SafeWeb

A Middleware for Securing Ruby-Based Web Applications

Petr Hosek
p.hosek@imperial.ac.uk **Imperial College London**

Matteo Migliavacca
migliava@doc.ic.ac.uk **Imperial College London**

Ioannis Papagiannis
ip108@doc.ic.ac.uk **Imperial College London**

David M. Eyers
dme@cs.otago.ac.nz **University of Otago**

David Evans
david.evans@cl.cam.ac.uk **University of Cambridge**

Brian Shand
brian.shand@cbcu.nhs.uk **ECRiC, NHS**

Jean Bacon
jean.bacon@cl.cam.ac.uk **University of Cambridge**

Peter Pietzuch
p.pietzuch@imperial.ac.uk **Imperial College London**

Data confidentiality

in **enterprise** web applications

Focus on applications processing sensitive data

- Requirements regarding compliance with legal policies

Problem of controlling all data flows

- Data protection across multiple layers at different granularities

Common threat model:

1. External environment is **hostile**
2. Application code is **not** explicitly **malicious**
3. Threats might be caused by **bugs** in implementation

Real-world case study

Provide web **portal** for accessing patient records

Make patient records accessible for review and feedback purposes

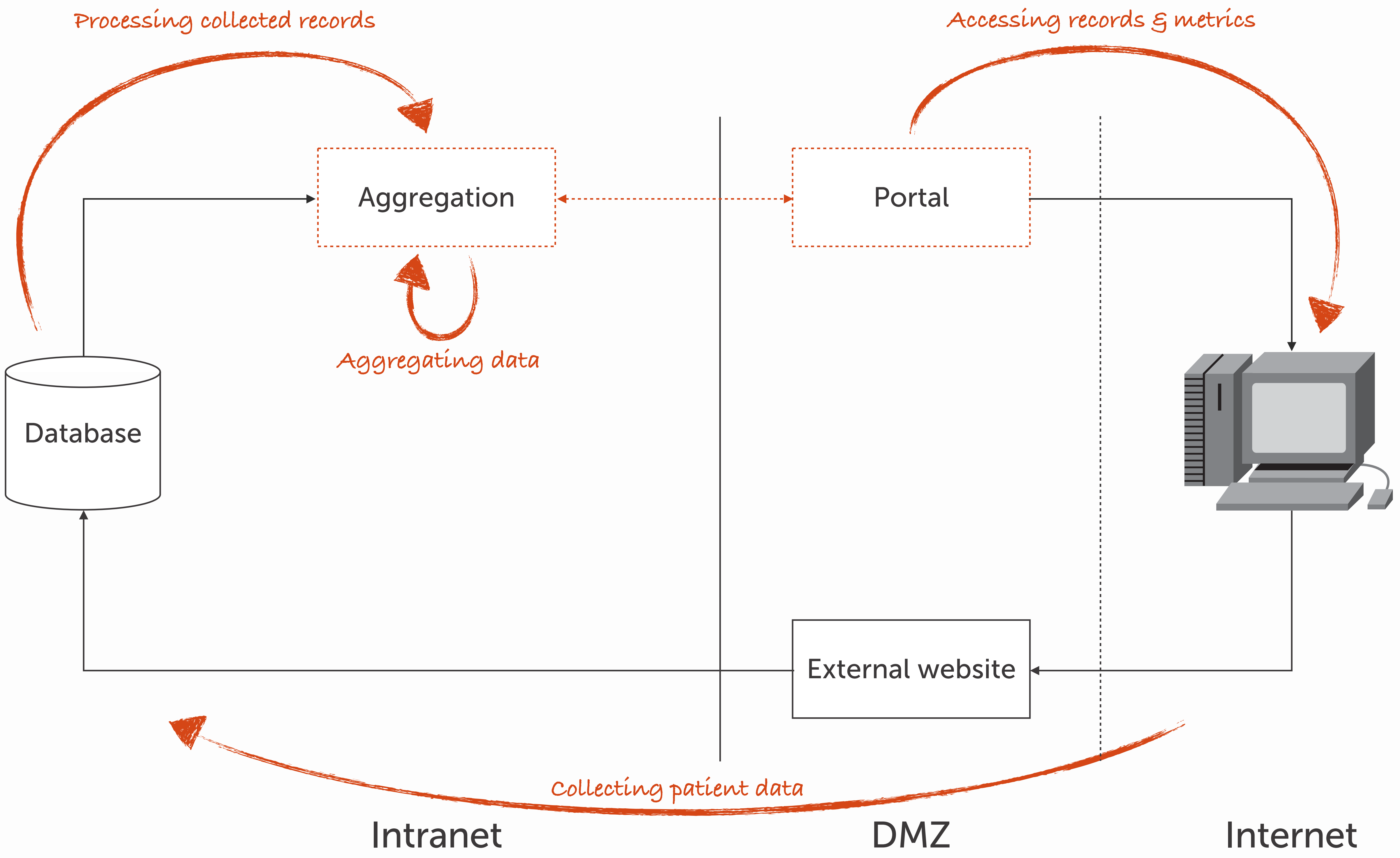
Strict security policy requirements:

1. Statistics & metric accessible to **all** staff
2. Patient details accessible **only** to patient treating staff

Current best practices are insufficient

Expensive and error-prone **source code auditing**

Limited exposure of collected data



Processing collected records

Accessing records & metrics

Aggregation

Portal

Aggregating data

Database

External website

Collecting patient data

Intranet

DMZ

Internet

1. Confidential data should be **protected end-to-end**
2. Access to confidential data by external users should be **static** & **one-way**

Contributions

SafeWeb middleware for **end-to-end** data protection

- Uses information flow control for data tracking
- Guarantees data confidentiality and integrity

Mechanisms for data tracking at different granularities

- Enforcement of data protection at event and variable levels
- Efficient implementation using Ruby dynamic programming features

Real-world evaluation in a healthcare environment

- Developed & deployed in collaboration with UK National Health Service (NHS)

