Preventing Information Leaks with Jeeves

Jean Yang, MIT

April 8, 2015

CSAIL

Data, Data Everywhere

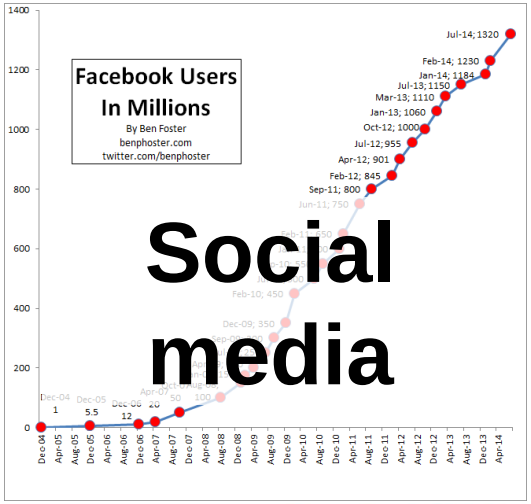
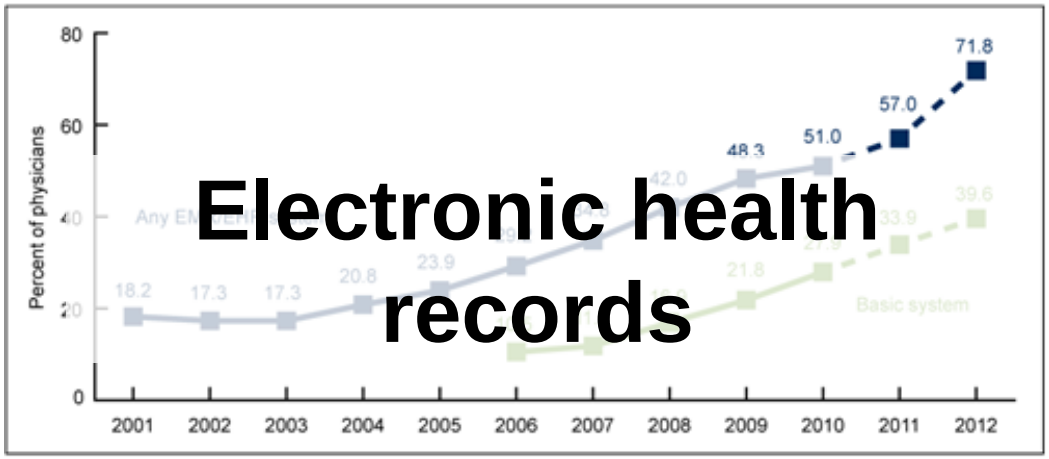
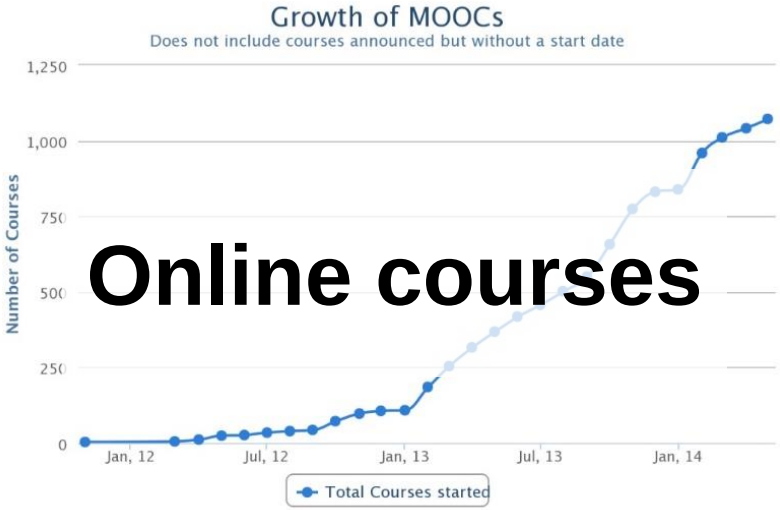
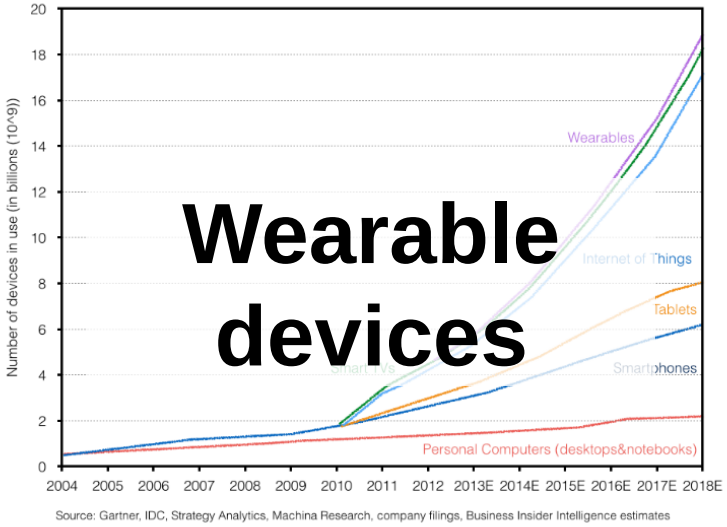


Figure 1. Percentage of office-based physicians with EMR/EHR systems: United States, 2001–2010 and preliminary 2011–2012

