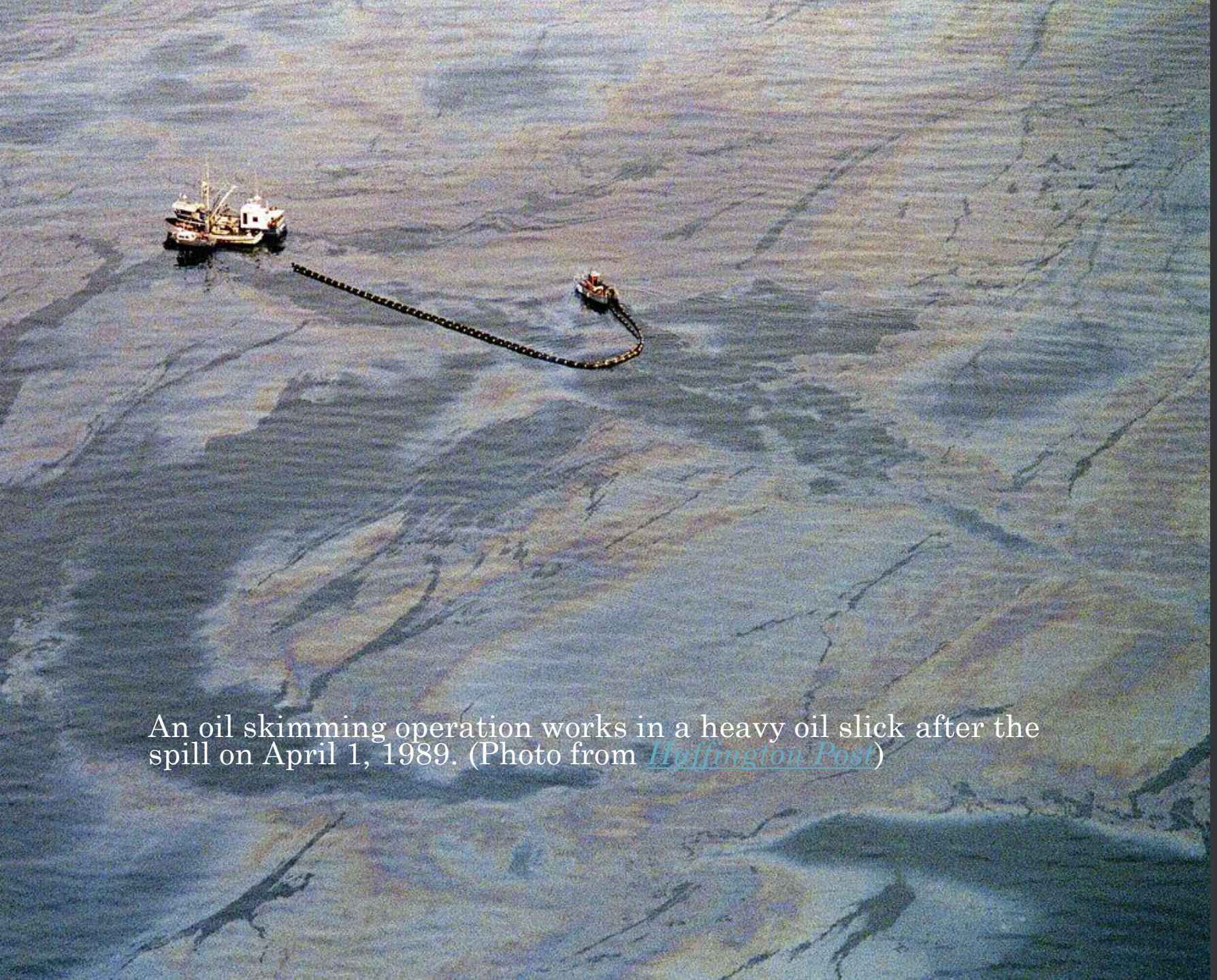
Precise, Dynamic Information Flow for Database-Backed Applications

Jean Yang, Travis Hance, Thomas H. Austin, Armando Solar-Lezama, Cormac Flanagan, and Stephen Chong

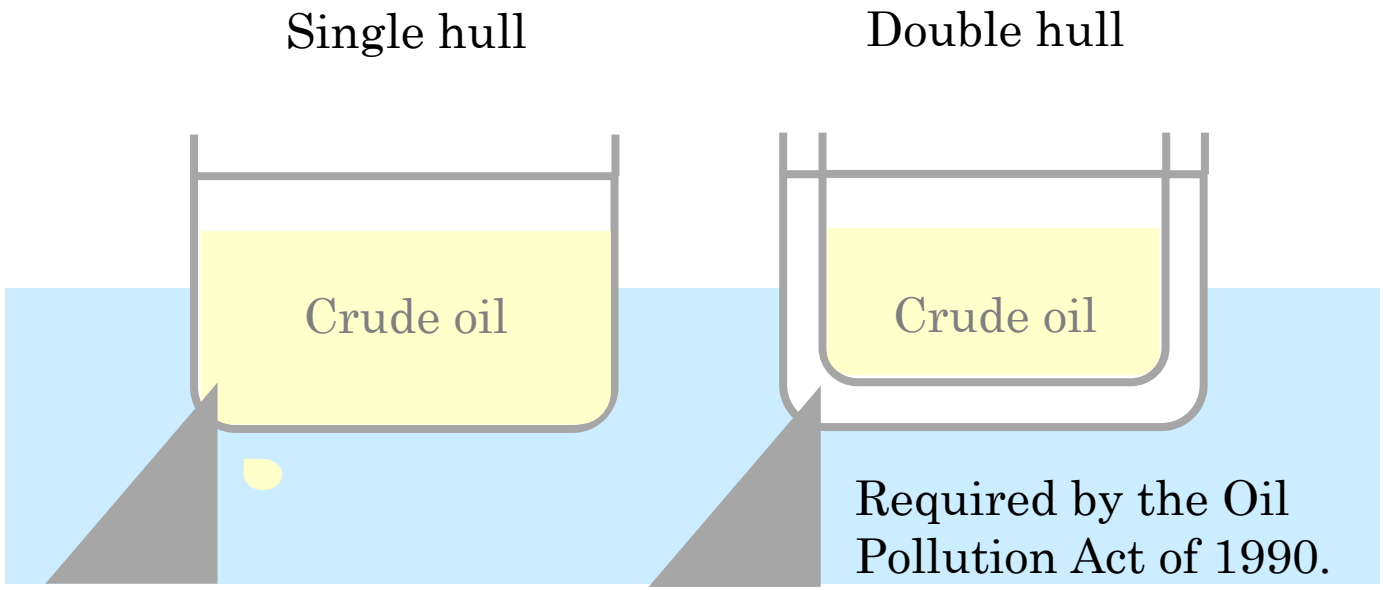
PLDI 2016



An oil skimming operation works in a heavy oil slick after the spill on April 1, 1989. (Photo from *Huffington Post*)

