to labels or removing it from them. (In [21], we detail how we can do this in an IFC-safe way, i.e. without the privilege delegations themselves leaking information.) When event broker nodes exchange labelled data, they need to translate any tags with global meaning into the local representation, e.g. using DNS-backed URIs as global names. However, the extra communication overhead is small, and local DIFC checks can then efficiently treat all tags as local.

4.2 Event Security with DIFC

DIFC is well-suited to event processing systems: DIFC can protect all event-based communication; any remaining communication channels or storage mechanisms that persist between events should be blocked or DIFC mediated. This combines nicely with our model of multi-part events in Section 2.1, as each event part (or attribute) can have its own security label. By restricting event handling to a controlled API, this allows code within an event broker to access only those parts of an event for which it has tag privileges. The existence of any other parts will effectively be hidden.

For efficient operation, we allow processes within an event broker to specify in advance that they will always release an event (possibly modified) for other processes to operate on [21]. Without this mechanism, all event parts would need to be labelled with all of the confidentiality tags (and only a subset of the integrity tags) from the releasing process's label, resulting in unnecessary complexity of design and operation.

DIFC for events improves the security of general purpose event processing systems, including event-based middleware, offering strong protection against communication leaks, and efficient end-to-end-security. The bulk of the event processing system can consist of code with only limited privileges.

Figure 1b illustrates the effect of DIFC tracking within an event broker: the security labels of multi-part events are tracked throughout an event-based system, both within and between event brokers (shown by the different line styles). If publishers and subscribers use a DIFC-enforcing API to interact with events, then all of their subsequent operations are protected too – event publication C must respect event B's security and integrity labels, unless additional declassification or endorsement privileges are exercised.

4.3 Enforcing OASIS Security with DIFC

Event systems can enforce Role-Based Access Control using DIFC restrictions. Our approach to this supports dynamic role assignment and revocation, and parametrised roles. We illustrate this using OASIS RBAC (outlined in Section 2.4), thus providing end-to-end DIFC support within the multi-domain security architecture of Section 3.

In essence, whenever an OASIS role is activated, DIFC confidentiality tags are created for the resulting privileges, and any event-handling processes that need access to the data are granted the appropriate confidentiality tag privilege. Trusted processes, such as broker-to-broker communication or shared matching, may need to be granted declassification privileges too, depending on the event system design.

In our example from Section 4.1, Miss Ali submits a subscription for all clinical events at Addenbrooke's Hospital pertaining to patient Mr Patterson. The policy for this message-type requires that a subscriber must have privilege priv-clinical(*patient-ID*) to receive events with the corresponding *patient-ID*.

Mr Patterson's pathology report is published by the pathology lab: the identifiable patient data is labelled with confidentiality tag $t_{\text{identifiable}}$, and the clinical details have tag $t_{\text{MrPatterson}}$ corresponding to the privilege of being able to handle clinical data with Mr Patterson's *patient-ID*, i.e. priv-clinical("Mr Patterson"). Relevant parts of the pathology event are shown in Table 3. Creation of this event was OASIS-mediated; the trusted OASIS support code retains the right to grant privileges over $t_{\text{MrPatterson}}$ on the local event broker node.

Table 3 A multi-part pathology report with security labels

Label (S, I)	Event part name	Event data
$(\{t_{\text{identifiable}}\}, \{\})$	patient-ID	1234567768
$(\{t_{\text{identifiable}}\}, \{\})$	patient-name	Roger John Patterson
$(\{\}, \{\})$	hospital-ID	RGT01
$(\{\}, \{t_{\text{pathlab}}\})$	lab-number	A10/21367
$(\{\}, \{t_{\text{pathlab}}\})$	sample-receipt-date	2010-02-04
$(\{t_{\text{MrPatterson}}\}, \{t_{\text{pathlab}}\})$	path-report-text	CASE HISTORY: Large solid tumour with some papillary elements on right side of trigone. Query TCC. MACROSCOPIC: ...
$(\{t_{\text{MrPatterson}}\}, \{t_{\text{pathlab}}\})$	SNOMED-CT-codes	302512001 |Bladder|, ...

