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

C₁ can output event iff \( \{ \text{ecric.org.uk/patient} \} \subseteq \{ \text{ecric.org.uk/patient/120820455} \} \)
C₂ can input event iff \( \{ \text{ecric.org.uk/patient/120820455} \} \subseteq \{ \text{ecric.org.uk/patient/*} \} \)

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

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

Taint propagation
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
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

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
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
Multi Disciplinary Team Feedback Portal

Hinchingbrooke Hospital (includes Mulberry Suite)

This application 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.

<table>
<thead>
<tr>
<th>Type</th>
<th>Month</th>
<th>Hospital</th>
<th>Total</th>
<th>Date Checked</th>
<th>Data Completeness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breast</td>
<td>2010-11</td>
<td>HINCHINGBROOKE HOSPITAL (INCLUDES MI 23)</td>
<td>23/02/2011</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NHS Number</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Forename</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Surname</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Address</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Postcode</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Date of Birth</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Diagnosis Date</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

143354460  Tomas  Denesik  100 Pierce Grove, West Yasmin  LG3 4FI  26/06/1977  27/11/2010
107811786  Pietro  Gorgczany  2329 Bogich Forest, Morisettometh  RH4 5GN  03/06/1950  14/11/2010
183865946  Abagail  Fischer  59228 Toy Vista, Larkinbury  SK8 2OE  20/02/1964  12/09/2010
191462777  Alexa  Stahr  6277 Micah Place, Williamsonmouth  LG89 0KU  11/02/1975  21/11/2010
105977731  Dolly  Grant  2127 Kobly Lake, Wallacebury  LS69 0BI  21/06/1974  29/11/2010
134034634  Marco  Schmeler  867 April Estates, Lake Coby  PY8 9VH  21/06/1974  29/11/2010
182926852  Hosea  Nolan  54563 Feeney Bypass, North Dortha  BH6 1GR  13/11/1978  02/11/2010
114622039  Domingo  Pfeffer  5795 Terry Creek, New April  FO6 6AQ  20/11/1972  01/09/2010
105782037  Van  Hill  95066 Vickle Land, Geoffreyreub  VF2 4GC  06/01/1947  03/08/2010
173025494  Zack  Daugherty  5047 Kohler Rest, Nicolasside  EE76 0FU  24/03/1945  25/10/2010
194606875  Bufo  Hermiston  87732 Donnelly Neck, Lake Alibua  YH67 3ZG  24/08/1977  24/10/2010
194299780  Jimmie  Ernser  71845 Aibrthera Dale, East Delmer  LP72 6PM  02/01/1965  21/11/2010
150007061  Savaanah  Powlsowski  2011 Demitr Rump, Durganfurt  ZG80 3JB  01/02/1976  14/09/2010
186721831  Shyanne  Kuhic  66077 Clementina Ranch, Armstrongbury V83 30I  28/05/1962  15/10/2010
120820455  Evlyn  Waters  90326 Jadon Stream, East Dino  BK38 4GX  10/03/1962  26/10/2010
115402572  Marco  Hoppe  48778 Benny View, Hudsonbury  UL5 1ZM  05/06/1963  19/11/2010
115213822  Khalil  Schamberger  94617 Kassey Mills, North Johnathon  TW8 3NO  06/05/1940  17/09/2010
196665038  Easter  Veum  92379 Gustmann Hille, Filbertofurt  RH9 8GI  26/02/1951  17/09/2010
182459356  Vladimir  Kozey  591 Fadel Valley, South Corrine  YH18 1OC  13/09/1961  16/10/2010
192878919  Lia  Corney  35313 Turcote Mount, Baunbachville  EI77 7GF  18/11/1973  16/10/2010

Gynaecology  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

Prevents common vulnerabilities (from CVE database)

<table>
<thead>
<tr>
<th>type of vulnerability</th>
<th>related CVE reports</th>
</tr>
</thead>
<tbody>
<tr>
<td>omitted access checks</td>
<td>2011-0701, 2010-2353, 2010-0752</td>
</tr>
<tr>
<td>errors in access checks</td>
<td>2011-0449, 2010-3092, 2010-4403</td>
</tr>
<tr>
<td>inappropriate access checks</td>
<td>2010-4775, 2009-2431</td>
</tr>
<tr>
<td>design errors</td>
<td>2011-0899, 2010-3933</td>
</tr>
</tbody>
</table>

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

Taint tracking using fine grained policy objects and source code rewriting

Using Java bytecode rewriting to propagate labels

Java thread isolation allowing communication only through labeled data

JVM runtime modifications to support label tracking