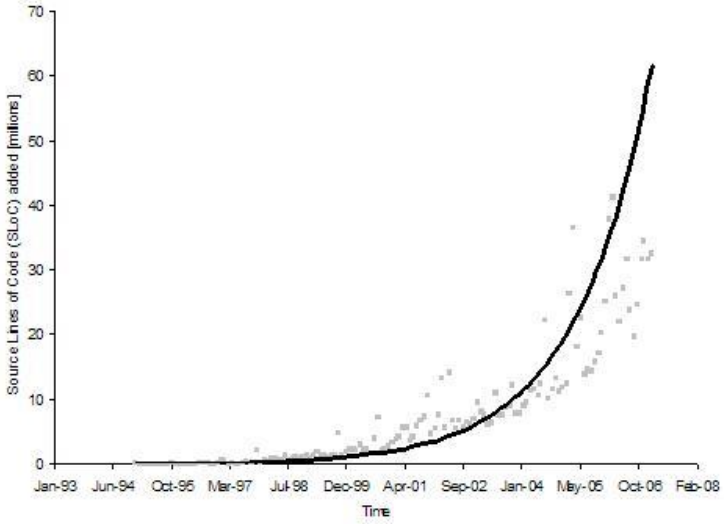






All Kinds of People Are Writing All Kinds of Code

Open source lines of code



Medical researchers

Social scientists

Journalists

Children

Even Trained Developers Leak Information

BUSINESS

Vodacom fixes phone number leaking bug

31 October 2014 15:18

How Facebook Explains User Data Bug That Leaked 6 Million People's Information

The Huffington Post | By Bianca Bosker

Posted: 06/25/2013 9:31 am EDT | Updated: 06/25/2013 3:47 pm EDT

Bang With Friends Isn't Actually That Anonymous

The Huffington Post | By Alexis Kleinman

Posted: 05/10/2013 11:39 am EDT | Updated: 05/11/2013 3:30 am EDT

HUFF POST TECH

Facebook apps leak personal data, says Symantec

Security company claims Facebook application coding erro allow third parties to access users' private details

Facebook apps 'leaking details to advertisers'

Social networking site will be concerned that applications such as FarmVille can reveal user IDs despite strong privacy settings

SHARE YOUR FRUSTRATIONS

HEALTH CARE NEWS

Open wide – your health information can even leak out from your dentist

Facebook only fixes privacy leak in new feature, group says

The leak allowed 'friends of friends' to see events another person attended

By Jeremy Kirk

March 17, 2013 10:53 PM ET | 1 Comment

COMPUTERWORLD

Vivino wine-lovers' app leaked personal information

By: Graham Cluley | comment : 2 | January 20, 2015 | Posted in: Industry News

Not OK, Cupid: dating site email security gaffe leaves your account wide open

Users may be exposing themselves without realizing it

By Adrianne Jeffries on August 20, 2013 01:00 pm Email @adrjeffries

Why Aren't Existing Approaches Enough?

Encrypting Data

But people are still showing the data wrong.

Defensive protection

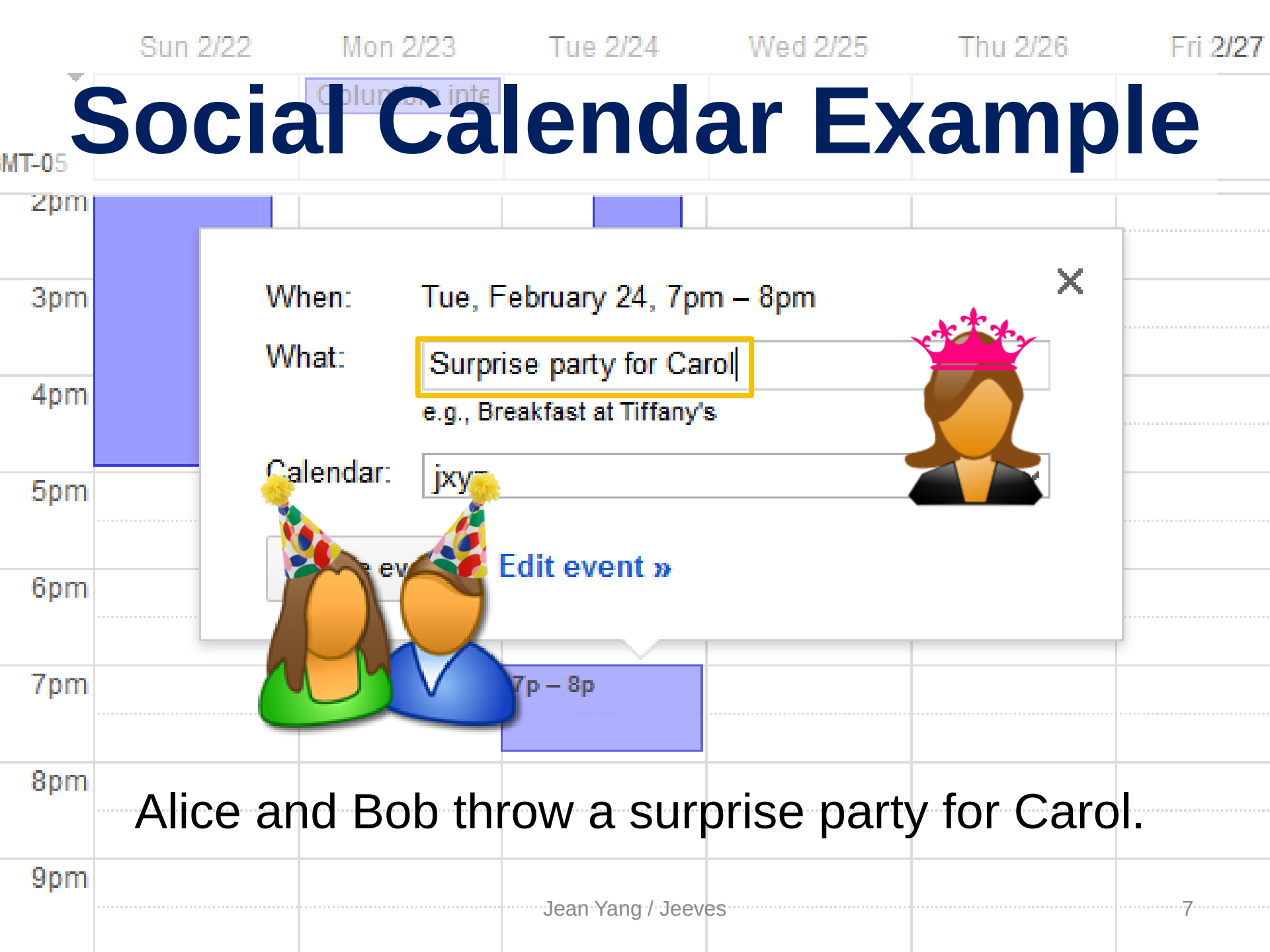
But leaves system builders a step behind.