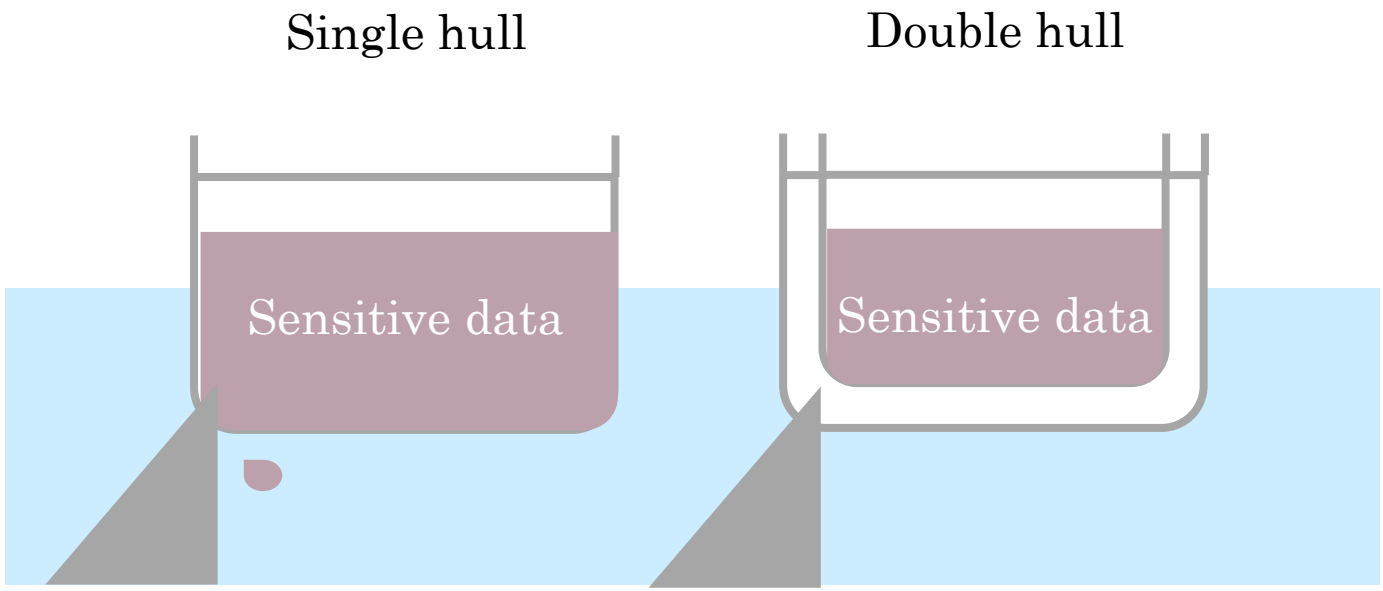
Oil-covered otter. (Photo from the Human Impact Project)

The Relationship Between Design and Accidents




But what about information leaks?

Wanted: Double Hull for Information Security



Research in language-based security looks at designs for double hulls [Sabelfeld and Myers, JSAC 2003].

Our goal: make double hulls that are as easy to construct as possible!

This Talk: Making It Easier to Secure Web Programs

1. Why it's hard to prevent information leaks.
2. A programming model that makes writing secure web programs easier.
3. How we support that programming model in database-backed applications.

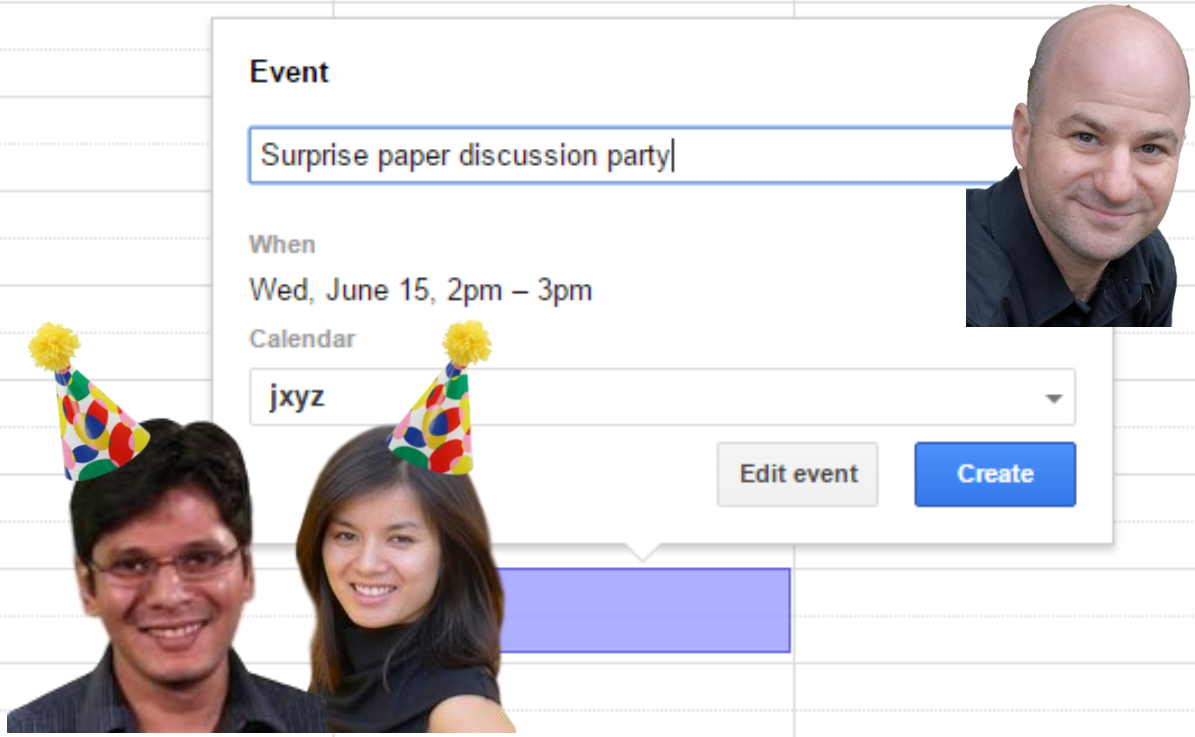
Tue 6/14 Wed 6/15 Thu 6/16 Fri 6/17

ilton Resort - The Fess Parker A Doubletree by Hilton Resort

W PLDI

Social Calendar Example

Let's say Arjun and I want to throw a surprise paper discussion party for Emery.