OUTLINE

motivation **& contribution**

information flow **control**

SafeWeb **architecture**

label **propagation**

real-world **case study**

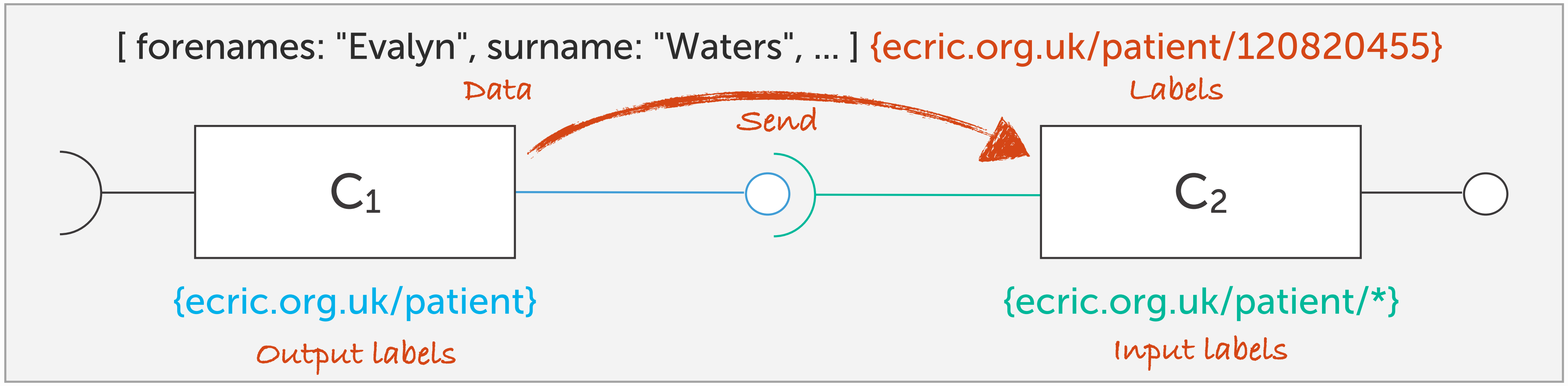
evaluation

conclusions

Information flow control

Protects the propagation of data

Attaching security labels to data and tracking their propagation



[forenames: "Evalyn", surname: "Waters", ...] {ecric.org.uk/patient/120820455}

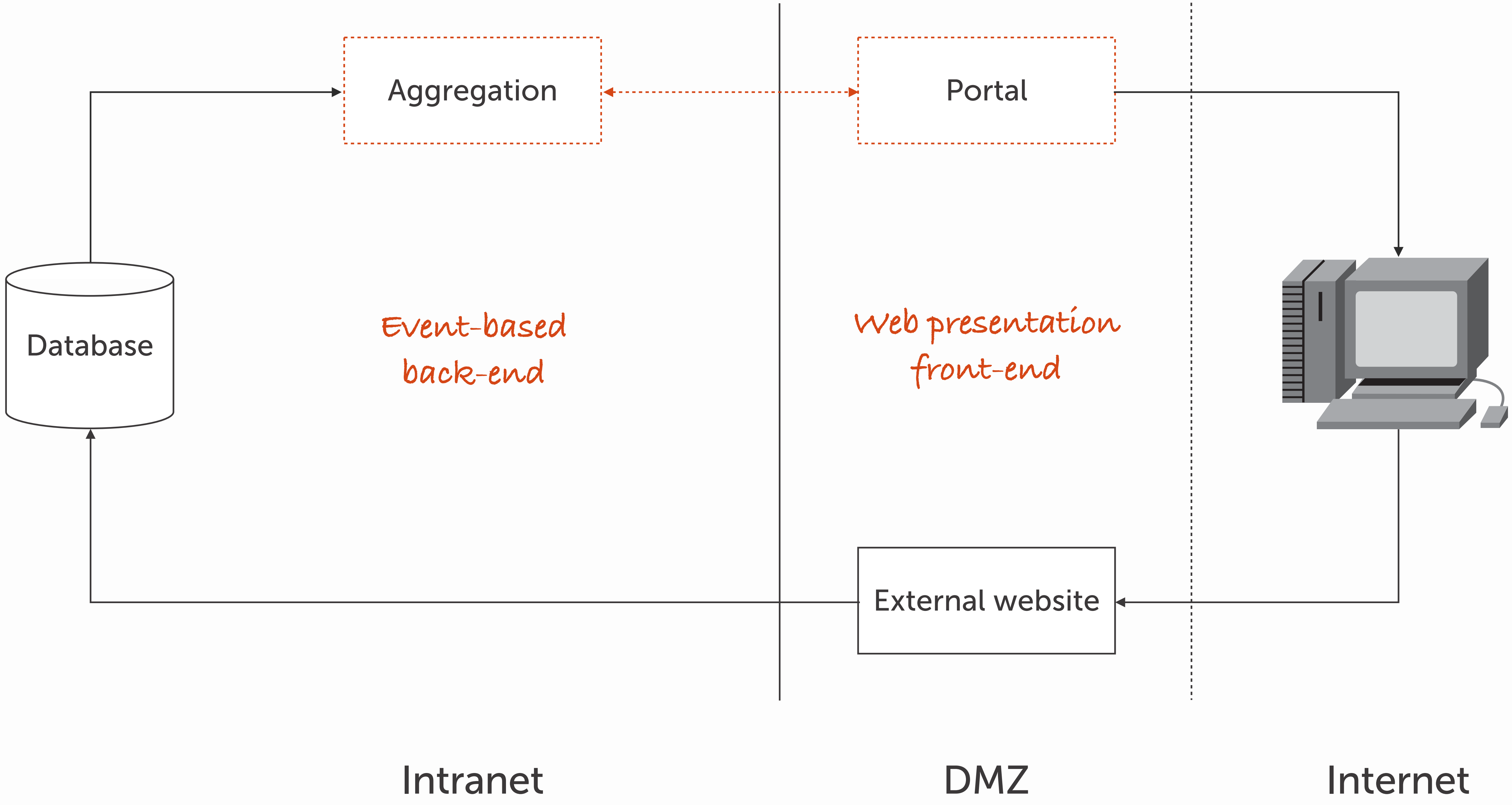
{ecric.org.uk/patient}

{ecric.org.uk/patient/*}

C_1 can output event iff {ecric.org.uk/patient} $\subseteq$ {ecric.org.uk/patient/120820455}

C_2 can input event iff {ecric.org.uk/patient/120820455} $\subseteq$ {ecric.org.uk/patient/*}

Bell, D., LaPadula, L.: Secure computer system: Unified exposition and Multics interpretation (1976)



