Enforcing End-to-end Application Security in the Cloud

Jean Bacon†, David Evans†, David M. Eyers†, Matteo Migliavacca‡, Peter Pietzuch‡, and Brian Shand∗

†University of Cambridge, UK {first.surname}@cl.cam.ac.uk
‡Imperial College London, UK {migliava, prp}@doc.ic.ac.uk
∗CBCU/ECRIC, National Health Service, UK brian.shand@cbcuj.nhs.uk

Middleware 2010
Motivation

• Cloud computing is all the rage
  – Steve Ballmer’s talk of “Hockey stick cloud growth”
  – Rackspace doubled cloud customers over 2009: 50,000 increase
  – Success stories: elasticity for on-line retail and services

• However security concerns in the cloud will be challenging
  – T-Mobile Sidekick, GMail outages, (older) Amazon S3 failures, ...
  – Availability and persistence issues. Responsibility for data recovery?

• Splitting responsibility: Legal due diligence?
  – Make application needs less abstract...
BI: Application-Level Virtualisation

- Server virtualisation: Isolatable, portable VMs
- ALV: Isolatable, portable, distributed application instances
  - Not just OS-level process or user-level isolation
  - Match the overall security needs of applications

- Cloud clients: explicit about application security needs
- Providers: provide basis for building secure applications

- End-to-end security as opposed to boundary protection
  - Data-linked guarantees on confidentiality, integrity and availability
Evolution from boundary security...
... into end-to-end security
Talk outline

- Enabling technologies
  - Clouds, IFC, Asynchronous messaging, RBAC & distributed policy
- Application-level virtualisation
- Case studies
  - Pairs trading
  - Healthcare scenarios
- Outstanding research issues
- Conclusion
Different cloud provider types

- **IaaS**: clients rent and manage VMs
  - EC2 (Amazon), Rackspace, and Nimbus

- **PaaS**: develop over restricted PL or services
  - Google’s App Engine and Microsoft’s Azure

- **SaaS**: Cloud hosts given application
  - Highest level of abstraction: clients just configure
  - Salesforce

- **Goal**: secure IaaS expressiveness with SaaS convenience
Information Flow Control (IFC)

- Strong guarantees over: data confidentiality and integrity
  - Labels tag data and processes: facilitates flow constraints
  - Data can only flow to processes with compatible labels
  - Data can only be released that is compatible with label of process
  - Privileges for declassification and endorsement are needed

- DIFC: decentralised, dynamic control of label definitions

<table>
<thead>
<tr>
<th>Event</th>
<th>name</th>
<th>data</th>
<th>integrity tags</th>
<th>confidentiality tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>bid</td>
<td>{i-trader-77}</td>
<td>∅</td>
<td></td>
</tr>
<tr>
<td>body</td>
<td>...</td>
<td>{i-trader-77}</td>
<td>{dark-pool}</td>
<td></td>
</tr>
<tr>
<td>trader_id</td>
<td>trader-77</td>
<td>{i-trader-77}</td>
<td>{dark-pool,s-trader-77}</td>
<td></td>
</tr>
</tbody>
</table>
Asynchronous messaging with IFC

• Non-trivial cloud apps are large-scale distributed systems
  – Decoupling is often employed for robustness
  – Even for cloud services that are stateless message passing is useful
  – Robustness during failures and reconfiguration

• Treat all communicated messages as multi-part structures
  – Independent IFC labelling of parts
  – Data-oriented access control: amenable to middleware protection
  – Parallel processing: each part has its own data and security label.
Role-based Access Control (RBAC)

- Effective technology for large scale privilege management
  - Decouple users from privileges by using roles
  - See ANSI standards

- Parameterised RBAC: needs inference engine

- Applications with multiple administrative domains: need Service Level Agreements (SLAs)
  - Distributed RBAC with DIFC
Application-level virtualisation

- Much cloud security is at boundary of infrastructure
- ALV: end-to-end, data-linked security using DIFC

- Application specifies explicit data containment policy
  - Useful authority requires coverage of net, IPC, & disk data paths
  - Secrecy, privacy, integrity, availability...
  - Encode legal-level concerns: responsibilities for data management