Challenge: Different Viewers Should See Different Events

Guests

Surprise discussion for Emery at **Chuck E. Cheese.**

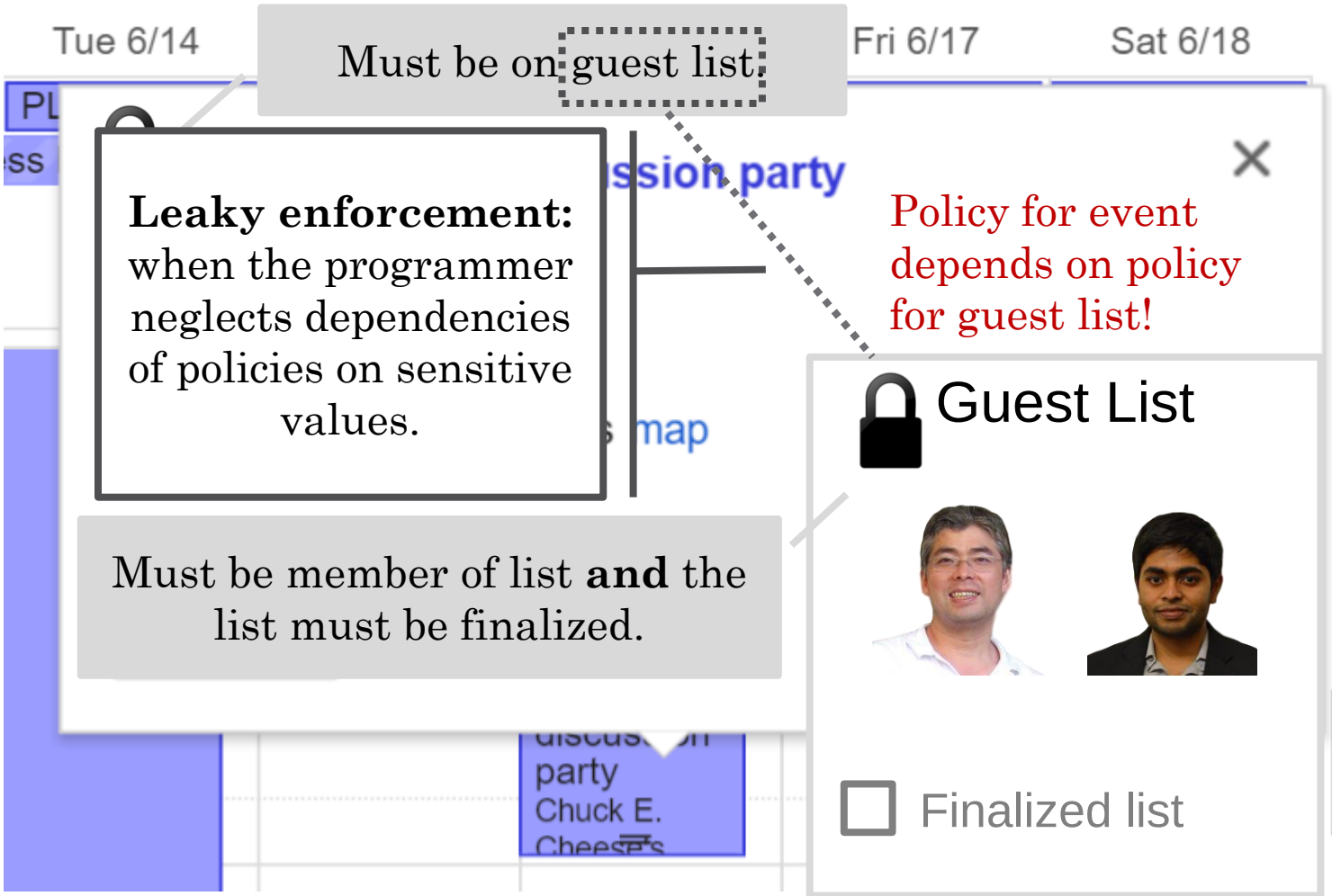
Emery

Pizza with Arjun/Jean.

Strangers

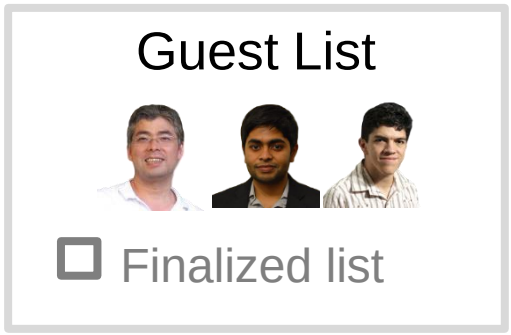
Private event at Chuck E. Cheese.

Policies May Depend on Sensitive Values

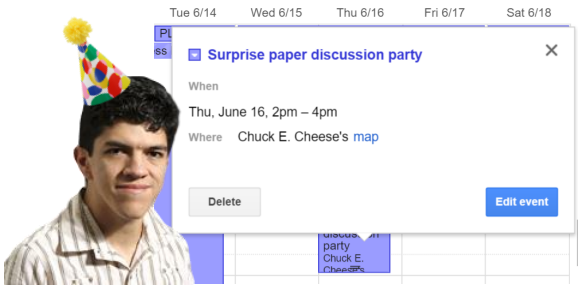


A Story of Leaky Enforcement

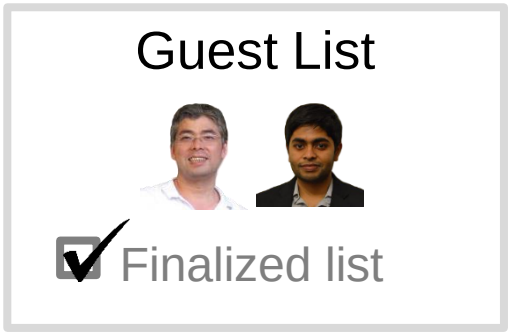
1 We add Armando to non-final guest list.



2 Armando sees the event on his calendar.



3 We run out of space and remove Armando.



4 Armando figures out he was uninvited.

A Story of Leaky Enforcement

Problem: implementation for event policy neglected to take into account guest list policy.

This arises whenever we trust programmers to get policy checks right!

Need to Track Policies and Viewers Across the Code

"What is the most popular location among friends 7pm Tuesday?"

Update to all calendar users

Need to track how information flows through derived values *and* where derived values flow!

“Policy Spaghetti” in HotCRP

```
        if ($ptype === 0)
            $ptype = "";
        if ($this->privChair && !$ptype && $Conf->timeUpdatePaper())
            $this->limitName = "all";
        else if (($me->privChair && $ptype == "act")
                 || ($me->isPC
                     && (!$ptype || $ptype == "act" || $ptype == "all")
                     && $Conf->can_pc_see_all_submissions()))
            $this->limitName = "act";
        else if ($me->privChair && $ptype == "unm")
            $this->limitName = "unm";
        else if ($me->isPC && (!$ptype || $ptype == "s" || $ptype == "unm"))
            $this->limitName = "s";
        else if ($me->isPC && ($ptype == "und" || $ptype == "undec"))
            $this->limitName = "und";
        else if ($me->isPC && ($ptype == "acc" || $ptype == "revs"
                               || $ptype == "reqrevs" || $ptype == "req"
                               || $ptype == "lead" || $ptype == "rable"
                               || $ptype == "manager"))
            $this->limitName = $ptype;
        else if ($this->privChair && ($ptype == "all" || $ptype == "unsub"))
            $this->limitName = $ptype;
        else if ($ptype == "r" || $ptype == "rout" || $ptype == "a")
            $this->limitName = $ptype;
        else if ($ptype == "rable")
            $this->limitName = "r";
        else if (!$me->is_reviewer())
            $this->limitName = "a";
        else if (!$me->is_author())
            $this->limitName = "r";
        else
            $this->limitName = "ar";

        // track other information
        $this->allowAuthor = false;
        if ($me->privChair || $me->is_author()
            || ($this->amPC && $Conf->submission_blindness() != Conference::BLIN
D_ALWAYS))
                                                      396,16        11%
```

