Event-Processing Middleware with Information Flow Control

David Eyers, David.Eyers@cl.cam.ac.uk
Ben Roberts, bgr25@cam.ac.uk
Jean Bacon, Jean.Bacon@cl.cam.ac.uk
University of Cambridge
Matteo Migliavacca, migliava@doc.ic.ac.uk
Ioannis Papagiannis, ip108@doc.ic.ac.uk
Peter Pietzuch, prp@doc.ic.ac.uk
Imperial College London
Brian Shand, Brian.Shand@cbcu.nhs.uk
NHS

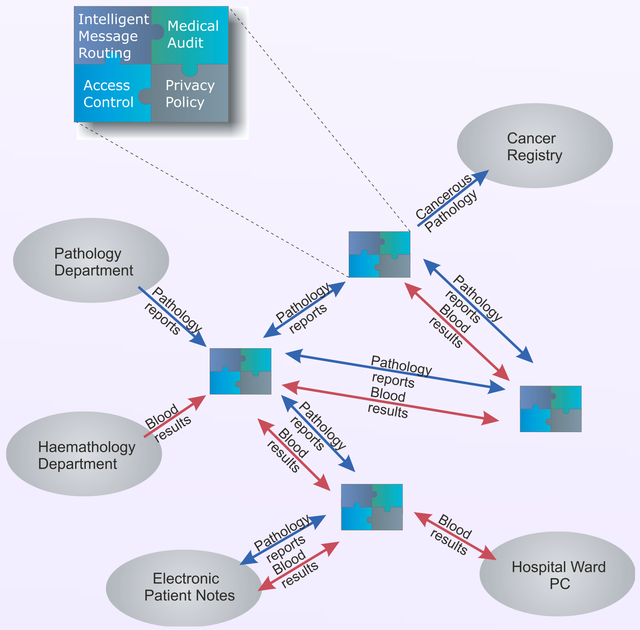
Background

Motivation

- Event processing systems have seen a recent surge in popularity and applicability
- Existing event-based systems have not focused on security issues
- Large-scale, multi-domain applications will require security
- Healthcare, e-Government and e-Business, are applicable domains

Example

- The diagram here shows an event-based healthcare system
- There is a strong case for collaboration between these organisations ...
- ... however strict access control policy must be maintained
- Policy enforcement that controls the information flow between components allows for overall system security properties to be verified



Securing data using Information Flow Control

- Information Flow Control (IFC) assigns labels to data, and to processes
- Processes can read or write data, only if a function of the current labels is satisfied
- Both data and processes can accumulate specific types of tainting
- Taint tracking allows containment of data until, for example, it is declassified

- Operating System IFC research includes Flume [1], HiStar [2], and AsbestosOS [3]. JIF [4] operates at programming language level, and DStar [5] in decentralised IFC.

DIFC is an attractive paradigm to apply in event-based systems, since events, and event sub-parts, can carry protection labels applied by different parts of a system.

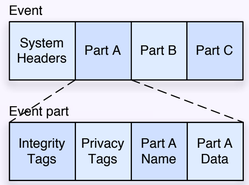
Design

Isolation at Middleware Granularity

- The host for event processing at each site is a SmartFlow engine that provides:
  - enforcement of the security of event data
  - publish/subscribe, intra-node communication services
  - application life-cycle management
  - global naming services