- Middleware abstractions should make code more portable
  - Across and between different cloud providers
Application-level virtualisation

- Cloud provider must explicitly specify interactions with the Trusted Computing Base (TCB)
  - e.g. IFC and middleware-managed containers
  - Extend to persistence mechanisms

- Provider can monitor, audit, enforce, check compliance

- Cloud provider can reuse security mechanisms to isolate different clients' applications
Case study: pairs trading

- Pairs trading is a simple algorithm for stock trading
  - Assume that stocks in a related market will move in a related way
  - Look for divergence and bet on divergence soon reducing

- Modern exchanges are electronic, event-based systems:
  - Orders are quick transactions, pub/sub for dissemination of stocks

- Competition in the market makes it very latency sensitive
  - Co-host trading algorithms on server co-located with exchange

- Large brokers can offer “dark pools”: avoids exchange
Case study: pairs trading

- Used IFC to support safely co-located pairs trading
  - Protection of market data integrity
  - Strict control of interactions between clients’ investment strategies
  - Ensure confidentiality of orders in “dark pool” trading
    - Bulk-move equities without revealing trader’s identity
    - Allow auditing by regulatory authorities on completed transactions

- Treating stock-exchange hosting as a “cloud” (in effect)

- Application-level virtualisation protects traders’ strategies while efficiently utilising infrastructure
‘DEFCon’ pairs trading event model

KEY:

- Unit that can declassify all of its state
- Unit that cannot declassify any of its state
- Unit that has some state that it cannot declassify
- Event, Tagged by Trader 1
- Event, Tagged by Trader 2
‘DEFCon’ pairs trading event model

**KEY:**
- Unit that can declassify all of its state
- Unit that cannot declassify any of its state
- Unit that has some state that it cannot declassify
- Event

**Diagram Description:**
- **Stock Exchange**
- **Pair Monitor**
  - **Tagged by Trader 1**
  - **Tagged by Trader 2**
- **Monitor**
  - **Symbol 1:** GOOG
  - **Symbol 2:** MSFT
  - **Mean:** ...
  - **SD:** ...
- **Trader 1** owns $t_1$
- **Trader 2** owns $t_2$
‘DEFCon’ pairs trading event model

Stock Exchange

Pair Monitor
- secrecy: $t_1$
- integrity: $s$

Monitor
- symbol1: GOOG
- symbol2: MSFT
- mean: ...
- sd: ...

Pair Monitor
- secrecy: $t_2$
- integrity: $s$

Monitor
- symbol1: GOOG
- symbol2: MSFT
- mean: ...
- sd: ...

Event

Trader 1
- owns $t_1$

Trader 2
- owns $t_2$

KEY:
- Unit that can declassify all of its state
- Unit that cannot declassify any of its state
- Unit that has some state that it cannot declassify
- Event Tagged by Trader 1
- Tagged by Trader 2
‘DEFCon’ pairs trading event model

Stock Exchange

Event

Pair Monitor
secrecy: \( t_1 \)
integrity: \( s \)

Match
buy: GOOG
sell: MSFT
mean: 1.92

Monitor
symbol1: GOOG
symbol2: MSFT
mean: ...
sd: ...

Monitor
symbol1: GOOG
symbol2: MSFT
mean: ...
sd: ...

Match
buy: MSFT
sell: GOOG
mean: 1.92

Stock Exchange

Tick
MSFT
1230$
12:22:13

Pair Monitor
secrecy: \( t_2 \)
integrity: \( s \)

Trader 1
owns \( t_1 \)

Trader 2
owns \( t_2 \)

KEY:

- Unit that can declassify all of its state
- Unit that cannot declassify any of its state
- Unit that has some state that it cannot declassify
- Event

Tagged by Trader 1
Tagged by Trader 2


DEFCon’ pairs trading event model

Stock Exchange

Tick
MSFT 1230$ 12:22:13

Pair Monitor
secrecy: $1$
integrity: $s$

Pair Monitor
secrecy: $t_2$
integrity: $s$

Monitor
symbol1: GOOG
symbol2: MSFT
mean: ...
sd: ...