My Approach: Privacy by Construction

Factor out privacy to reduce opportunity for leaks.

- Programmer specifies **high-level policies** about how sensitive data can be used.
- Rest of program is **policy-agnostic**.
- System **manages policies automatically**.

Social Calendar Example



Alice and Bob throw a surprise party for Carol.

Problem:
Even Seemingly Simple Policies Have Subtleties

Surprise party for Carol at Chuck E. Cheese.

Pizza with Alice/Bob.

Private event at Chuck E. Cheese.

Guests

Carol

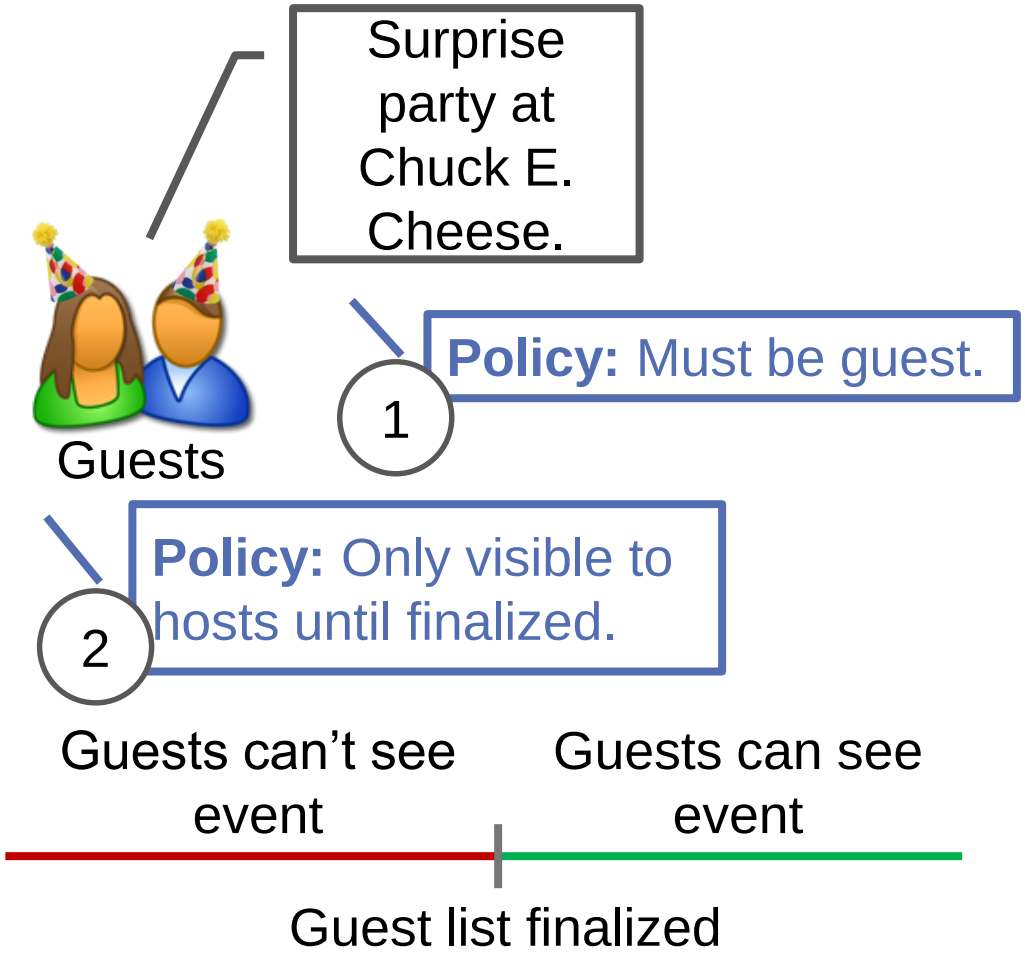
Strangers

Policy: Must be guest.

Policy: Only visible to hosts until finalized.

Policies can depend on sensitive values and other policies.

Enforcing Policies Can Leak Information!



- **Subtle mistake:** check for policy 1 neglects dependency on policy 2.
- Problem arises when programmers trusted to get dependencies right.

Problem:
Policies Are Intertwined Across the Code

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

Update to event subscribers

- Track information flow through derived values.
- Track where derived values flow.

“Policy Spaghetti” in Real Systems

```
        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
                $q .= "join ContactInfo on (ContactInfo.contactId=Paper.leadContactId an
d ContactInfo.contactId=$Me->contactId OR ".$Me->contactId." IN (SELECT contactId FROM PaperConflict WHERE conflictType>0))";
            $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"] == "1")
            $_REQUEST["paperId"] = "all";
        $search = new PaperSearch($Me, array("q" => $_REQUEST["paperId"], "t" =>
defval($_REQUEST, "t", 0)));
        $pl = $search->paperList();
        if (count($pl) == 1) {
            $pl = $search->session_list_object();
                                                        2143,24        93%
```

Code from HotCRP conference management system

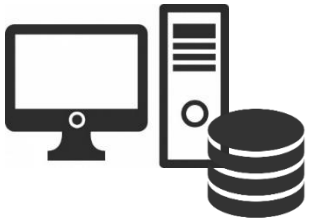
Highlighted: conditional permissions checks everywhere.

Automatic Enforcement with Jeeves

The well-intentioned programmer writes same code **no matter what policies are**.

Programming model.

Mathematical guarantees.



Implementations, web framework, and case studies.