When the event broker receives the pathology report, the event subscription causes role treating-doctor("Addenbrooke's Hospital", "Miss Ali", "Mr Patterson") to be activated, yielding privilege priv-clinical("Mr Patterson"). The corresponding DIFC tag $t_{\text{MrPatterson}}$ is retrieved, and the subscription's process is granted access to it. More precisely, the process receives the privilege to add $t_{\text{MrPatterson}}$ to its confidentiality component, and effectively spawns a lightweight copy of itself, allowing it to inspect the event contents without prejudicing its ability to operate with a lower confidentiality level on subsequent events.

When Miss Ali refers the patient to Dr Brown, he too obtains the appropriate treating-doctor role, and thus the right to read the pathology report in question, as well as the patient demographics tagged $t_{\text{identifiable}}$. The researcher, Dr Chen will only be granted privilege priv-clinical("Mr Patterson") to read the pathology report, but no access to the identifiable patient data. If patient linkage is required for the research, a trusted process can anonymise the *patient-ID*, adding a new event part *anonymised-ID* that Dr Chen can read.

Dynamic privilege assignment adds an extra complexity glossed over above: instead of using a single tag t for a privilege at all points in time, a sequence of tags $t_1, t_2, t_3, \ldots$ is created as needed whenever the roles actively supporting the privilege change. This is needed to enforce OASIS privilege revocation, without mandatory revocation of DIFC tag privileges (as such a mechanism would violate DIFC restrictions). Therefore, when a role is revoked, a new DIFC tag t_{n+1} is used for subsequent data, essentially making the revokee's existing tag t_n worthless, except for processing that has already been authorised.

DIFC tags provide efficient, local enforcement of privileges. The additional overheads of tag management are minor, as they affect only the relatively expensive OASIS-mediated activities of exercising privileges and roles, and role revocation. OASIS policy also provides effective translation between local DIFC tags and domain-level privileges.

We have shown that DIFC enforcement of RBAC restrictions can provide fine-grained end-to-end security for event flows, both within event processing broker nodes, and for event publishers and subscribers subject to DIFC restrictions. This also reduces the trusted code footprint of the event processing system, supporting the principle of least privilege.

As the following section shows, OASIS privileges or DIFC tags in the application domain integrate well with other aspects of secure system design, such as physical security.

5 Reasoning about Event Security

A policy-based approach to event security enables formal reasoning about information flow in an event-based system. This reasoning can be performed either at the level of the policy language, such as OASIS rules (Section 2.4), or in terms of their projection into DIFC labels, where DIFC enforces event security (Section 4.3). Policy language reasoning is generally simpler, but DIFC reasoning offers extra flexibility, such as integration with multiple policy languages, direct links to environmental factors such as physical location, and better support for end-to-end policy enforcement.

5.1 Policy-Based Reasoning

Linking RBAC privileges to particular message-types (Section 2.5) immediately allows straightforward reasoning about certain security properties. For example, suppose the pathology message-type's policy requires privilege priv-clinical(*patient-ID*) to access the event part named "path-report-text", for events with that *patient-ID*, and the only OASIS policy rules yielding priv-clinical permissions are:

$$\text{treating-doctor}(\textit{hospital-ID}, \textit{doctor-ID}, \textit{patient-ID}) \vdash \text{priv-clinical}(\textit{patient-ID})$$
$$\text{clinical-researcher}(\textit{study-ID}, \textit{researcher-ID}, \textit{patient-ID}) \vdash \text{priv-clinical}(\textit{patient-ID})$$

Then we can reason that only treating doctors and authorised clinical researchers can access the pathology text, as long as the assignment of the treating-doctor and clinical-researcher roles is suitably restricted.

Similarly, if consultants can refer patients to each other through consultant-referral role appointment certificates, then these can be supported with the following OASIS policy rule:

$$\begin{aligned}
&\text{treating-doctor}(\textit{referring-hospital-ID}, \textit{referring-doctor-ID}, \textit{patient-ID}),\\
&\text{consultant-doctor}(\textit{hospital-ID}, \textit{doctor-ID}),\\
&\text{consultant-referral}(\textit{referring-doctor-ID}, \textit{doctor-ID}, \textit{patient-ID})\\
&\quad \vdash \text{treating-doctor}(\textit{hospital-ID}, \textit{doctor-ID}, \textit{patient-ID})
\end{aligned}$$

We can then conclude that the consultant referral from Miss Ali to Dr Brown described in Section 4.1's example will enable Dr Brown to read the pathology report's text when the patient is referred – as long as he receives a copy, and his treating-doctor role is still valid.