Monitor
symbol1: GOOG
symbol2: MSFT
mean: ...
sd: ...

Match
buy: GOOG
sell: MSFT
mean: 1.92

Match
buy: MSFT
sell: GOOG
mean: 1.92

Trade 1

owns $t_1$

owns $t_2$

Bid
sell: MSFT
price: 1234$
name: Trader 1

Ask
buy: MSFT
price: 1234$
name: Trader 2

KEY:
- Unit that can declassify all of its state
- Unit that cannot declassify any of its state
- Unit that has some state that it cannot declassify
- Event

Tagged byTrader 1

Tagged byTrader 2
'DEFCon' pairs trading event model

**KEY:**
- Unit that can declassify all of its state
- Unit that cannot declassify any of its state
- Unit that has some state that it cannot declassify
- Event

**Stock Exchange**

**Pair Monitor**
- secrecy: $t_1$
- integrity: $s$

**Pair Monitor**
- secrecy: $t_2$
- integrity: $s$

**Monitor**
- symbol1: GOOG
- symbol2: MSFT
- mean: ...
- sd: ...

**Match**
- buy: GOOG
- sell: MSFT
- mean: 1.92

**Match**
- buy: MSFT
- sell: GOOG
- mean: 1.92

**Monitor**
- symbol1: GOOG
- symbol2: MSFT
- mean: ...
- sd: ...

**Broker**
- owns $b$

**Trader 1**
- owns $t_1$

**Trader 2**
- owns $t_2$

**Tick**
- MSFT
- 1230$
- 12:22:13

**Bid**
- sell: MSFT
- price: 1234$
- name: Trader 1

**Ask**
- buy: MSFT
- price: 1234$
- name: Trader 2
‘DEFCon’ pairs trading event model

KEY:
- Light blue: Unit that can declassify all of its state
- Light orange: Unit that cannot declassify any of its state
- Light brown: Unit that has some state that it cannot declassify
- Event: Tagged by Trader 1
- Tagged by Trader 2

Stock Exchange

Pair Monitor
- secrecy: \( t_1 \)
- integrity: \( s \)

Pair Monitor
- secrecy: \( t_2 \)
- integrity: \( s \)

Monitor
- symbol1: GOOG
- symbol2: MSFT
- mean: ...
- sd: ...

Match
- buy: GOOG
- sell: MSFT
- mean: 1.92

Match
- buy: MSFT
- sell: GOOG
- mean: 1.92

Monitor
- symbol1: GOOG
- symbol2: MSFT
- mean: ...
- sd: ...

Trader 1

Trader 2

Broker

Match
- buy: GOOG
- sell: MSFT
- mean: ...

Bid
- sell: MSFT
- price: 1234$
- name: Trader 1

Trade
- price: 1234$
- symbol: MSFT
- buyer: Trader 1
- seller: Trader 2

Ask
- buy: MSFT
- price: 1234$
- name: Trader 2

Tick
- MSFT
- 1230$
- 12:22:13

owns \( s \)
‘DEFCon’ pairs trading event model
‘DEFCon’ pairs trading event model

- **Stock Exchange**
  - **Tick**: MSFT 1230$ 12:22:13
  - **Pair Monitor** (secrecy: $t_1$, integrity: $s$)
  - **Pair Monitor** (secrecy: $t_2$, integrity: $s$)

- **Trades**
  - **Match**: buy: GOOG sell: MSFT mean: 1.92
  - **Monitor**: symbol1: GOOG symbol2: MSFT mean: ... sd: ...
  - **Monitor**: symbol1: GOOG symbol2: MSFT mean: ... sd: ...