Aggregation

Portal

Database

Event-based back-end

Web presentation front-end

External website

Intranet

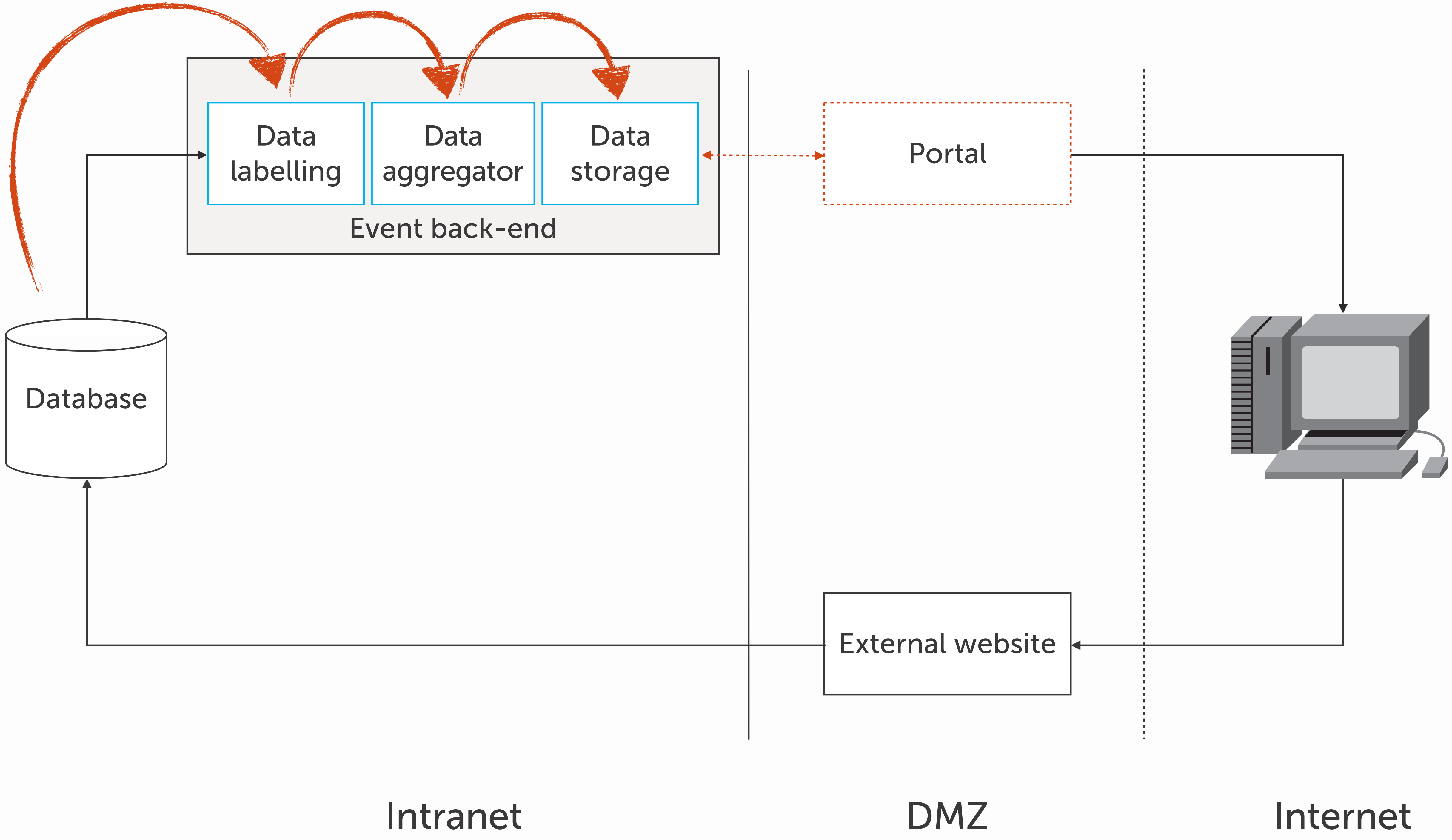
DMZ

Internet

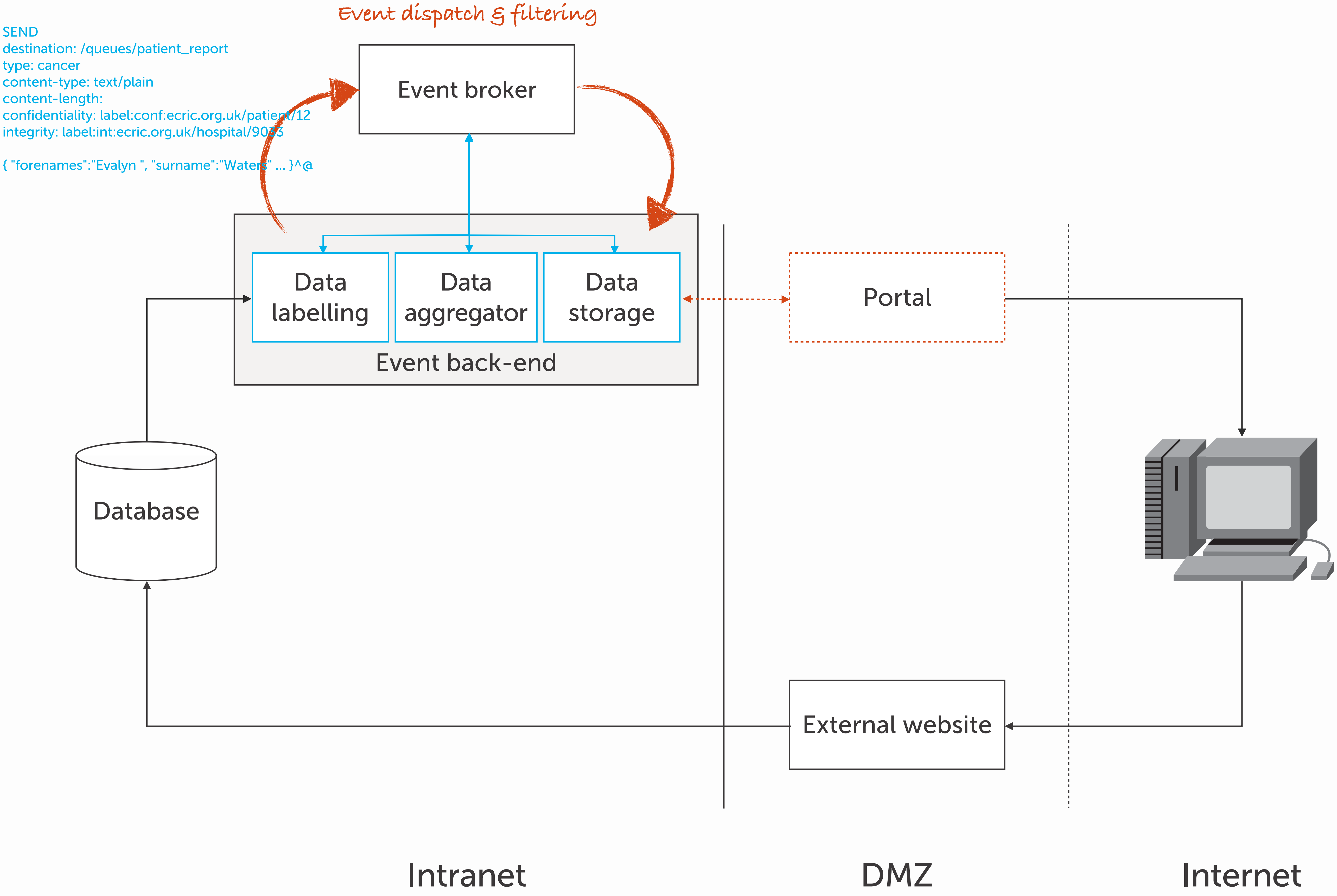
1. Confidential data should be **protected end-to-end**
2. Access to confidential data by external users should be **static** & **one-way**

```
SEND
destination: /queues/patient_report
type: cancer
content-type: text/plain
content-length:
confidentiality: label:conf:ecric.org.uk/patient/12
integrity: label:int:ecric.org.uk/hospital/9033

{ "forenames":"Evalyn ", "surname":"Waters" ... }^@
```

Event-based communication



1. Confidential data should be **protected end-to-end**
2. Access to confidential data by external users should be **static** & **one-way**



```
SEND
destination: /queues/patient_report
type: cancer
content-type: text/plain
content-length:
confidentiality: label:conf:ecric.org.uk/patient/12
integrity: label:int:ecric.org.uk/hospital/9033

{ "forenames":"Evalyn ", "surname":"Waters" ... }^@
```

1. Confidential data should be **protected end-to-end**
2. Access to confidential data by external users should be **static** & **one-way**

Event back-end

Support & control of unit execution

- Checking and tracking events security labels
- Prevent units from **disclosing** confidential data

Tracking the security labels at the level of **events**

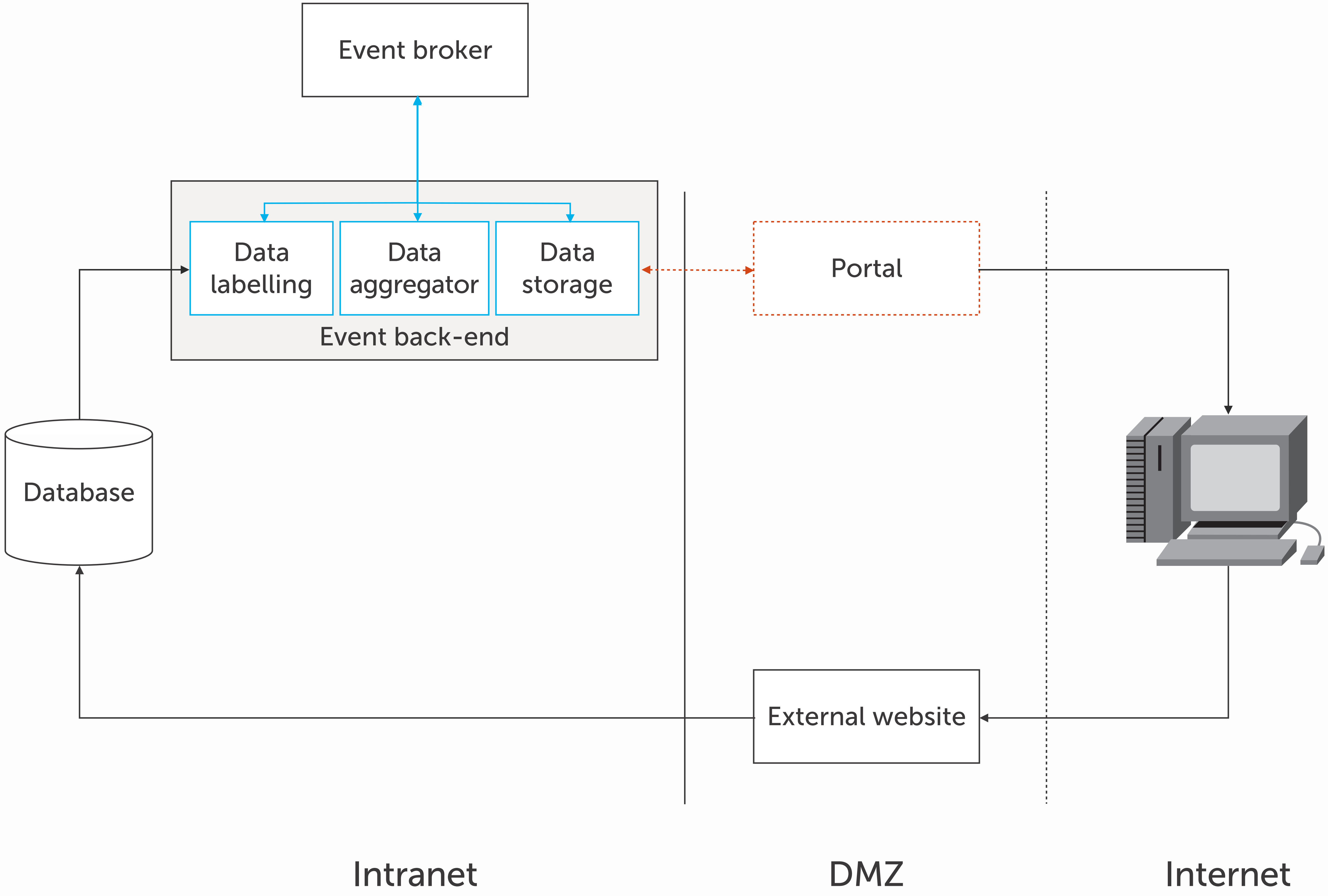
- Simple event data model using set of key-value attribute pairs and data payload
- STOMP-based event protocol extended with support for security labels