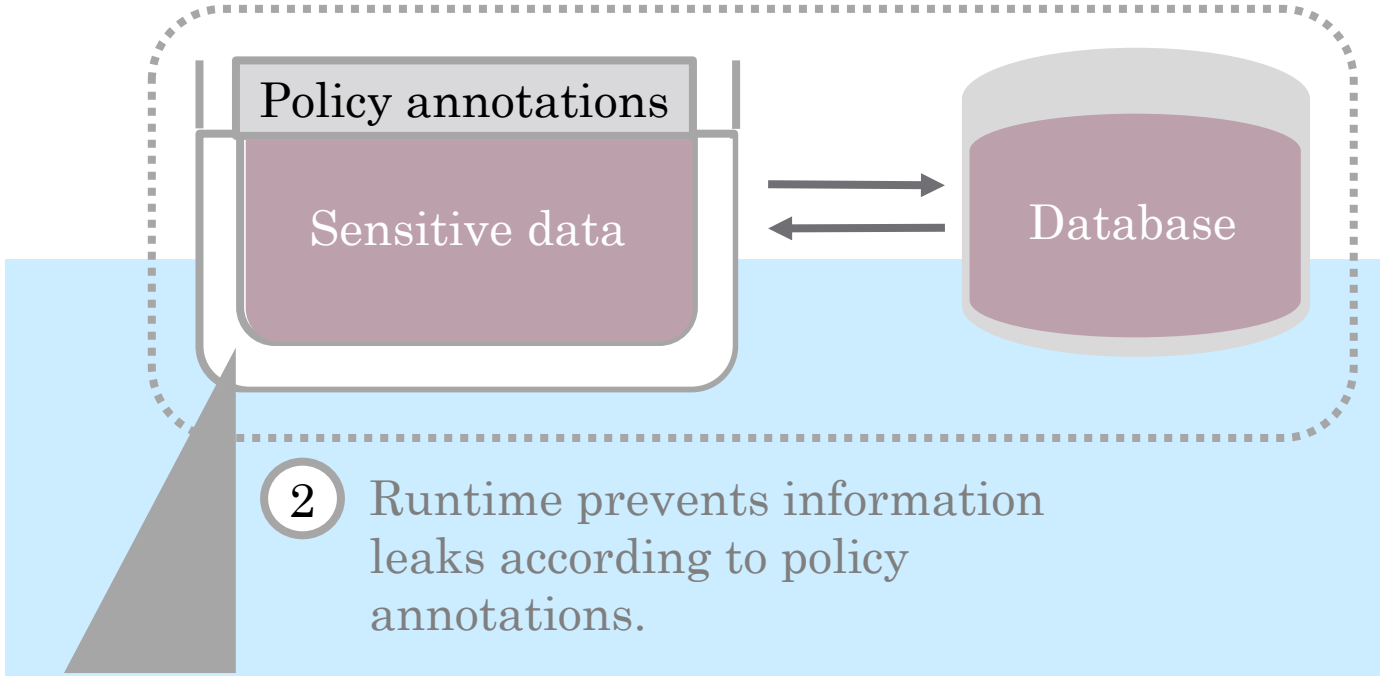
```
        // if a complex request, or a form upload, don't search
        foreach ($_REQUEST as $k => $v)
            if ($k != "p" && $k != "paperId" && $k != "m" && $k != "mode"
                && $k != "forceShow" && $k != "go" && $k != "actas"
                && $k != "ls" && $k != "t"
                && !isset($_COOKIE[$k]))
                return false;

        // if no paper ID set, find one
        if (!isset($_REQUEST["paperId"])) {
            $q = "select min(Paper.paperId) from Paper ";
            if ($Me->isPC)
                $q .= "where timeSubmitted>0";
            else if ($Me->has_review())
                $q .= "join PaperReview on (PaperReview.paperId=Paper.paperId an
d PaperReview.contactId=$Me->contactId)";
            else
                $q .= "join ContactInfo on (ContactInfo.paperId=Paper.paperId an
d ContactInfo.contactId=$Me->contactId and ContactInfo.conflictType>=" . CONFLIC
T_AUTHOR . ")";
            $result = $Conf->q($q);
            if (($paperId = edb_row($result)))
                $_REQUEST["paperId"] = $paperId[0];
            return false;
        }

        // if invalid contact, don't search
        if ($Me->is_empty())
            return false;

        // actually try to search
        if ($_REQUEST["paperId"] == "(All)")
            $_REQUEST["paperId"] = "";
        $search = new PaperSearch($Me, array("q" => $_REQUEST["paperId"], "t" =>
defval($_REQUEST, "t", 0)));
        $pl = $search->paperList();
        if (count($pl) == 1) {
            $pl = $search->session_list_object();
                                                                      93%
```

Conditional permissions checks everywhere!

Jacqueline Web Framework to the Rescue!

1. Programmer specifies information flow policies separately from other functionality.

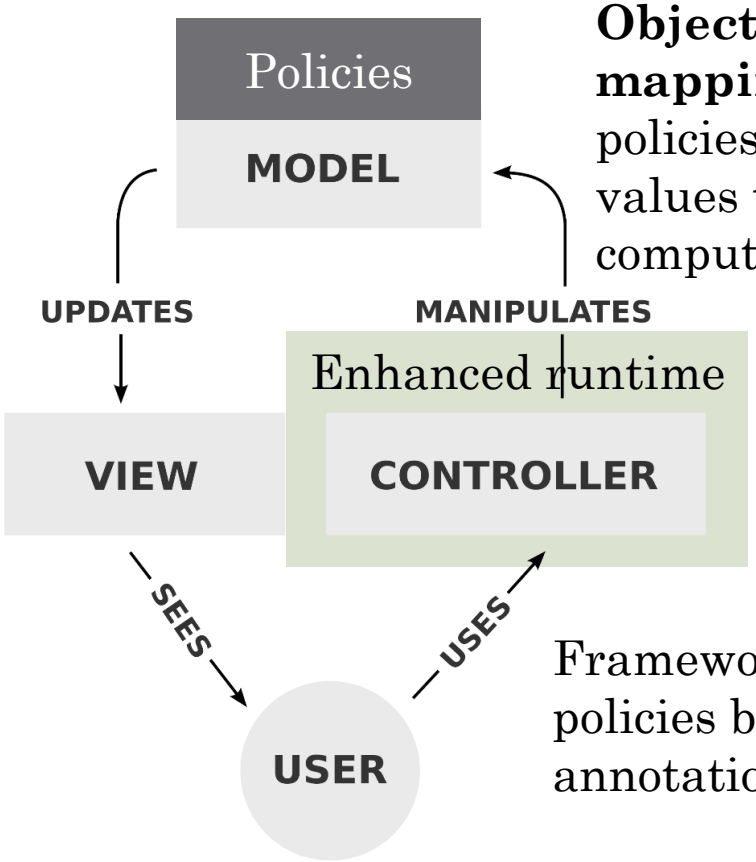
3. Enhanced runtime encompasses applications and databases, preventing leaks between the two.