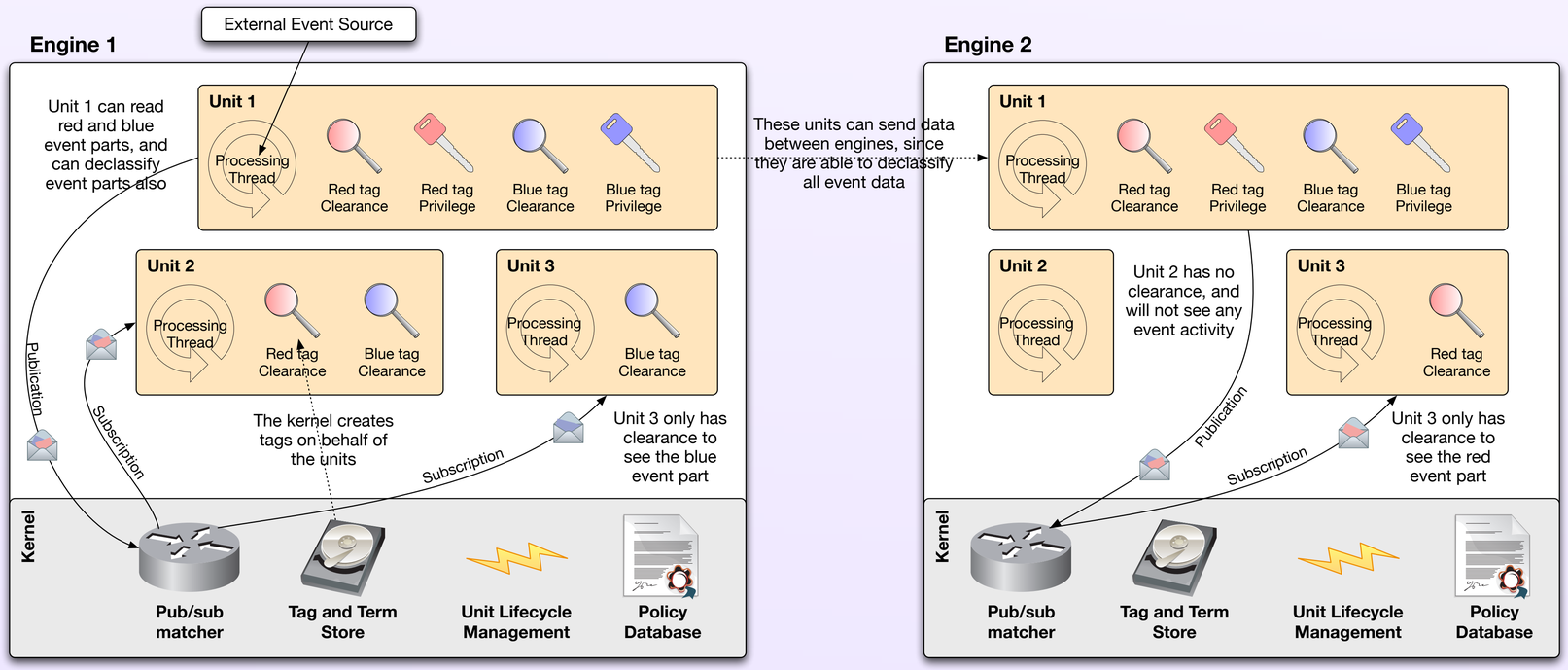
- Event processing units are hosted by each engine.
- Units are programmed by the parties that want to share SmartFlow infrastructure.
- Units subscribe to events of interest, and may annotate events in transit, or emit their own events.

Anatomy of SmartFlow Events



- The engine passes events between units
- Each event is made up of a number of parts
- Event parts carry integrity and security tags
- Tag sets form labels that are used to effect IFC
- Terms are hierarchically-scoped names
- Terms allow distributed units to match tag semantics
- Engines record term/tag relationships if suitable digital certificates are available
- For details of the tag model, see the PrivateFlow demo [6].

SmartFlow suits environments in which different organisations process the same event streams, but have a requirement for mutual isolation.



Implementation

Object Isolation in Modern Programming Languages

- SmartFlow engines' units will be grouped into isolates that communicate frequently
- However, IFC requires that unit code should not establish storage channels
- Isolation support in the programming language runtime will help effect containment
- Ideally, the runtime system of the language in which units are implemented should be modified as little as possible

Java Isolates

- Problem areas: native methods, static fields, and dynamic dispatch
- Past research efforts in Java Virtual Machine isolation include:
  - I-JVM: duplicate all static fields per isolate. What about native methods?
  - Sun's Barcelona Project: plug potential holes in the Java libraries manually

Our results so far...

- Java runtime library shrinking: only includes necessary reachable classes
- Classification of common coding patterns that are safe, but fail static analysis
- Use of aspect-oriented weaving tools to intercept call paths that have not been whitelisted by to static analysis or manual examination
- Investigating just-in-time application of I-JVM's static field techniques
- Appreciation that the Java type system, particularly type-erasure, makes makes some analyses more difficult than if using the .NET CLR.

References:
[1] M. Krohn, A. Yip, et al. Information flow control for standard OS abstractions. In *SOSP '07: Proceedings*, pp. 321–334, New York, NY, USA, 2007. ACM.
[2] N. Zeldovich and S. Boyd-Wickizer, et al. Making information flow explicit in HiStar. In *OSDI '06: Proceedings*, Seattle, WA, USA, 2006.
[3] S. Vandebogart, P. Efstathopoulos, et al. Labels and event processes in the Asbestos operating system. *ACM Trans. Comput. Syst.*, 25(4):11, 2007.
[4] A. Myers and B. Liskov. Protecting privacy using the decentralized label model. *ACM Trans. Softw. Eng. Methodol.*, 9(4):410–442, 2000.
[5] N. Zeldovich and S. Boyd-Wickizer, et al. Securing distributed systems with information flow control. In *NSDI'08: Proceedings*, pp. 293–308 San Francisco, CA, USA, 2008.
[6] I. Papagiannis, M. Migliavacca, et al. PrivateFlow: Decentralised Information Flow Control in Event Based Middleware. *Demonstration at DEBS'09*. See http://platypus.doc.ic.ac.uk/research/node/34