Enforces unit sandboxing & isolation

- Controlling the use of all I/O operations
- Preventing access to variables outside of local scope

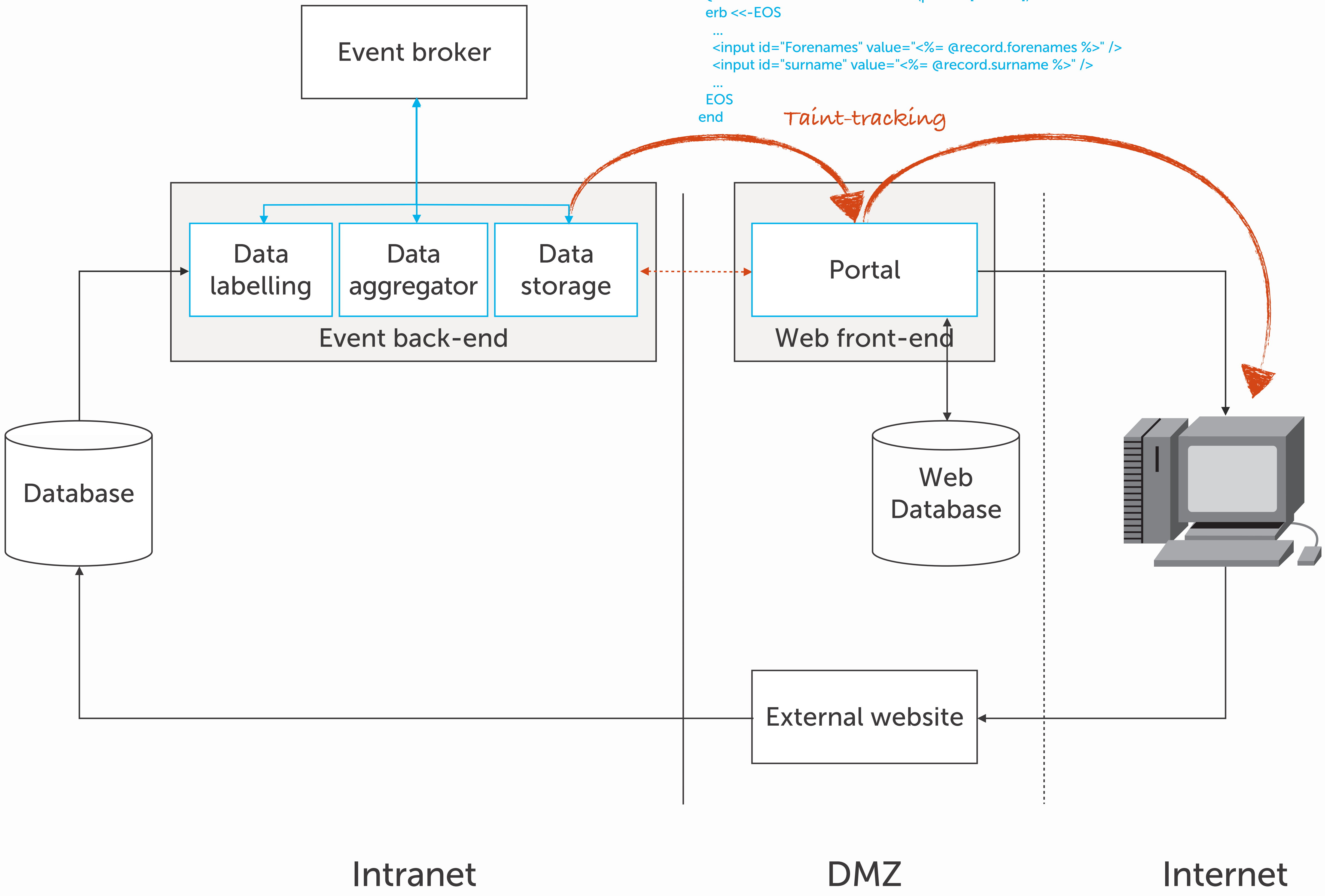
```
list = get 'patient_list' << event.patient
puts $patients << event.patient
publish '/daily_report', list, :add => ['label:conf:ecric.org.uk/patient_list']
```