Contributions

- *Policy-agnostic* programming model for database-backed web applications.
- Semantics and proofs for policy-agnostic programming that encompasses SQL databases.
- Demonstration of practical feasibility with Python implementation and application case studies.

Jacqueline Web Framework



Framework shows appropriate values based on viewer and policies.

Object-relational mapping propagates policies and sensitive values through computations.

Framework attaches policies based on annotations.

Coding in Jacqueline

Base schema

```
class Event(JacquelineModel):
  name = CharField(max_length=256)
  location = CharField(max_length=512)
  time = DateTimeField()
  description = CharField(max_length)=1024)
```

Policy helper functions

```
@jacqueline
def has_host(self, host):
  return EventHost.objects.get(
    event=self, host=host) != None
@jacqueline
def has_guest(self, guest):
  return EventGuest.objects.get(
    event=self, host=host) != None
```

Information flow policy for location field

```
@staticmethod
@label_for('location')
def restrict_event(event, ctxt):
  return event.has_host(ctxt) or event.has_guest(ctxt)
```

Public value for location field

```
@staticmethod
def jacqueline_get_private_location(event):
  return "Undisclosed location"
```

Centralized Policies in Jacqueline

```
class Event(Model):
    VISIBILITY = (('E', 'Everyone'), ('G', 'Guests' ))

    name = CharField(max_length=256)
    location = CharField(max_length=512)
    time = DateTimeField()
    description = CharField(max_length=1024)
    visibility = CharField(max_length=1, choices=VISIBILITY, default='E

    @jeeves
    def has_host(self, host):
        return EventHost.objects.get(event=self, host=host) != None

    @jeeves
    def has_guest(self, guest):
        return EventGuest.objects.get(event=self, guest=guest) != None
```

```
    @staticmethod
    def jeeves_get_private_name(event):
        return "Private event"
    @staticmethod
    def jeeves_get_private_location(event):
        return "Undisclosed location"
    @staticmethod
    def jeeves_get_private_time(event):
        return datetime.now(tz=pytz.utc)
    @staticmethod
    def jeeves_get_private_description(event):
        return "An event."

    @staticmethod
    @label_for('name', 'location', 'time', 'description')
    @jeeves
    def jeeves_restrict_event(event, ctxt):
        if event.visibility == 'G':
            return event.has_host(ctxt) or event.has_guest(ctxt)
        else:
            return True
```

Model

```
<h2 class="form-heading">Edit Your Event.</h2>
 <div class="form-group">
  <label for="name" class="control-label"><a rel="tooltip" title="TODO."><span class="glyphicon gl
  <div class="row">
    <div class="col-xs-4">
      <input type="text" class="form-control" name="name" id="name" value="{{ concretize(name) }}"
    </div>
  </div>
</div>
<div class="form-group">
  <label for="location" class="control-label"><a rel="tooltip" title="TODO."><span class="glyphico
  <div class="row">
    <div class="col-xs-4">
      <input type="text" class="form-control" name="location" id="location" value="{{ concretize(l
    </div>
  </div>
</div>
<div class="form-group">
  <label for="time" class="control-label"><a rel="tooltip" title="TODO."><span class="glyphicon gl
  <div class="row">
    <div class="col-xs-4">
      <input type="datetime" class="form-control" name="time" id="time" value="{{ concretize(time)
    </div>
  </div>
</div>
<div class="form-group">
  <label for="description" class="control-label"><a rel="tooltip" title="TODO."><span class="glyph
  <div class="row">
    <div class="col-xs-4">
      <textarea class="form-control" name="description" id="description">{{ concretize(description
    </div>
  </div>
</div>

<div class="form-group">
  <label for="visibility" class="control-label"><a rel="tooltip" title="TODO."><span class="glyphi
  <div class="row">
    <div class="col-xs-4">
      <div class="btn-group">
        <button type="button" class="btn btn-default {% if visiblity=='E' %}active{% endif %}" dat
        <button type="button" class="btn btn-default {% if visibility=='G' %}active{% endif %}" da
      </div>
    </div>
  </div>
</div>
<button class="btn btn-primary" type="submit" value="Submit" onClick="myApp.showPleaseWait();">Submi
</form>
```

View

```
@login_required
@request_wrapper
@jeeves
def profile_view(request, user_profile):
    profile = UserProfile.objects.get(username=request.user.username)
    if profile == None:
        profile = user_profile

    if request.method == 'POST':
        assert(request.user.username==userprofile.username)
        profile.name = request.POST.get('name', '')
        profile.email = request.POST.get('email', '')
        profile.save()

    host_events = EventHost.objects.filter(host=profile).all()
    guest_events = EventGuest.objects.filter(guest=profile).all()

    return ("profile.html", {
        "profile": profile,
        "is_own_profile": request.user.username==user_profile.username,
        "host_events": host_events,
        "guest_events": guest_events,
        "which_page": "profile",
    })

def register_account(request):
    if request.user.is_authenticated():
        return HttpResponseRedirect("index")

    if request.method == 'POST':
        form = UserCreationForm(request.POST)
        if form.is_valid():
            user = form.save()
            user.save()

            UserProfile.objects.create(
                username=user.username,
                email=request.POST.get('email', ''),
            )
```

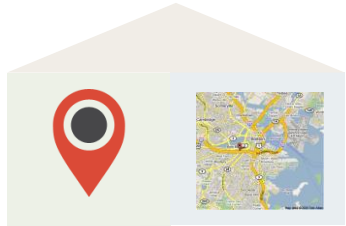
Controller

Centralized policies! No checks or declassifications needed anywhere else!

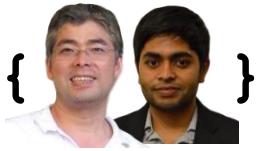
Closer Look at the Policy-Agnostic Runtime

Jeeves [Yang *et al* 2012, Austin *et al* 2013] uses facets [Austin *et al* 2012] to simulate simultaneous multiple executions.