This sort of reasoning focuses principally on confidentiality restrictions, i.e. guaranteeing that data will not be received by unauthorised recipients. This focus is appropriate for our target scenarios, such as protecting healthcare data against disclosure. However, protecting integrity and availability are equally important for safety-critical systems.

Integrity can be linked to the event type, implying that all publications of a particular message-type are of high integrity, or that only high-integrity messages will have certain attributes set. Limited integrity restrictions can also be implemented indirectly in OASIS, by treating them each as a forced low-integrity role that is held by all principals except for a few high-integrity entities, like the high-integrity pathology lab in our example. However, these approaches require event publishers to manage their own integrity tracking, to ensure that they do not inadvertently raise the integrity level of data from an external source. DIFC restrictions provide much more robust, fine-grained support for protecting both confidentiality and integrity.

OASIS policy checking can also prove a limited version of availability: it can show that no access control restrictions will prevent the delivery of a particular event to a subscriber. However, this depends on the underlying availability of the event processing system and supporting infrastructure.

5.2 Event Security Reasoning with DIFC Labels

DIFC labels allow direct reasoning about confidentiality and integrity properties. The benefits of this include: (1) DIFC restrictions protect against many security vulnerabilities directly, (2) the correctness of policy enforcement can be verified in terms of its transformation to DIFC tags, (3) fine-grained, end-to-end confidentiality/integrity reasoning is supported, (4) multiple policy languages can translate to common enforcement tags, and (5) DIFC tag analysis can facilitate reasoning about physical security.

The greatest advantage of translating policy into DIFC restrictions is that the basic Information Flow Control restrictions are not violated. This system-wide property subsumes many individual security checks, and its power should not be understated. For example, this means that checking "No unauthorised users may access patient data" translates to checking what program code can declassify tagged patient data in the event processing system, and the safeguards on corresponding role assignment.

Relying on a common enforcement mechanism allows safe, consistent policy enforcement. Correctness of the OASIS policy enforcement mechanism can be assessed simply in terms of its transformation of policy rules to DIFC tags. Provided this transformation is correct, DIFC restrictions will then enforce the OASIS RBAC guarantees. Furthermore, policy-based reasoning will apply to the resulting DIFC tags too.

End-to-end confidentiality and integrity are possible, because publishers and subscribers can process event data within a DIFC-enforcing framework. This makes many security checks on client code unnecessary, as this code is protected by the DIFC framework. For example, a subscriber cannot forward patient data to an unauthorised third party, because DIFC restrictions will automatically block access to that data. Event subscribers (and untrusted code within the event broker system such as network status monitors) are also prevented from accessing or retaining confidential data, even in encrypted form; this provides essential protection against the long-term risk of disclosure.

This end-to-end support is also potentially more fine-grained than the underlying policy may suggest. If an event monitor were introduced, to count the number of pathology reports by anatomical site (e.g. the number of pathology reports with a SNOMED-CT code of "302512001 |Bladder|"), the monitor could declassify the non-disclosive result immediately after checking the "SNOMED-CT-codes" field. Then the only program code that would need to be trusted would be the code reading that field. A more coarse-grained policy-based analysis would instead suggest the entire monitoring codebase, and all subsequent messages, as potential data leaks.

DIFC tags can also protect against cross-contamination of patient data: if a recipient uses confidentiality privileges to merge two patients' data, the result will be contaminated with tags for both priv-clinical privileges, preventing accidental release as a single patient's data.

Multiple policy languages can be supported, provided that they can all translate their requirements into DIFC tags. This may limit the potential to perform security reasoning at the policy level – unless the DIFC tags of one policy language can be translated back into another, or the tag overlap between the languages can be characterised. On the other hand, this does support safe, consistent interaction between otherwise independent policy languages.

Finally, location-based policies and reasoning are well supported by DIFC tags – based on either network or physical location. For example, a confidentiality tag can be used to keep all patient data within the health service's physically secure network. This can enhance the multi-domain security architecture of Section 3, by supporting high-level specification of the interface between domains and preventing unauthorised event bridges between them. Similarly, physical security can be linked to the DIFC system, so that data displayed or entered at a hospital terminal is automatically linked to a DIFC confidentiality level, which is in turn bound to OASIS security policy rules. This allows the security of the event system to extend beyond the event interface.