Accessing global variable & I/O operation

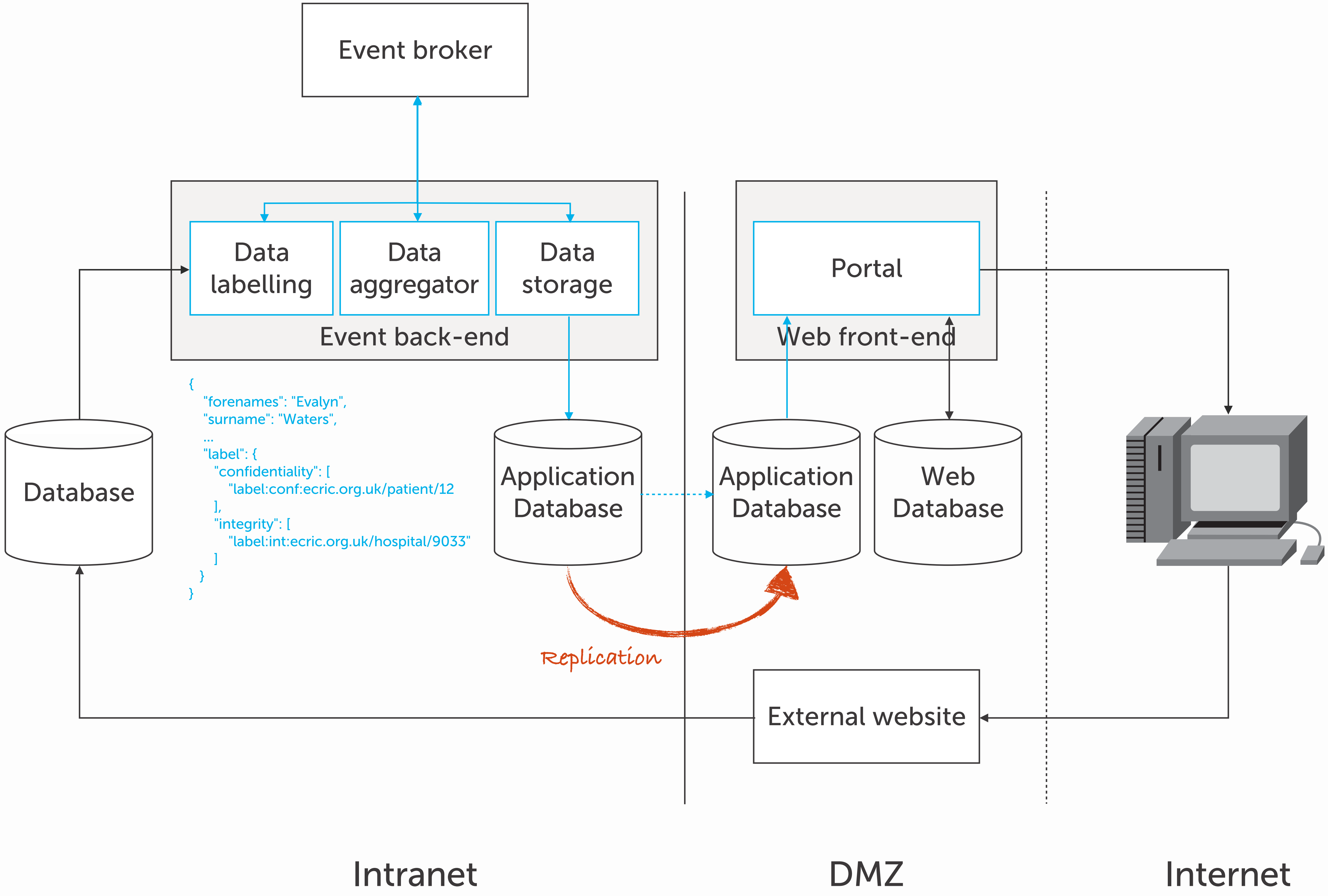


1. Confidential data should be **protected end-to-end**
2. Access to confidential data by external users should be **static** & **one-way**

```
get '/record/:record' do
  @record = SomersetRecord.find(params[:record])
  erb <<-EOS
    ...
    <input id="Forenames" value="<%= @record.forenames %>" />
    <input id="surname" value="<%= @record.surname %>" />
    ...
  EOS
end
```



1. Confidential data should be **protected end-to-end**
2. Access to confidential data by external users should be **static** & **one-way**



```
{
   "forenames": "Evalyn",
   "surname": "Waters",
   ...
   "label": {
     "confidentiality": [
        "label:conf:ecric.org.uk/patient/12
     ],
     "integrity": [
        "label:int:ecric.org.uk/hospital/9033"
     ]
   }
}
```

1. Confidential data should be **protected end-to-end**
2. Access to confidential data by external users should be **static** & **one-way**

Web front-end

Presents results from back-end to users

Sinatra-based web framework with traditional database-driven architecture

Enforces data flow control

Using data security labels assigned by data processing units

Taint tracking at the level of **variables**

Associating security labels with individual variables

Checking labels on HTTP response

```
@name = @patient.forenames + " " + @patient.surname
@name.add_tags! ['label:ecric.org.uk/patient/fullname']
erb "<input id='name' value='<%= @name >' />"
```

Taint propagation

Label propagation

without runtime or code modification

SafeWeb uses **unmodified** Ruby runtime

- Avoiding unnecessary code transformations
- Simplified deployment & maintenance

Sandboxing & isolation through Ruby **$SAFE** levels

- Simple taint-tracking mechanism with set of **predefined** levels

Exploiting Ruby's meta-programming features

- Using Rubinius **meta-circular** Ruby VM implementation

A. Yip et al. Improving Application Security with Data Flow Assertions
S. Naira et al. A Virtual Machine Based Information Flow Control System for Policy Enforcement

```
id = measurement[:id]
report = Report.new(measurement)

subscribe "/queues/measurements/#{id}/cancer_cases" do |event|
   report.append event
end

subscribe '/queues/measurements/release' do |timestamp|
   report.mark timestamp
   publish '/queues/reports', report
end
```

Snippet of patient record aggregation logic

Part of data aggregator unit

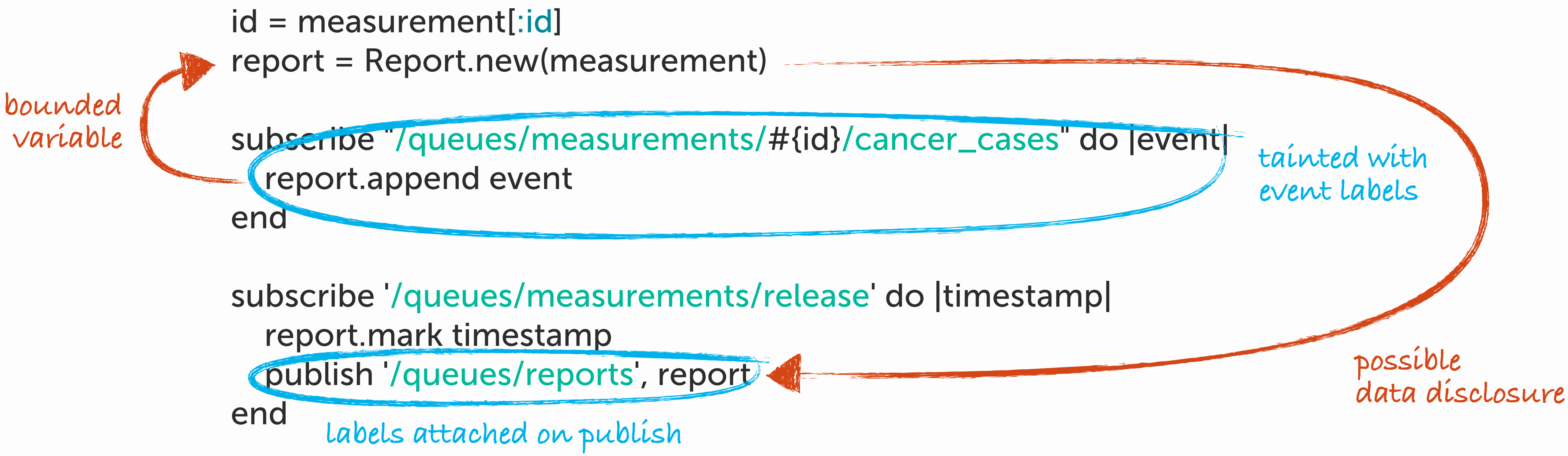
```
id = measurement[:id]
report = Report.new(measurement)

subscribe "/queues/measurements/#{id}/cancer_cases" do |event|
  report.append event
end

subscribe '/queues/measurements/release' do |timestamp|
  report.mark timestamp
  publish '/queues/reports', report
end
```

tainted with
event labels

labels attached on publish

Snippet of patient record aggregation logic

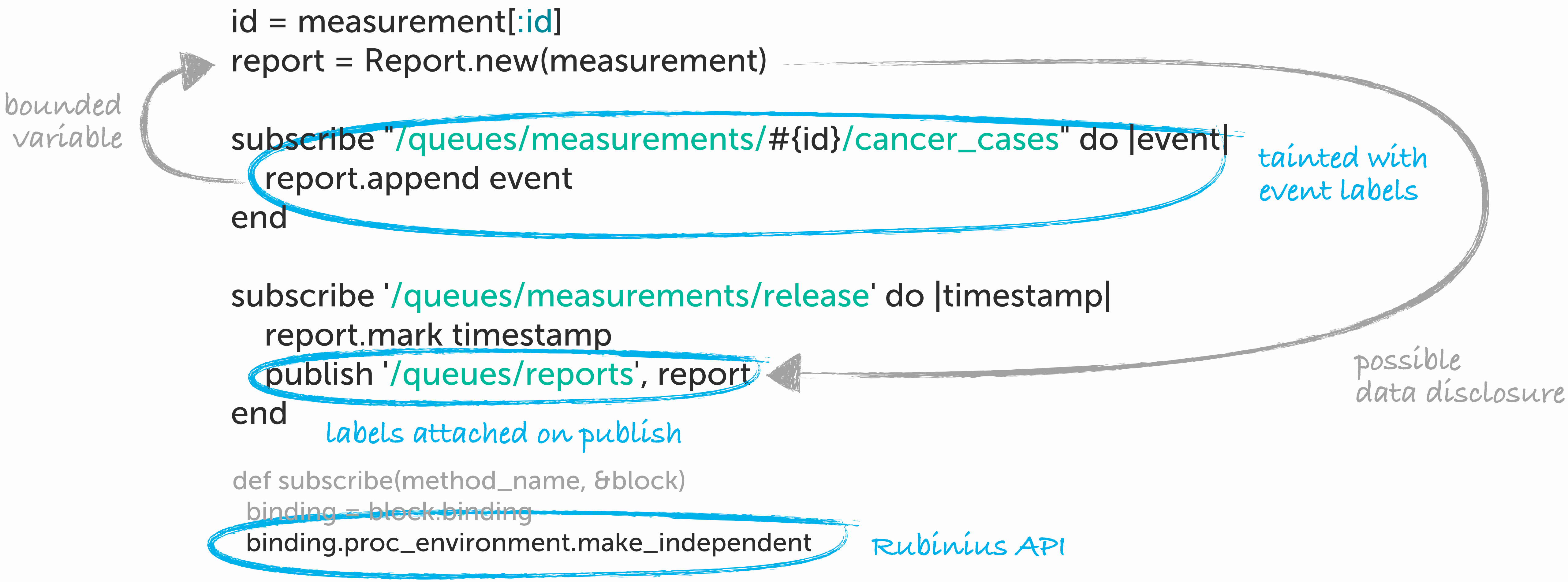
Part of data aggregator unit



```
id = measurement[:id]
report = Report.new(measurement)

subscribe "/queues/measurements/#{id}/cancer_cases" do |event|
   report.append event
end

subscribe '/queues/measurements/release' do |timestamp|
   report.mark timestamp
   publish '/queues/reports', report
end
```