Jeeves Factors Out Policies

```python
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

```html
<h2 class="form-heading">Edit Your Event.</h2>
 <div class="form-group">
  <label for="name" class="control-label"><a rel="tooltip" title="TO
  <div class="ro
    class="
    <input typ "text" l="form-control na="name" ne" ue="{{ 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="ro
    <div class="
    <input typ "text" l="form-control name="location" id="location" value="{{ concretize
   </div>
  </div>
 </div>
 <div class="form-group">
  <label for="time" class="control-label"><a rel="tooltip" title="TODO."><span class="glyphicon gl
  <div class="ro
    <div class="
     <input typ "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="ro
    <div class="
     <textarea m-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="ro
    <div class="
     <div class="btn-group">
      <button type="button" class="btn btn-default {% if visibility=='E' %}active{% endif %}" dat
      <button type="button" class="btn btn-default {% if visibility=='G' %}active{% endif %}" da
     </div>
    </div>
  </div>
 </div>
<button class="btn n-lg" type="submit" value="Submit" onclick="myApp.showPleaseWait();">Submi
</form>
```

View

```python
@login_required
@request_wrapper
@jeeves
def profile(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.
- Policy-agnostic program.
- Runtime differentiates behavior.

Background.

Automatically Managing Information Flow

Jeeves supports more expressive policies than prior work.

1. Specifying policies.
2. Specifying differentiated behavior.
3. Factoring out policies from other code.
4. Enforcing policies that depend on sensitive values.

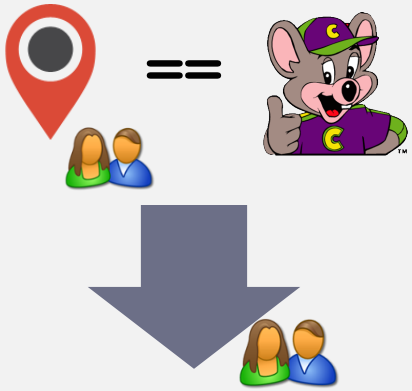
Only Jeeves supports 3 and 4.

Related Work:
Checking Information Flow

Programmer tags data.

Must explicitly define and tag alternate behavior.

```
if viewer ==
```



System prevents leaks by raising error.

true **error**

```
elif viewer !=
```

false

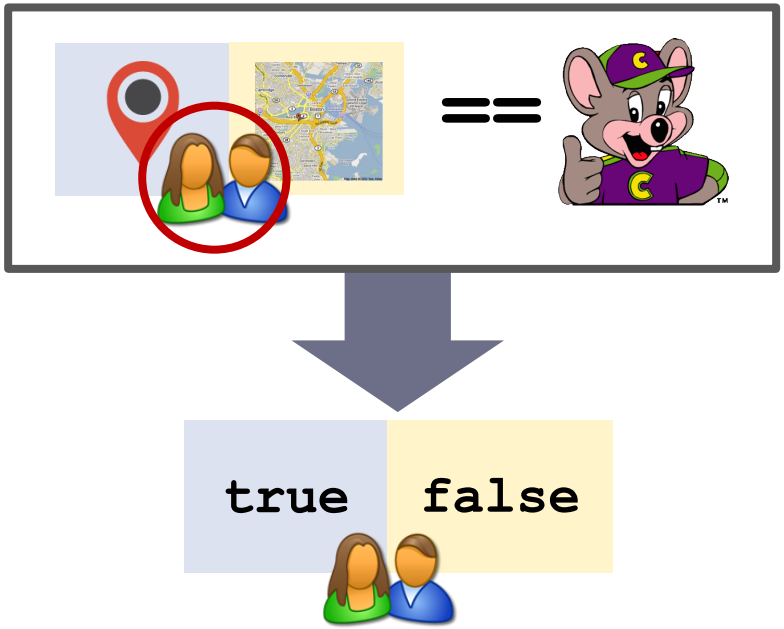
Information flow controls check that only allowed flows occur [HiStar, Flume, Resin, LIO/Hails, Jif/Sif, flow locks, F*, and more].

My previous work.

Related Work:
Computing with Multiple Values

Must encode policies as tags.

- "Policy spaghetti."
- Policies can leak information.



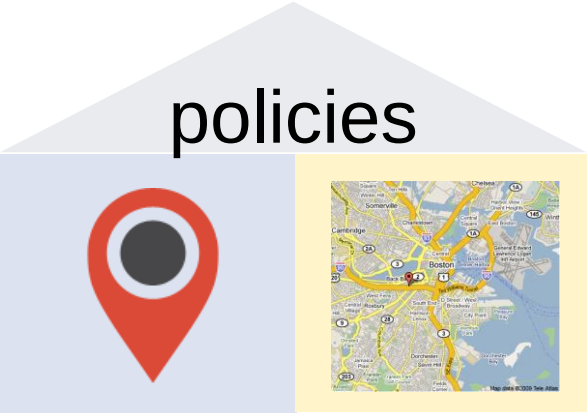
Multi-execution approaches differentiate behavior based on viewer [secure multi-execution (Devriese et al); faceted execution (Austin and Flanagan)].

Jeeves programming model.

Challenges

1. Factoring out information flow.
2. Supporting expressive policies that can depend on sensitive values.

Policy-Agnostic Programming in Jeeves



Policies describe rules for how values may flow to output contexts.

Sensitive values encapsulate multiple behaviors.

Application Code

Separate from policies.

Jeeves Supports Expressive Policies

A policy is an arbitrary function that takes the output context and returns a Boolean value.

```
def isNotCarol(oc): return oc != 👰
```

Output context can be of arbitrary type.

Policies can depend on sensitive values.