For example, a doctor might be allowed to view clinical data only on a trusted terminal, e.g. inside a hospital, by defining a secure-terminal role for this (backed by environmental constraints). This results in DIFC confidentiality labels as described above, and our earlier rule can be changed to:

$$\begin{aligned}&\textsf{treating-doctor}(\textsf{\textit{hospital-ID}}, \textsf{\textit{doctor-ID}}, \textsf{\textit{patient-ID}}),\\&\textsf{secure-terminal} \vdash \textsf{priv-clinical}(\textsf{\textit{patient-ID}})\end{aligned}$$

Event security reasoning scales well to large-scale distributed systems: our domain model ensures that event types are managed in a structured, domain-based way. Parametrised RBAC reduces the number of rules and the frequency of rule change. This lets policy authors and users reason about event security on the basis of relatively static, domain-local information. Where policy references roles from other domains, either external rules can be checked directly, or partial conclusions can be drawn, with open-ended external role dependencies. For example, in reasoning about the consultant referral rule, the consultant-doctor role could be assumed to be assigned correctly by the appropriate governing body (the General Medical Council in the UK), without needing complete knowledge of the rules governing this assignment. High-level interface policies can further simplify analysis, particularly in large organisations such as a health service, by allowing reasoning about long-lasting security guarantees, independently of more dynamic local rules.

Thus end-to-end DIFC security enhances the scope for reasoning about event security, enabling policy-based reasoning to be extended to include event publishers, subscribers, and the environmental context in which they operate.

6 Conclusions

The ability to reason about event security depends on the underlying security architecture. In this chapter we have outlined our approach, which integrates strong, policy-based security into event-based systems. OASIS Role-Based Access Control allows simple administration and rich distributed security policy, with parametrised roles and appointment certificates. OASIS secures not only individual events, but also the event type system, supporting evolving rules and event structures for long-lived systems and data. We have shown how to use this in developing multi-domain systems, allowing security policy to reflect complex organisational structures and interconnections.

The security of event systems is enhanced by adopting fine-grained Distributed Information Flow Control restrictions on all event data. Extending these restrictions to event publishers and subscribers allows robust end-to-end security, which is essential for protecting long-term confidential data such as healthcare records. By translating OASIS policy enforcement into DIFC restrictions, we support both policy-based reasoning about event security, and the extension of this reasoning framework beyond the event processing middleware's boundaries.

In this way, event security reasoning can play a significant part in system-wide security audit and information governance, and can protect sensitive data in large-scale distributed systems.

Acknowledgement. We thank Jem Rashbass and Jon Crowcroft for their comments on this chapter. This work has been supported by grants from the UK Engineering and Physical Sciences Research Council (EPSRC) since 2000, including OASIS (GR/M75686), Contract-driven Systems (GR/S94919), EDSAC21 (GR/T28164), CareGrid (EP/C53719X) and, most recently, SmartFlow—Extendable Event-Based Middleware (EP/F042469 & EP/F044216; `http://www.smartflow.org/`).

References

1. Vargas, L., Bacon, J., Moody, K.: Integrating Databases with Publish/Subscribe. In: Proceedings of the 4th International Workshop in Distributed Event-Based Systems (DEBS 2005), June 2005, pp. 392–397. IEEE Press, Los Alamitos (2005)
2. Singh, J., Vargas, L., Bacon, J., Moody, K.: Policy-based information sharing in publish/subscribe middleware. In: IEEE Workshop on Policies for Distributed Systems and Networks (Policy 2008), IBM Palisades, New York, pp. 137–144. IEEE Press, Los Alamitos (2008)
3. Singh, J., Vargas, L., Bacon, J.: A Model for Controlling Data Flow in Distributed Healthcare Environments. In: Pervasive Health 2008: Second International Conference on Pervasive Computing Technologies for Healthcare, Tampere, Finland, January 2008, pp. 188–191. IEEE Press, Los Alamitos (2008)
4. Bacon, J., Moody, K., Yao, W.: Access control and trust in the use of widely distributed services. In: Liu, H. (ed.) Middleware 2001. LNCS, vol. 2218, pp. 295–310. Springer, Heidelberg (2001)
5. Bacon, J., Eyers, D.M., Moody, K., Pesonen, L.I.W.: Securing publish/Subscribe for multi-domain systems. In: Alonso, G. (ed.) Middleware 2005. LNCS, vol. 3790, pp. 1–20. Springer, Heidelberg (2005)