Snippet of patient record aggregation logic

Part of data aggregator unit



```
id = measurement[:id]
report = Report.new(measurement)

subscribe "/queues/measurements/#{id}/cancer_cases" do |event|
  report.append event
end

subscribe '/queues/measurements/release' do |timestamp|
  report.mark timestamp
  publish '/queues/reports', report
end
```

```
def subscribe(method_name, &block)
  ~~binding = block.binding~~
  binding.proc_environment.make_independent

  define_method(method_name, &block)
  unbound_method = instance_method method_name
  @subscriptions[method_name] = proc {
    unbound_method.bind(self).call
  }
end
```

Snippet of patient record aggregation logic

Part of data aggregator unit

```
id = measurement[:id]
report = Report.new(measurement)
set id, report

subscribe "/queues/measurements/#{id}/cancer_cases" do |event|
  report = get id
  report.append event
  set id, report
end
```

tainted with
event labels

```
subscribe '/queues/measurements/release' do |timestamp|
  report = get id
  report.mark timestamp
  publish '/queues/reports', report
end
```

tainted with stored data labels

labels attached on publish

Snippet of patient record aggregation logic

Part of data aggregator unit

Real-world case study

in a healthcare organisation

Eastern Cancer Registry and Information Centre (ECRiC)

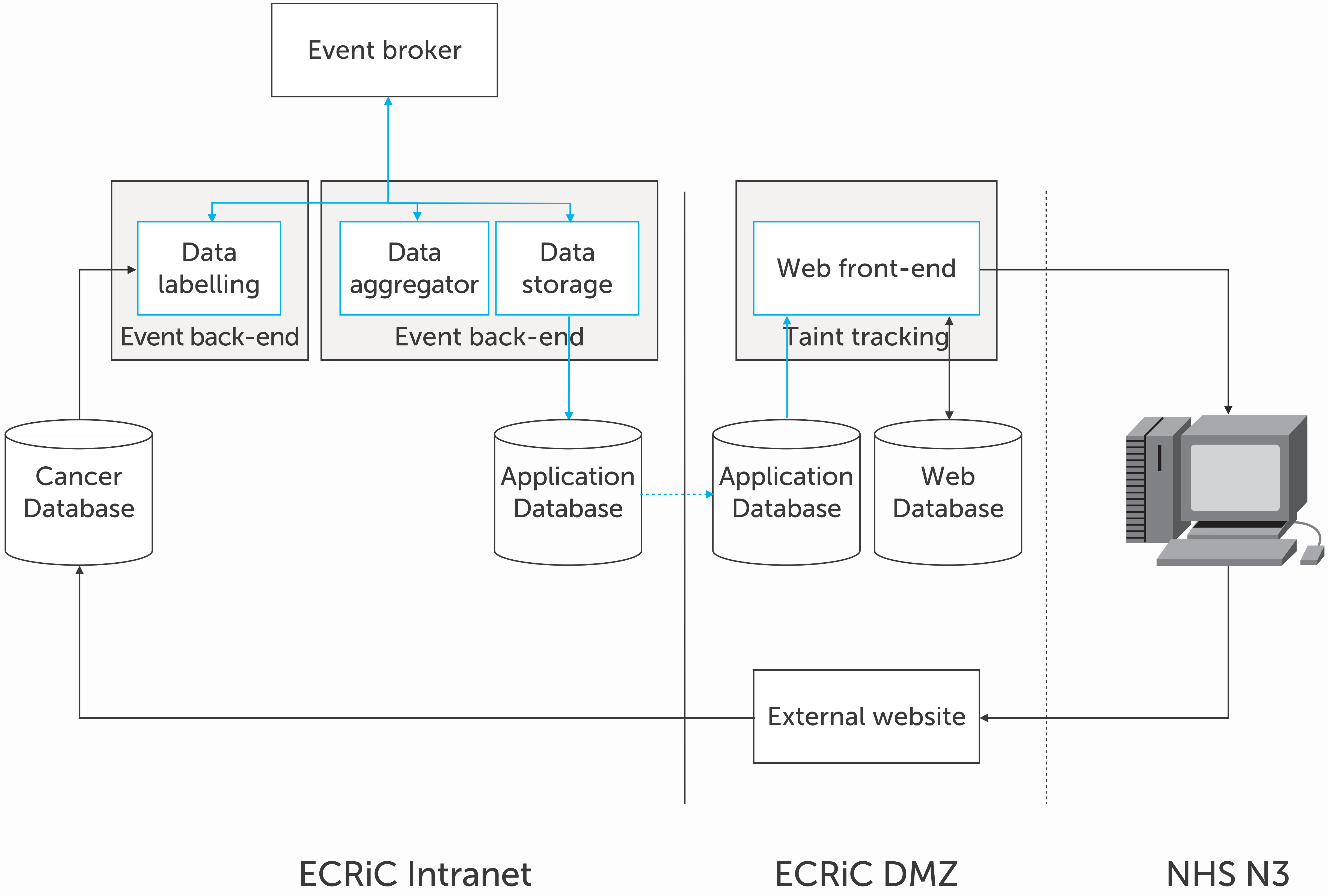
- Collects histories of cancer cases in the East of England