1 Runtime propagates values and policies.

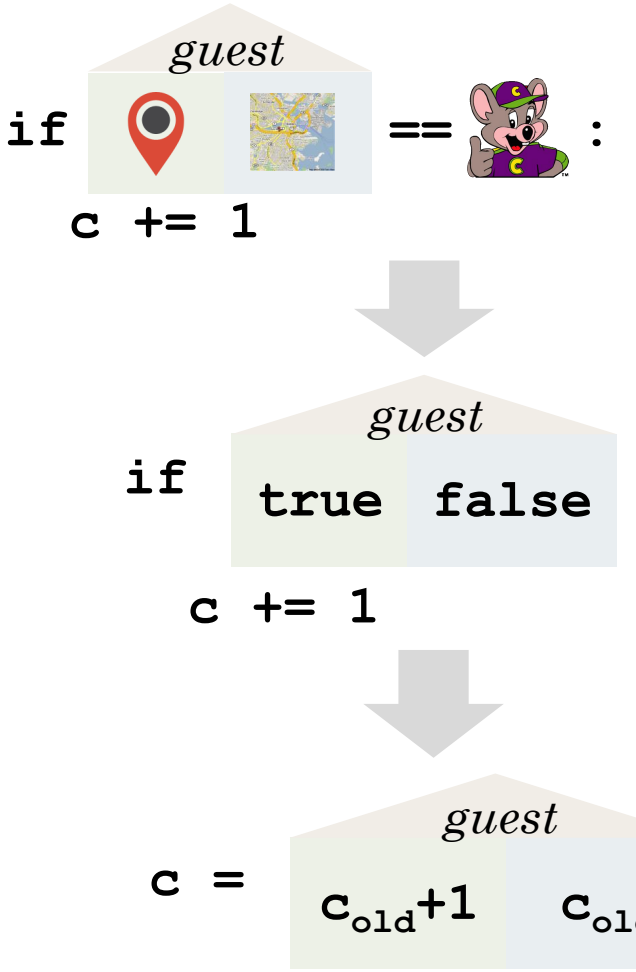
```
userCount = 0
if  ==  :
    userCount += 1
return userCount
```

2 Runtime solves for values to show based on policies and viewer.

```
print {  }        print {  }
```

1 0

Labels Track Sensitive Values to Prevent Leaks

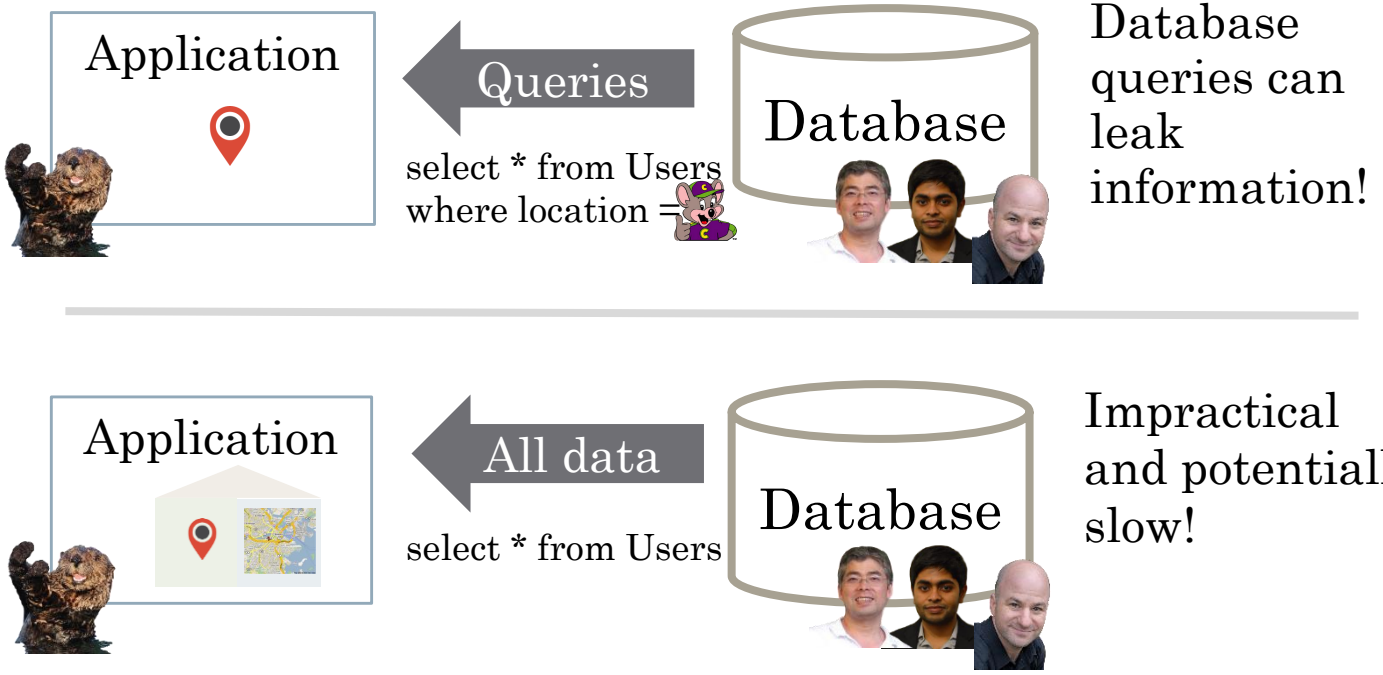


Labels follow values through all computations, including conditionals and assignments.

Emery can't see secret party information or results of computations on those values!


MOZAH
موزه

The Dangers of Interacting with Vanilla Databases



Challenge: Support faceted execution when interacting with an unmodified SQL database.

Need faceted queries!

Semantics of a Faceted Database

Conceptual row

Primary key	Location
1	

Store facets as strings?

New database for each label?

Too expensive! Too difficult to extend the formal semantics!

Solution: Use ORM to Map Facets onto Database Rows

Conceptual row

Primary key	Location
1	*a* 

Jeeves key	Location	Labels
1		$\{a\}$
1		$\{\neg a\}$

select * from Users
where location = 

Jeeves key	Location	Labels
1		$\{a\}$

ORM refacets

a

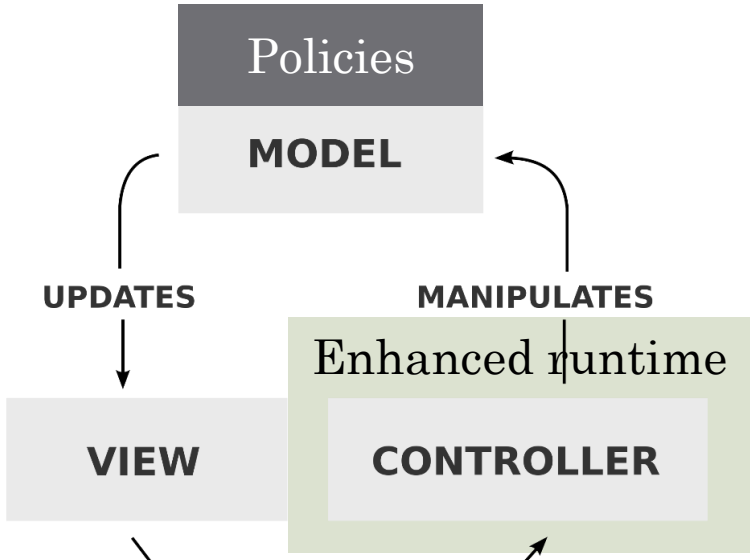
Jeeves key	Location	NULL
1		

Supporting Queries in Jacqueline

Jacqueline Supports	SQL Implements	ORM Implements
get	select	refaceting
all	select	refaceting
filter	select	refaceting
sort	order by	refaceting
foreign keys	join	-
save	delete, insert	turning a faceted value into multiple rows
delete	delete	keeping track of which facets to delete

Can use SQL implementations for many queries!