6. Bacon, J., Eyers, D.M., Singh, J., Shand, B., Migliavacca, M., Pietzuch, P.: Security in multi-domain event-based systems. IT - Information Technology 51(5), 277–284 (2009), doi:10.1524/itit.2009.0552
7. Eugster, P.T., Felber, P.A., Guerraoui, R., Kermarrec, A.M.: The many faces of publish/subscribe. ACM Computing Surveys 35(2), 114–131 (2003)
8. Pietzuch, P.R., Bacon, J.M.: Hermes: A distributed event-based middleware architecture. In: 1st International Workshop on Distributed Event-Based Systems (DEBS 2002), Vienna, Austria. ICDCS, pp. 611–618. IEEE Press, Los Alamitos (2002)
9. Pietzuch, P.R., Bacon, J.M.: Peer-to-peer overlay broker networks in an event-based middleware. In: 2nd International Workshop on Distributed Event-Based Systems (DEBS 2003). ICDCS, pp. 1–8. ACM SIGMOD, New York (2003)
10. Bacon, J., Eyers, D.M., Singh, J., Pietzuch, P.R.: Access control in publish/subscribe systems. In: Proceedings of the Second International Conference on Distributed Event-Based systems (DEBS 2008), pp. 23–34. ACM, New York (2008)
11. Sandhu, R., Coyne, E., Feinstein, H.L., Youman, C.E.: Role-based access control models. IEEE Computer 29(2), 38–47 (1996)
12. Bacon, J., Moody, K., Yao, W.: A model of OASIS role-based access control and its support for active security. ACM Transactions on Information and System Security (TISSEC) 5(4), 492–540 (2002)
13. Pietzuch, P., Eyers, D., Kounev, S., Shand, B.: Towards a Common API for Publish/Subscribe. In: Proceedings of the Inaugural Conference on Distributed Event-Based Systems (DEBS 2007), pp. 152–157. ACM Press, New York (2007) (short paper)
14. Pesonen, L.I.W., Bacon, J.: Secure Event Types in Content-Based, Multi-domain Publish/Subscribe Systems. In: SEM 2005: Proceedings of the 5th International Workshop on Software Engineering and Middleware, Lisbon, Portugal, September 2005, pp. 98–105. ACM Press, New York (2005)
15. Pesonen, L.I., Eyers, D.M., Bacon, J.: Access control in decentralised publish/subscribe systems. Journal of Networks 2(2), 57–67 (2007)
16. Pesonen, L.I.W., Eyers, D.M., Bacon, J.: Encryption-Enforced Access Control in Dynamic Multi-Domain Publish/Subscribe Networks. In: Proceedings of the Inaugural Conference on Distributed Event-Based Systems (DEBS 2007), June 2007, pp. 104–115. ACM Press, New York (2007)
17. Dierks, T., Allen, C.: The TLS protocol version 1.0. IETF RFC 2246 (January 1999)
18. Bell, D.E., La Padula, L.J.: Secure computer systems: Mathematical foundations and model. Technical Report M74-244. The MITRE Corp., Bedford, MA (May 1973)
19. Myers, A., Liskov, B.: Protecting privacy using the decentralized label model. ACM Transactions on Software Engineering and Methodology 9(4), 410–442 (2000)
20. Krohn, M., Yip, A., Brodsky, M., Cliffer, N., Kaashoek, M.F., Kohler, E., Morris, R.: Information flow control for standard OS abstractions. In: Proceedings of Twenty-First ACM SIGOPS Symposium on Operating Systems Principles (SOSP 2007), pp. 321–334. ACM, New York (2007)
21. Miglivacca, M., Papagiannis, I., Eyers, D., Shand, B., Bacon, J., Pietzuch, P.: High-performance event processing with information security. In: USENIX Annual Technical Conference, Boston, MA, USA, pp. 1–15 (2010)