Aims to provide patient records feedback application

- Following the existing data protection & security requirements

Compatibility with existing production environment

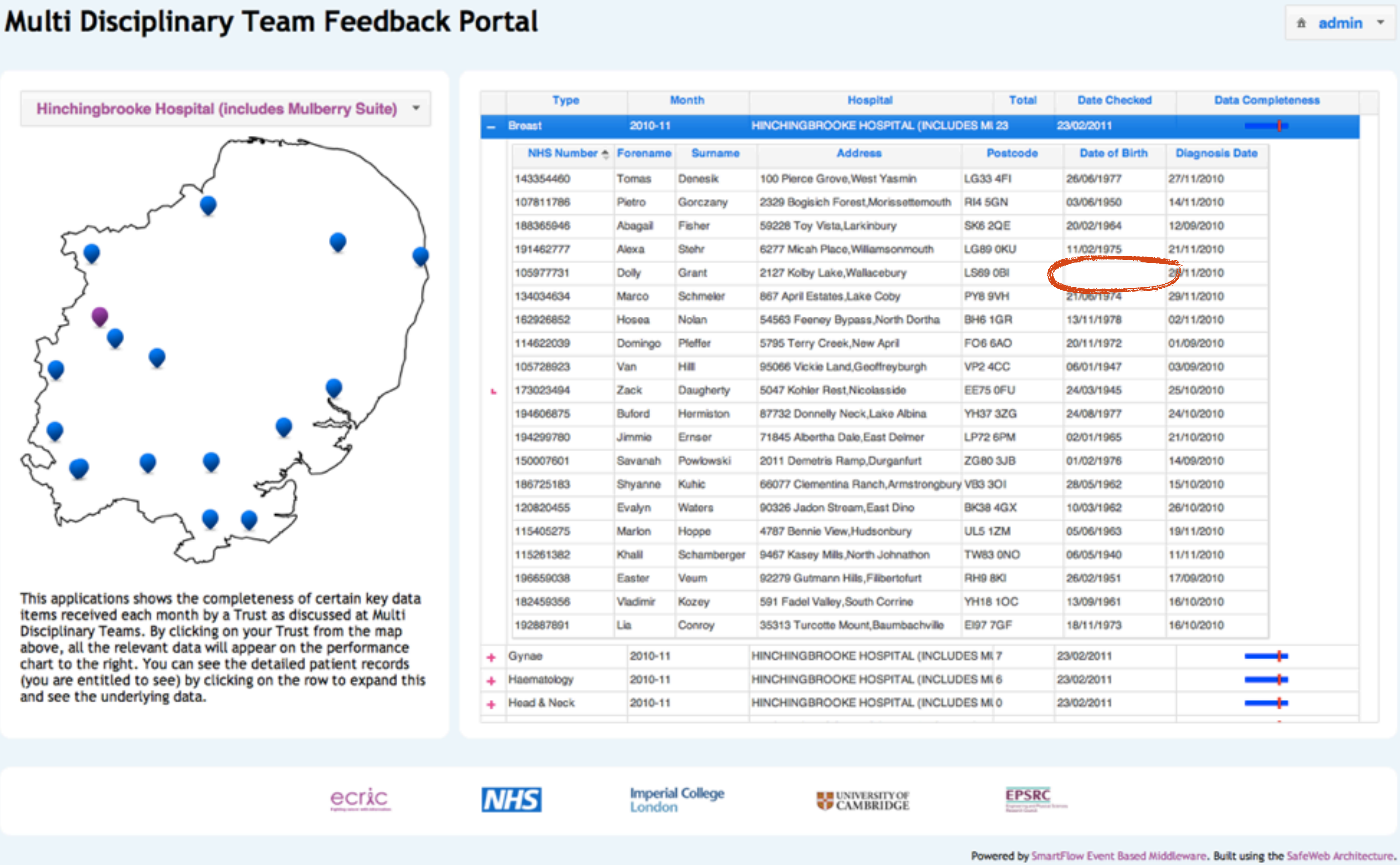
- No changes at organisational or infrastructure level
- Reusing the components of existing system implemented in Ruby



Real-world deployment

Within ECRiC

Multi Disciplinary Team Feedback Portal



This applications shows the completeness of certain key data items received each month by a Trust as discussed at Multi Disciplinary Teams. By clicking on your Trust from the map above, all the relevant data will appear on the performance chart to the right. You can see the detailed patient records (you are entitled to see) by clicking on the row to expand this and see the underlying data.

	Type	Month	Hospital	Total	Date Checked	Data Completeness
–	Breast	2010-11	HINCHINGBROOKE HOSPITAL (INCLUDES MI	23	23/02/2011	

NHS Number	Forename	Surname	Address	Postcode	Date of Birth	Diagnosis Date
143354460	Tomas	Denesik	100 Pierce Grove,West Yasmin	LG33 4FI	26/06/1977	27/11/2010
107811786	Pietro	Gorczany	2329 Bogisich Forest,Morissettemouth	RI4 5GN	03/06/1950	14/11/2010
188365946	Abagail	Fisher	59228 Toy Vista,Larkinbury	SK6 2QE	20/02/1964	12/09/2010
191462777	Alexa	Stehr	6277 Micah Place,Williamsonmouth	LG89 0KU	11/02/1975	21/11/2010
105977731	Dolly	Grant	2127 Kolby Lake,Wallacebury	LS69 0BI		2[illegible]/11/2010
134034634	Marco	Schmeler	867 April Estates,Lake Coby	PY8 9VH	21/06/1974	29/11/2010
162926852	Hosea	Nolan	54563 Feeney Bypass,North Dortha	BH6 1GR	13/11/1978	02/11/2010
114622039	Domingo	Pfeffer	5795 Terry Creek,New April	FO6 6AO	20/11/1972	01/09/2010
105728923	Van	Hill	95066 Vickie Land,Geoffreyburgh	VP2 4CC	06/01/1947	03/09/2010
173023494	Zack	Daugherty	5047 Kohler Rest,Nicolasside	EE75 0FU	24/03/1945	25/10/2010
194606875	Buford	Hermiston	87732 Donnelly Neck,Lake Albina	YH37 3ZG	24/08/1977	24/10/2010
194299780	Jimmie	Ernser	71845 Albertha Dale,East Delmer	LP72 6PM	02/01/1965	21/10/2010
150007601	Savanah	Powlowski	2011 Demetris Ramp,Durganfurt	ZG80 3JB	01/02/1976	14/09/2010
186725183	Shyanne	Kuhic	66077 Clementina Ranch,Armstrongbury	VB3 3OI	28/05/1962	15/10/2010
120820455	Evalyn	Waters	90326 Jadon Stream,East Dino	BK38 4GX	10/03/1962	26/10/2010
115405275	Marlon	Hoppe	4787 Bennie View,Hudsonbury	UL5 1ZM	05/06/1963	19/11/2010
115261382	Khalil	Schamberger	9467 Kasey Mills,North Johnathon	TW83 0NO	06/05/1940	11/11/2010
196659038	Easter	Veum	92279 Gutmann Hills,Filibertofurt	RH9 8KI	26/02/1951	17/09/2010
182459356	Vladimir	Kozey	591 Fadel Valley,South Corrine	YH18 1OC	13/09/1961	16/10/2010
192887891	Lia	Conroy	35313 Turcotte Mount,Baumbachville	EI97 7GF	18/11/1973	16/10/2010

+	Gynae	2010-11	HINCHINGBROOKE HOSPITAL (INCLUDES MI	7	23/02/2011	
+	Haematology	2010-11	HINCHINGBROOKE HOSPITAL (INCLUDES MI	6	23/02/2011	
+	Head & Neck	2010-11	HINCHINGBROOKE HOSPITAL (INCLUDES MI	0	23/02/2011	

Powered by SmartFlow Event Based Middleware. Built using the SafeWeb Architecture.

Security improvements

Prevents common vulnerabilities (from CVE database)

type of vulnerability	related CVE reports
omitted access checks	2011-0701, 2010-2353, 2010-0752
errors in access checks	2011-0449, 2010-3092, 2010-4403
inappropriate access checks	2010-4775, 2009-2431
design errors	2011-0899, 2010-3933