Early Pruning Optimization



Observation: Framework can often (but not always) track viewer.

Optimization: Can often explore fewer possible paths!

Precise, Dynamic Information Flow for Database-Backed Applications

Jean Yang
Carnegie Mellon University and Harvard Medical School, USA

Travis Hance
Dropbox, USA

Thomas H. Austin
San Jose State University, USA

Armando Solar-Lezama
Massachusetts Institute of Technology, USA

Cormac Flanagan
University of California, Santa Cruz, USA

Stephen Chong
Harvard University, USA

Abstract

We present an approach for dynamic information flow control across the application and database. Our approach reduces the amount of policy code required, yields formal guarantees across the application and database, works with existing relational database implementations, and scales for realistic applications. In this paper, we present a programming model that factors out information flow policies from application code and database queries, a dynamic semantics for the underlying λ^{JDB} core language, and proofs of termination-insensitive non-interference and policy compliance for the semantics. We implement these ideas in Jacqueline, a Python web framework, and demonstrate feasibility through three application case studies: a course manager, a health record system, and

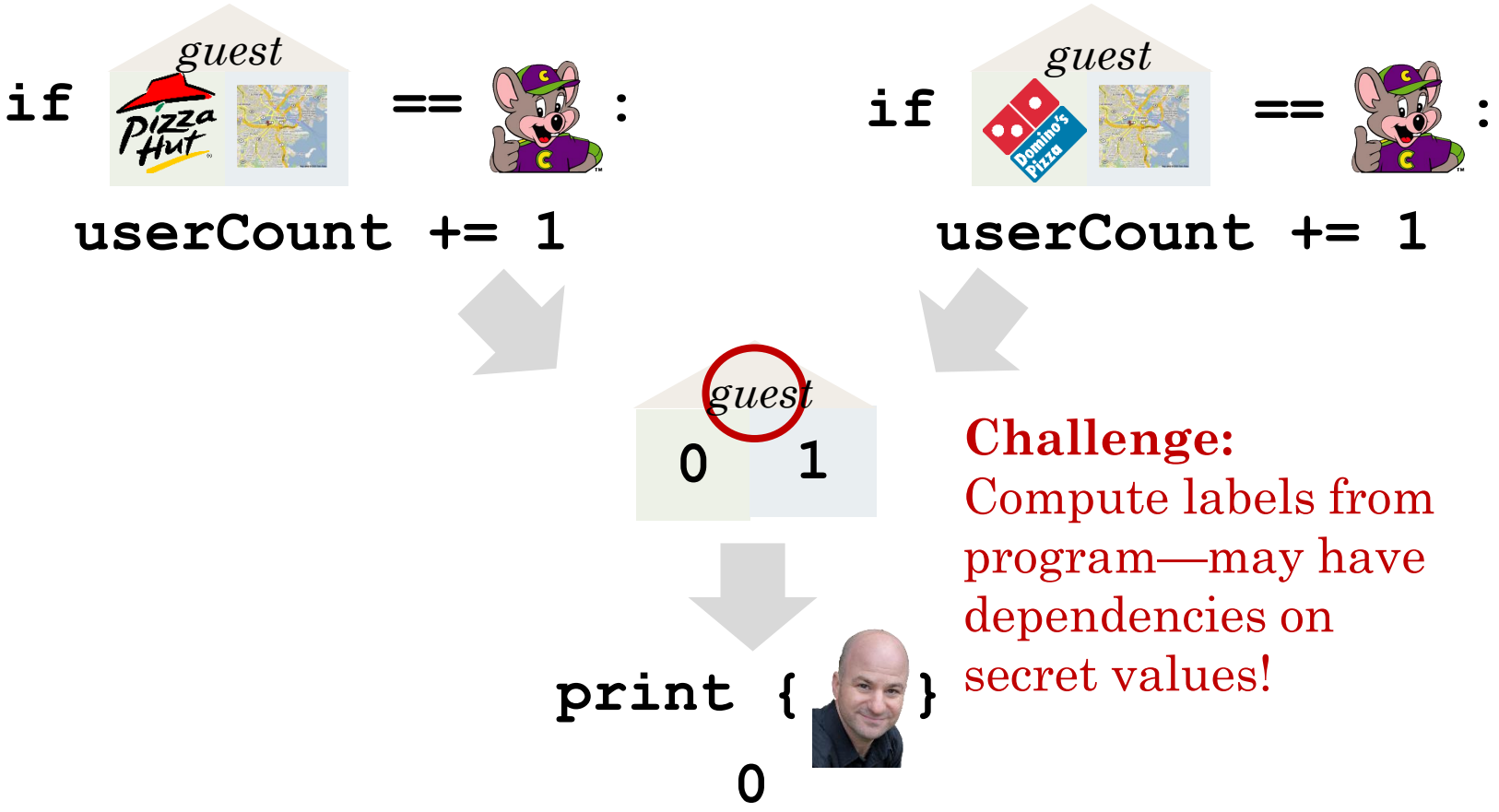
1. Introduction

From social networks to electronic health record systems, programs increasingly process sensitive data. As information leaks often arise from programmer error, a promising way to reduce leaks is to reduce opportunities for programmer error.

A major challenge in securing web applications involves reasoning about the flow of sensitive data across the application and database. According to the OWASP report [42], errors frequently occur at component boundaries. Indeed, the difficulty of reasoning about how sensitive data flows through both application code and database queries has led to leaks in systems from the HotCRP conference management system [3] to the social networking site Facebook [47]. The patch for the recent HotCRP bug involves policy checks

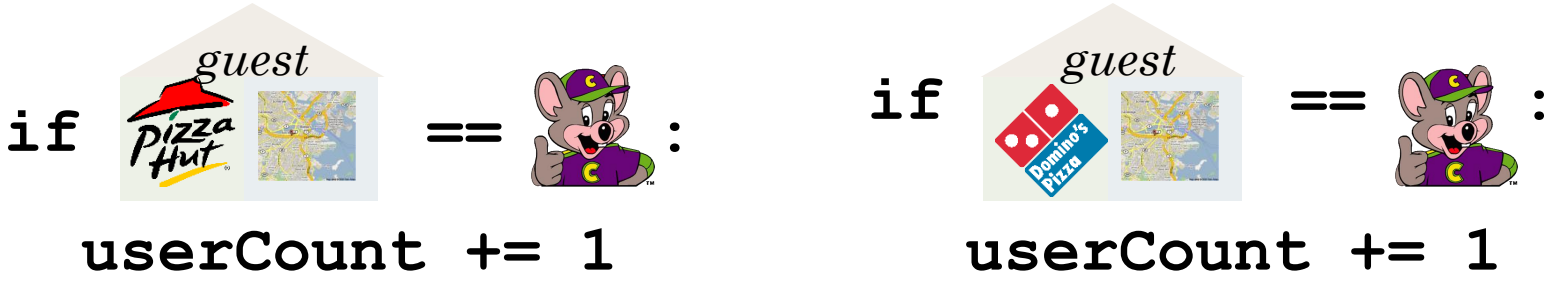
Review: Traditional Non-Interference

Secret values should not affect public output.