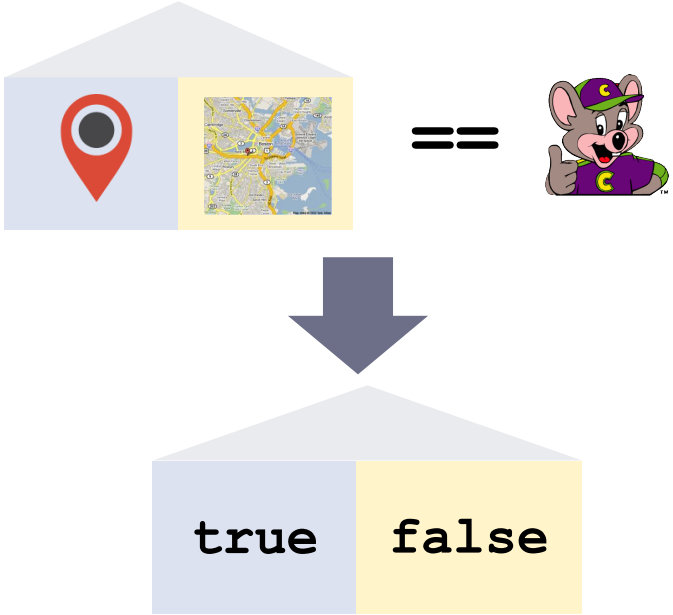
```
def isGuest(oc): return oc in 🎉.guests
```

🎉.guests [ ]

Jeeves Programming Model

1 Programmer specifies **policies** and **facets**.

2 Programmer writes **policy-agnostic** programs.



3 Runtime propagates values and policies.

4 Runtime produces **differentiated output** based on the viewer.

The Jeeves Programming Model

- Well-defined **runtime semantics** for policy-agnostic programming with information flow policies.
- Can be implemented **standalone** or embedded **as a library**.
- Has been adapted **across runtimes** in **web frameworks**.

How Jeeves works.

Challenges

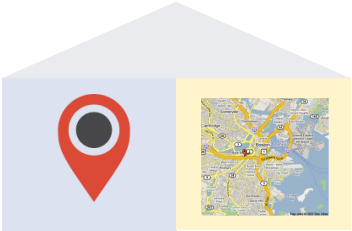
1. Handling policies with dependencies.
2. Enforcing policies when state changes.

Jeeves Execution Model

Runtime simulates simultaneous multiple executions.

1 Runtime propagates values and policies.

```
x = 0
```

if  ==  :

```
    x += 1
return x
```

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

print {  }

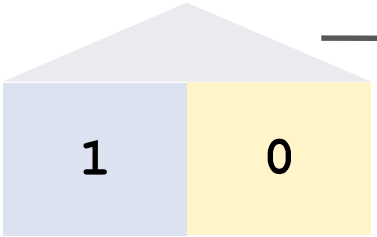
1

print {  }

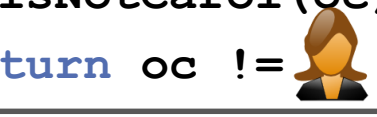
0

Using Policies to Produce Outputs

Jeeves uses policies to defacet appropriately.




```
def isNotCarol(oc):
  return oc !=
```



```
print {
```

```
}
```

```
policy(
```

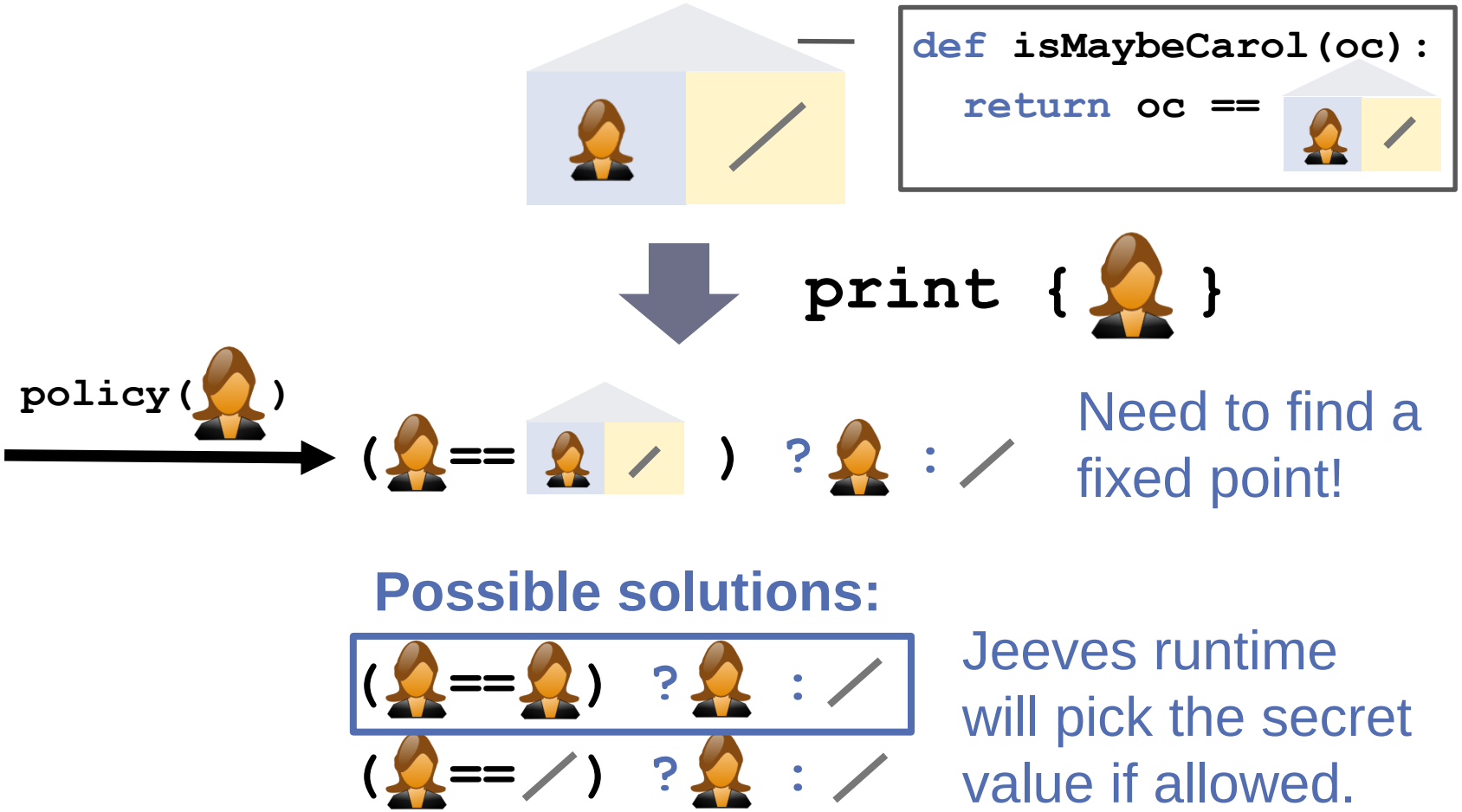
```
)
```


```
!=
```

```
) ? 1 : 0
```

```
0
```

But What About Dependencies?