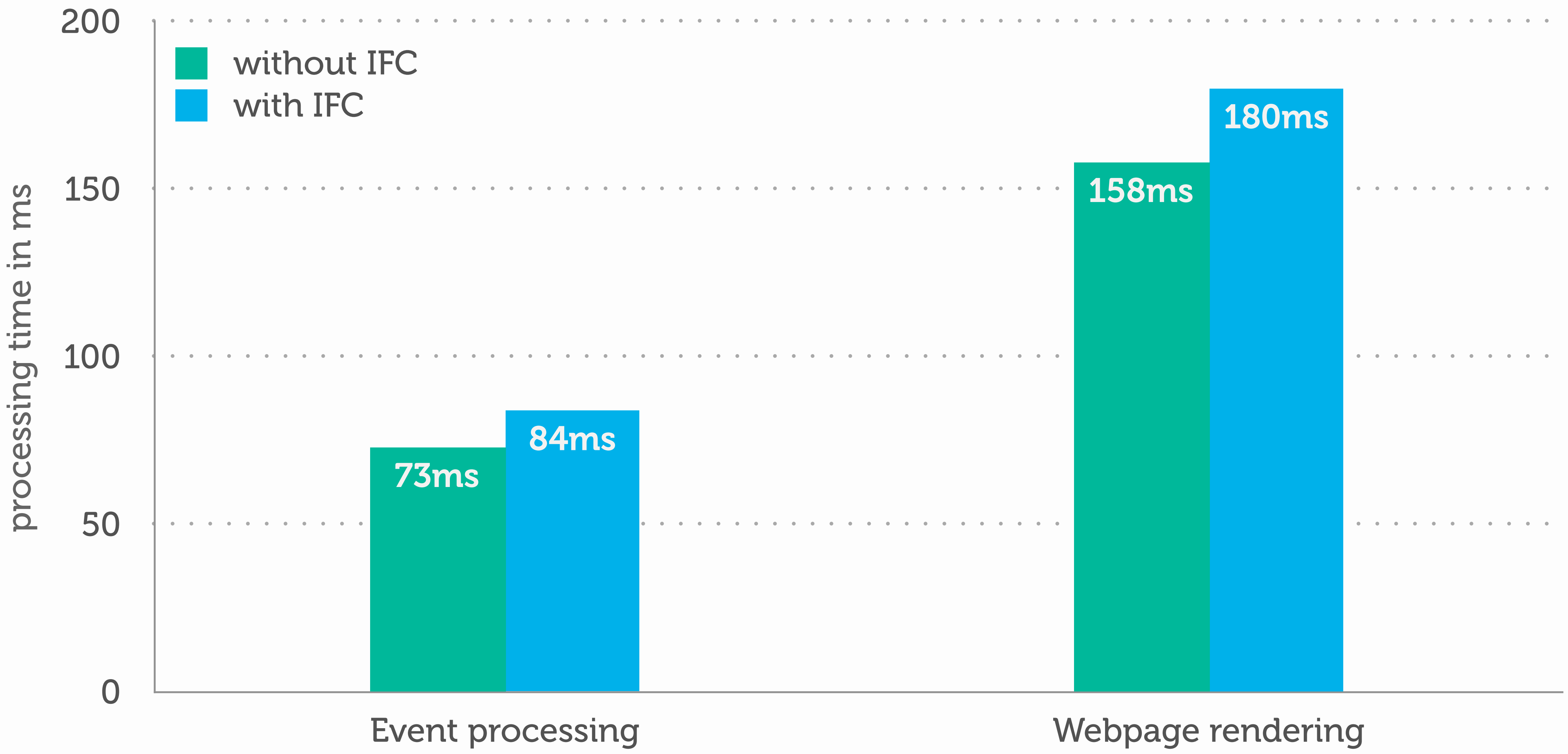
Only a small trusted code-base requires code audit

3121 LOC feedback portal

2841 LOC unprivileged code

280 LOC privileged code

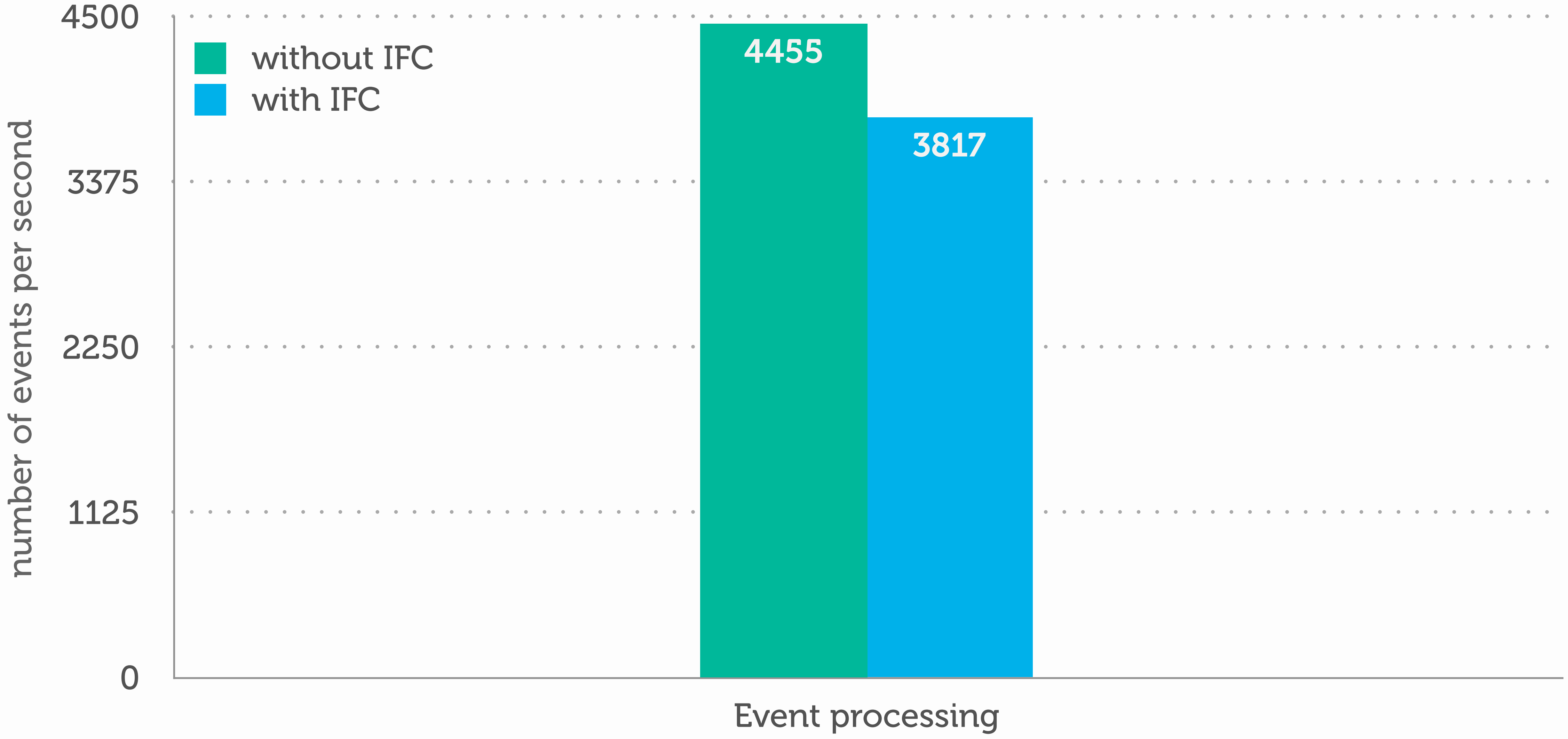
+14% **event processing** latency
+15% **webpage rendering** latency



Measured the time to handle/process 1000 requests/events

Taken on AMD Opteron 6136 2.4GHz, 16GiB of RAM, Ubuntu 10.04

-17% **event processing** throughput



Sampled throughput once per second for 1000 seconds

Taken on AMD Opteron 6136 2.4GHz, 16GiB of RAM, Ubuntu 10.04

Conclusions

Control over data flows in enterprise web applications

- Data protection across multiple layers at different granularities

Strong end-to-end security guarantees

- Application of information flow control to both back-end & front-end

The importance of efficient isolation as lesson learned

- Necessary to ensure data protection & prevent undisclosed data leaks

Real-world demonstration in a healthcare environment

- Part of a web application for assisting cancer treatment practices within the UK NHS

SafeWeb, part of the EPSRC-funded SmartFlow project
http://smartflow.org/safeweb

Related work

Taint tracking for Ruby on Rails web application framework

Burket, J., Mutchler, P., Weaver, M., Zaveri, M., Evans, D.: GuardRails: A data-centric web application security framework (2011)

Taint tracking using fine grained policy objects and source code rewriting

Yip, A., Wang, X., Zeldovich, N., Kaashoek, M. F.: Improving Application Security with Data Flow Assertions (2009)

Using Java bytecode rewriting to propagate labels

Yoshihama, S., Yoshizawa, T., Watanabe, Y. Kudoh, M. Oyanagi, K.: Dynamic Information Flow Control Architecture for Web Applications (2007)

Java thread isolation allowing communication only through labeled data

Migliavacca, M., Papagiannis, I., Eyers, D., Shand, B., Bacon, J., Pietzuch, P.: High-performance event processing with information security (2010)

JVM runtime modifications to support label tracking

Roy, I., Porter, D., Bond, M., McKinley, K., Witchel, E.: Laminar: Practical fine-grained decentralized information flow control (2009)