Challenge:
Compute labels from program—may have dependencies on secret values!

Policy-Agnostic Non-Interference



Theorem:
All executions where ***guest*** **must be public** produce equivalent outputs.

Can't tell apart secret values that require *guest* to be public.

Application Case Studies

Course manager

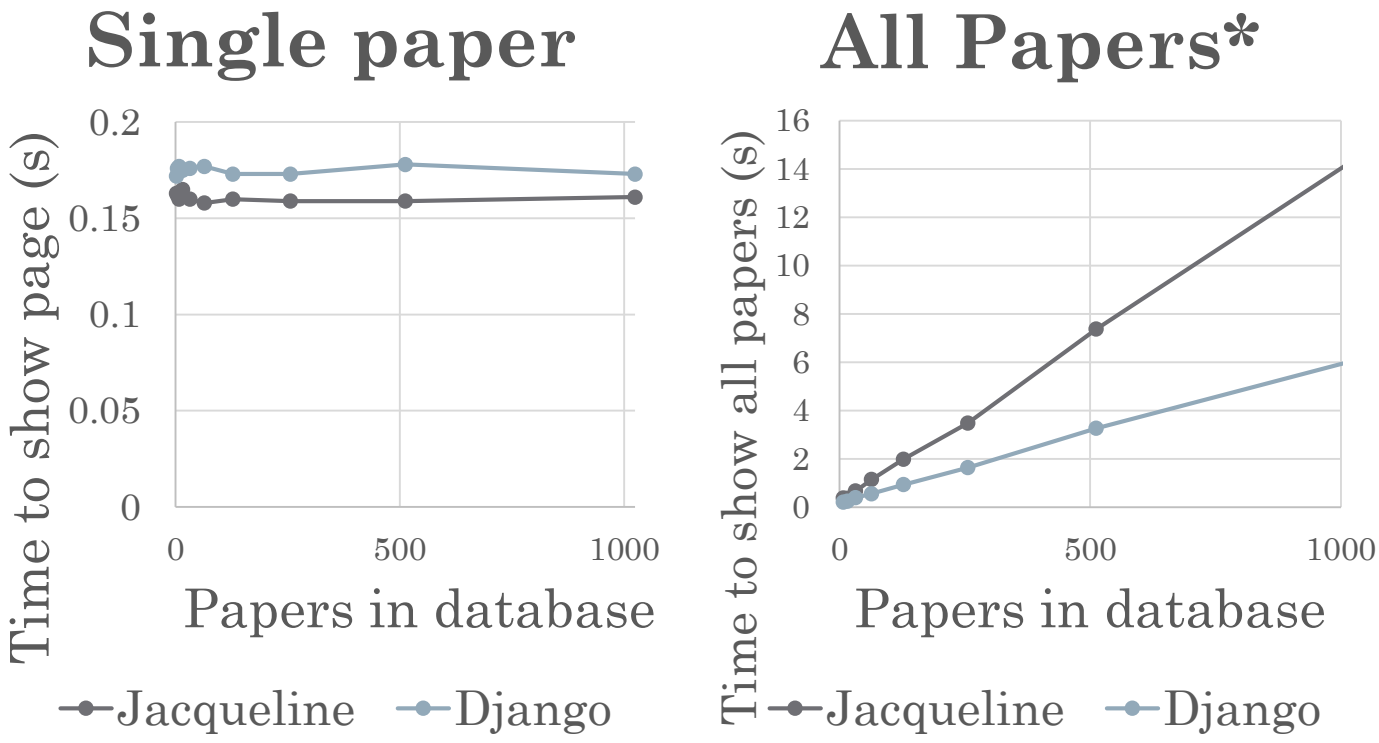
Health record manager

Conference management system (deployed!)

Jacqueline reduces the number of lines of policy code and has reasonable overheads!

Demo

Conference Management System Running Times



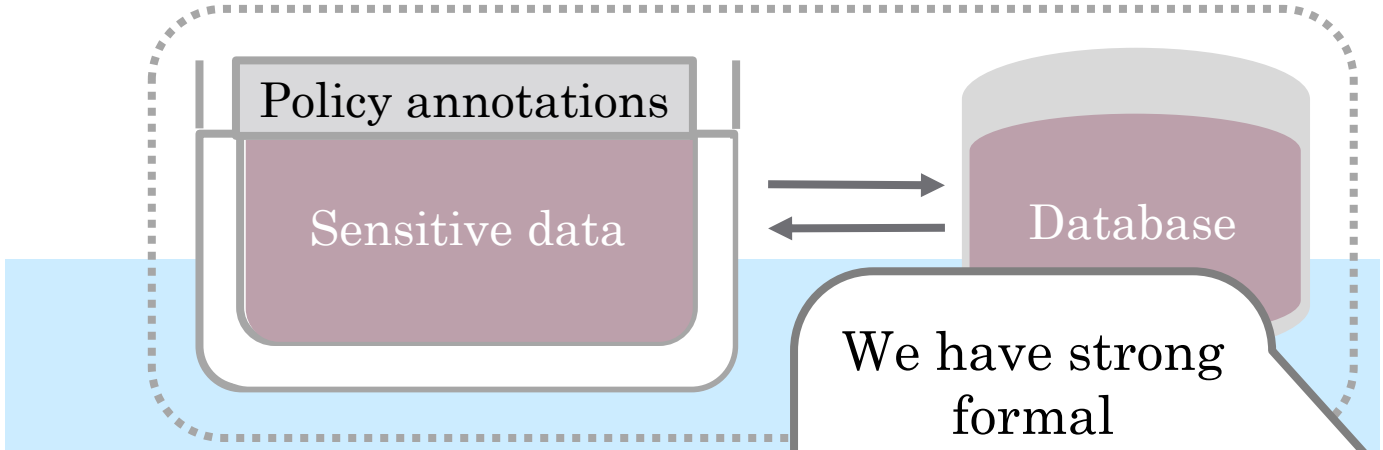
Tests from Amazon AWS machine via HTTP requests from another machine.

*Different from numbers in paper.

Summary: Policy-Agnostic Web Programming with Jacqueline

1. Programmer specifies information flow policies separately from other functionality.

3. Enhanced runtime encompasses applications and databases, preventing leaks between the two.



2. Runtime prevents information leaks according to policy annotations.

We have strong formal guarantees *and* evidence that this can be practical!


You *can* factor out information flow policies from other code to avoid policy spaghetti!
You *can* enforce policies across the application and database by using a carefully-crafted ORM!
You *can* build realistic systems using this approach!
http://jeeveslang.org
http://github.com/jeanqasaur/jeeves