Using Constraints to Handle Dependencies

Evaluated with respect to state at time of output.

a	
1	0

Label	Policy
a	`def isGuest(oc):` `  return oc in 🎉.guests`

$a \in \{secret, public\}$

`print {` 🤐 `}`

`policy(` 🤐 `)`

$a = secret \Rightarrow$ 🤐 `in` 🎉`.guests`

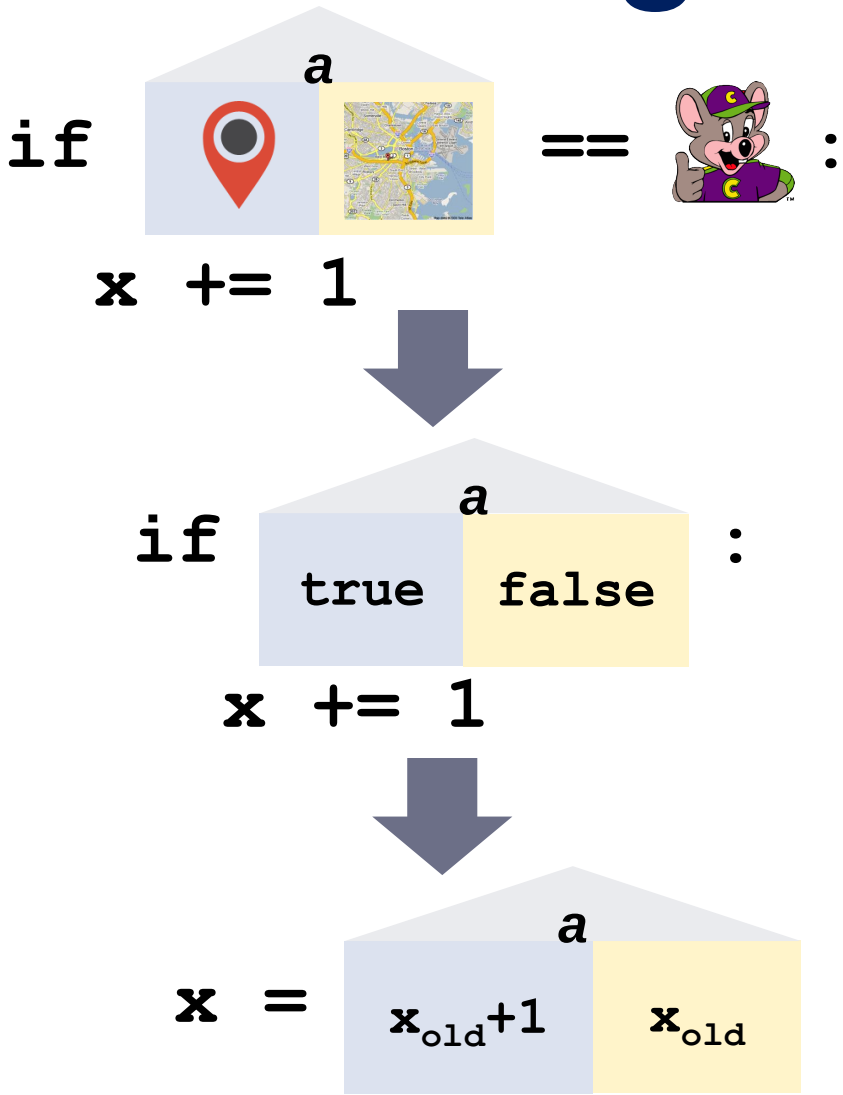
$\vdash \; a = secret \Rightarrow false$

$\vdash \; \neg(a = secret)$

0

- Constraints contain only Boolean variables.
- Always a consistent assignment.

Handling Indirect Flows



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

Jeeves Runtime Semantics: Outputting to Sinks

Statement evaluation $\Sigma, S \Downarrow V_p, oc: R$

Expression evaluation $\Sigma, E \Downarrow_{pc} \Sigma', V$

$$\Sigma, E_{oc} \Downarrow_{\emptyset} \Sigma_{oc}, V_{oc}$$
$$\Sigma_{oc}, E_r \Downarrow_{\emptyset} \Sigma_r, V_r$$

Evaluate output context and expression to print.

$$\{k_1 \ldots k_n\} = closeK(labels(E_{oc}) \cup labels(E_r), \Sigma_2)$$
$$E_p = \lambda x.true \wedge_f \ldots \wedge_f \Sigma_2(k_n)$$

Retrieve labels and policies.

$$\Sigma_r, (E_p\ V_{oc}) \Downarrow_{\emptyset} \Sigma_p, V_p$$

Evaluate policies applied to the output context.

$$\frac{\text{pick } pc \text{ such that } pc(V_{oc}) = oc, pc(V_r) = R, \boxed{pc(V_p) = true}}{\Sigma, \text{print}\ \{E_{oc}\}\ E_r \Downarrow V_p, oc: R}$$

Policies

Output context

Result

Defacet using satisfying policy assignment.