- **Orders**
  - **Bid**: sell: MSFT price: 1234$ name: Trader 1
  - **Trade**: price: 1234$ symbol: MSFT buyer: Trader 1 seller: Trader 2
  - **Ask**: buy: MSFT price: 1234$ name: Trader 2

- **Events**
  - **Warning**: msg: Trading volume exceeded quota

- **Delegation**
  - **user**: Trader 2

**KEY:**
- Unit that can declassify all of its state
- Unit that cannot declassify any of its state
- Unit that has some state that it cannot declassify
- Event Tagged by Trader 1
- Event Tagged by Trader 2

**Tagged by Trader 1**
- Tagged by Trader 2
`DEFCon` pairs trading event model

**Key**:
- **White** box: Unit that can declassify all of its state.
- **Light gray** box: Unit that cannot declassify any of its state.
- **Dark gray** box: Unit that has some state that it cannot declassify.
- **Event** box: Tagged by Trader 1
- **Event** box: Tagged by Trader 2

**Diagram**:
- **Stock Exchange**
- **Pair Monitor**
  - secrecy: \( t_1 \)
  - integrity: \( s \)
- **Monitor**
  - symbol1: GOOG
  - symbol2: MSFT
  - mean: ...
  - sd: ...
- **Match**
  - buy: GOOG
  - sell: MSFT
  - mean: 1.92
- **Tick**
  - MSFT 1234$
  - 12:32:13
- **Tick**
  - MSFT 1230$
  - 12:22:13
- **Trader 1**
  - owns \( t_1 \)
- **Trader 2**
  - owns \( t_2 \)
- **Bid**
  - sell: MSFT
  - price: 1234$
  - name: Trader 1
- **Ask**
  - buy: MSFT
  - price: 1234$
  - symbol: MSFT
  - buyer: Trader 1
  - seller: Trader 2
- **Trade**
  - price: 1234$
  - symbol: MSFT
  - buyer: Trader 1
  - seller: Trader 2
- **Delegation**
  - user: Trader 2
- **Warning**
  - msg: Trading volume exceeded quota
  - owned by: Trader 2
- **Regulator**
  - owned by: Trader 2
Case study: UK NHS and ECRIC

• At a high level:
  – UK NHS involves many collaborating but separate organisations
  – Some organisations do similar work—aggregate!
  – The organisations’ software needs to handle access control: ALV

• E.g. ECRIC is a cancer registry
  – has legal privilege to collect patients’ records without consent

• Multi-Disciplinary Teams work in hospitals
  – Work with cancer patients
  – ECRIC currently not connected with them on-line
Case study: UK NHS and ECRIC

- Two separate potential clouds: ECRIC and the hospitals
Case study: UK NHS and ECRIC

• Explore programmer-friendly layers over DEFCon
  – Compile higher-level policy representations into IFC constraints
  – e.g. DPL language for expressing constraints over information flows
  – Couple RBAC policy with event types and thence DIFC

```prolog
\begin{align*}
  & \text{treating-doctor}( \text{referring-hospital-ID, referring-doctor-ID, patient-ID} ), \\
  & \text{consultant-doctor}( \text{hospital-ID, doctor-ID} ), \\
  & \text{consultant-referral}( \text{referring-doctor-ID, doctor-ID, patient-ID} ) \\
  & \vdash \text{treating-doctor}( \text{hospital-ID, doctor-ID, patient-ID} ) \\
  & \text{treating-doctor}( \text{hospital-ID, doctor-ID, patient-ID} ) \vdash \text{priv-clinical}( \text{patient-ID} ) \\
  & \text{clinical-researcher}( \text{study-ID, researcher-ID, patient-ID} ) \vdash \text{priv-clinical}( \text{patient-ID} ) 
\end{align*}
```
Open research challenges

• Cloud security mechanisms
  – different trusted code bases and their applicability
  – performance: can policy establishment avoid the critical path?
  – what’s the impact of enforcement in general?

• Policy specification in the cloud
  – programmer friendly
  – make it easy for programmer and for inter-domain agreement
  – RBAC relevance?
  – protecting policy itself: e.g. sensitive tags in labels
  – policy review and formal verification?
Conclusion

• Security in the cloud must be designed in from the start
  – need strong isolation of data, even during outsourced processing
  – isolate users of shared services as well as services

• Integrate strong end-to-end security mechanisms
  – Decentralised Information Flow Control
  – event-based communication for robust service interconnection
  – role-based policy specification including on processing and storage
  – explicit trusted computing base

• Our research to date demonstrates feasibility of subparts