Challenges

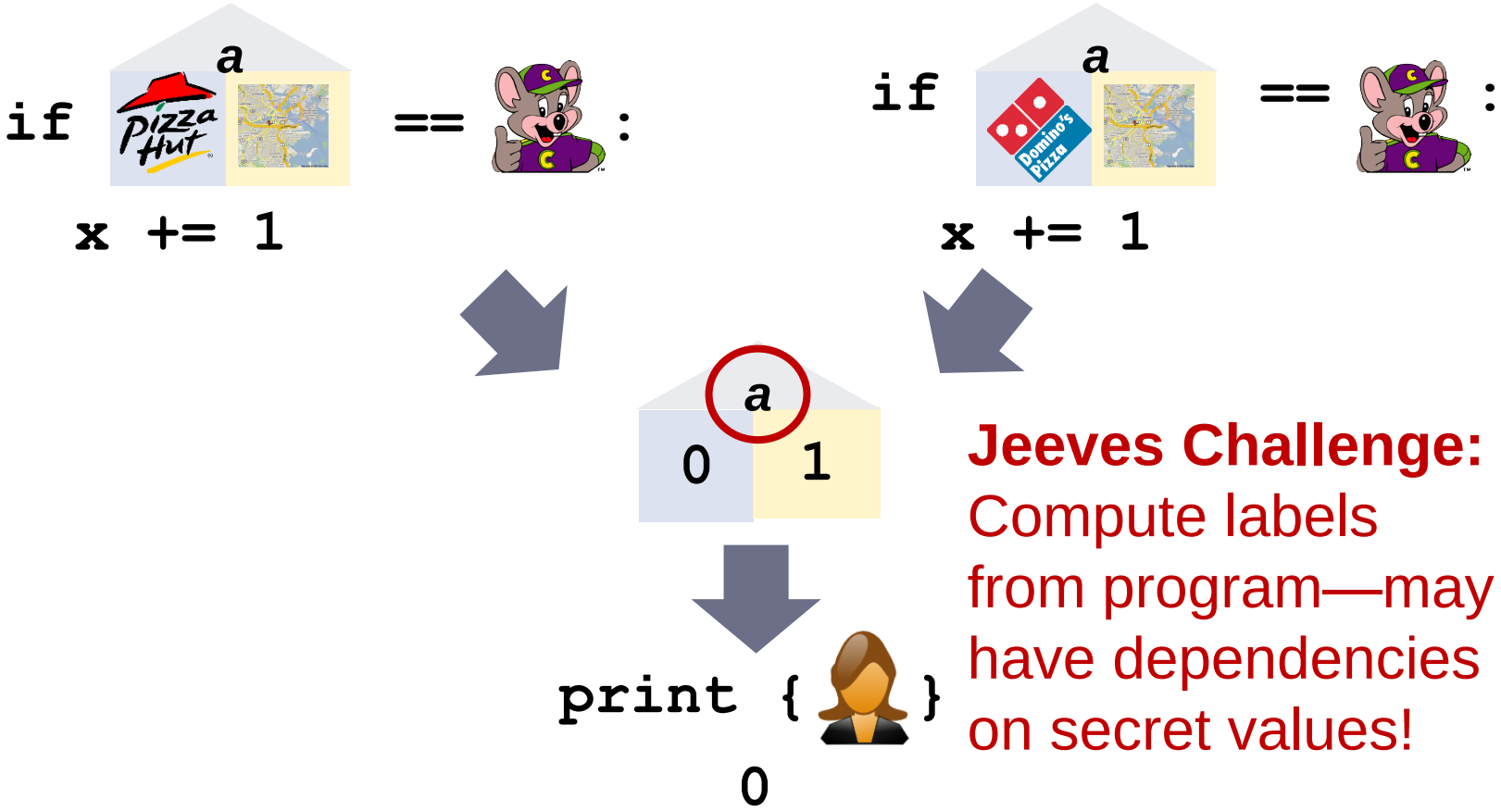
1. Defining what it means to enforce policies.

Theoretical guarantees.

2. Proving enforcement with respect to our semantics.

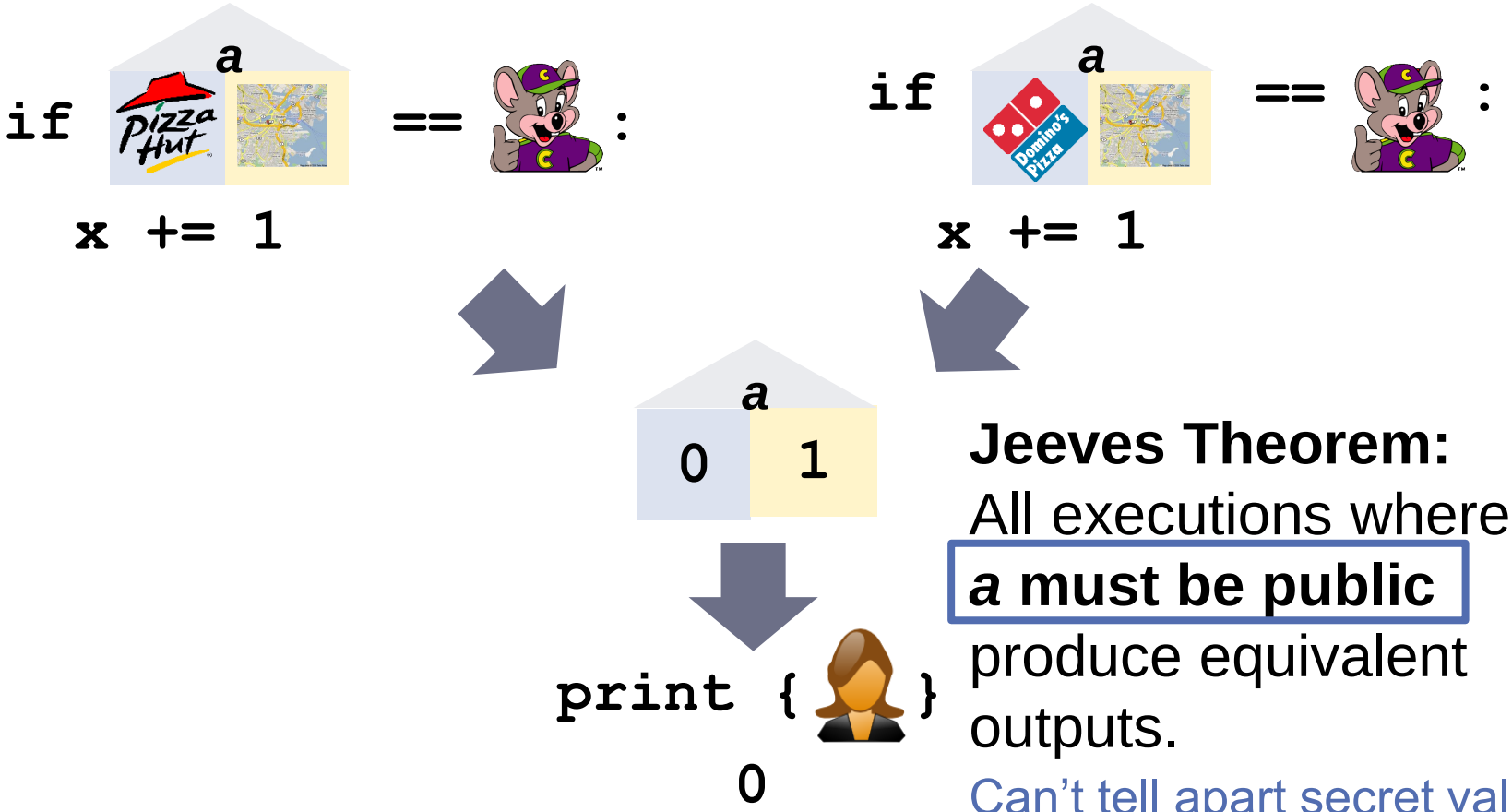
Non-Interference

Secret values should not affect public output.



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

Jeeves Non-Interference and Policy Compliance



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

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

Jeeves Policy Compliance Theorem

Suppose we have

$S_1 = print\ \{e\}\ C[\ E_1\ {}^{a}\ E_P\ ]$ $\quad$ $S_2 = print\ \{e\}\ C[\ E_2\ {}^{a}\ E_P\ ]$

Where a must be public.

Then the set of values S_1 evaluates to is equivalent to the set of values S_2 evaluates to.

$$\{f : R \mid \exists V.\ \emptyset, S_1 \Downarrow V, f : R\} = \{f : R \mid \exists V.\ \emptyset, S_2 \Downarrow V, f : R\}$$

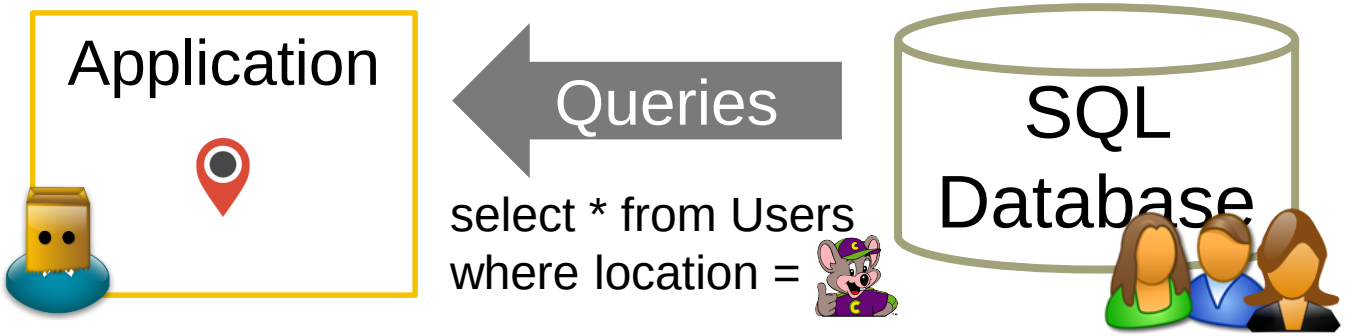
Need to account for policies that depend on sensitive values!

Challenges

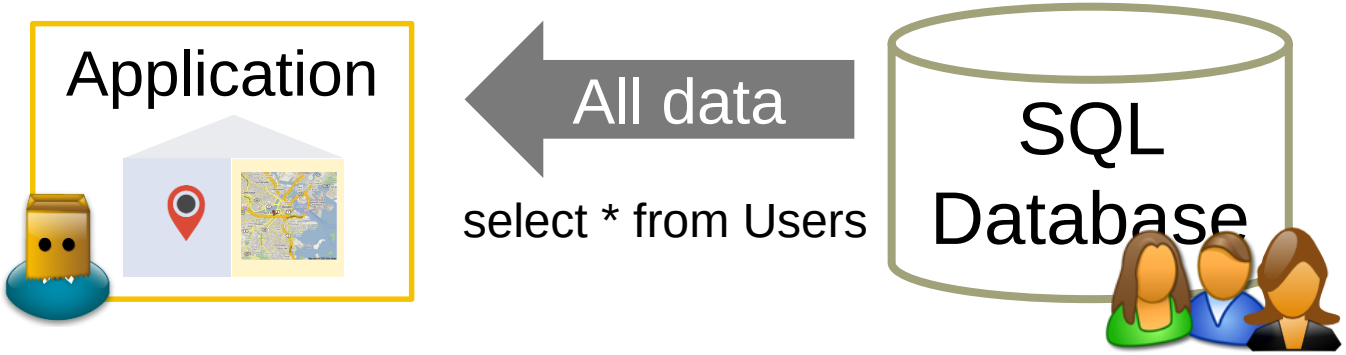
1. Adapting the approach for web applications.
2. Scaling the approach.

Implementation and evaluation.

Facets in the Application and Database



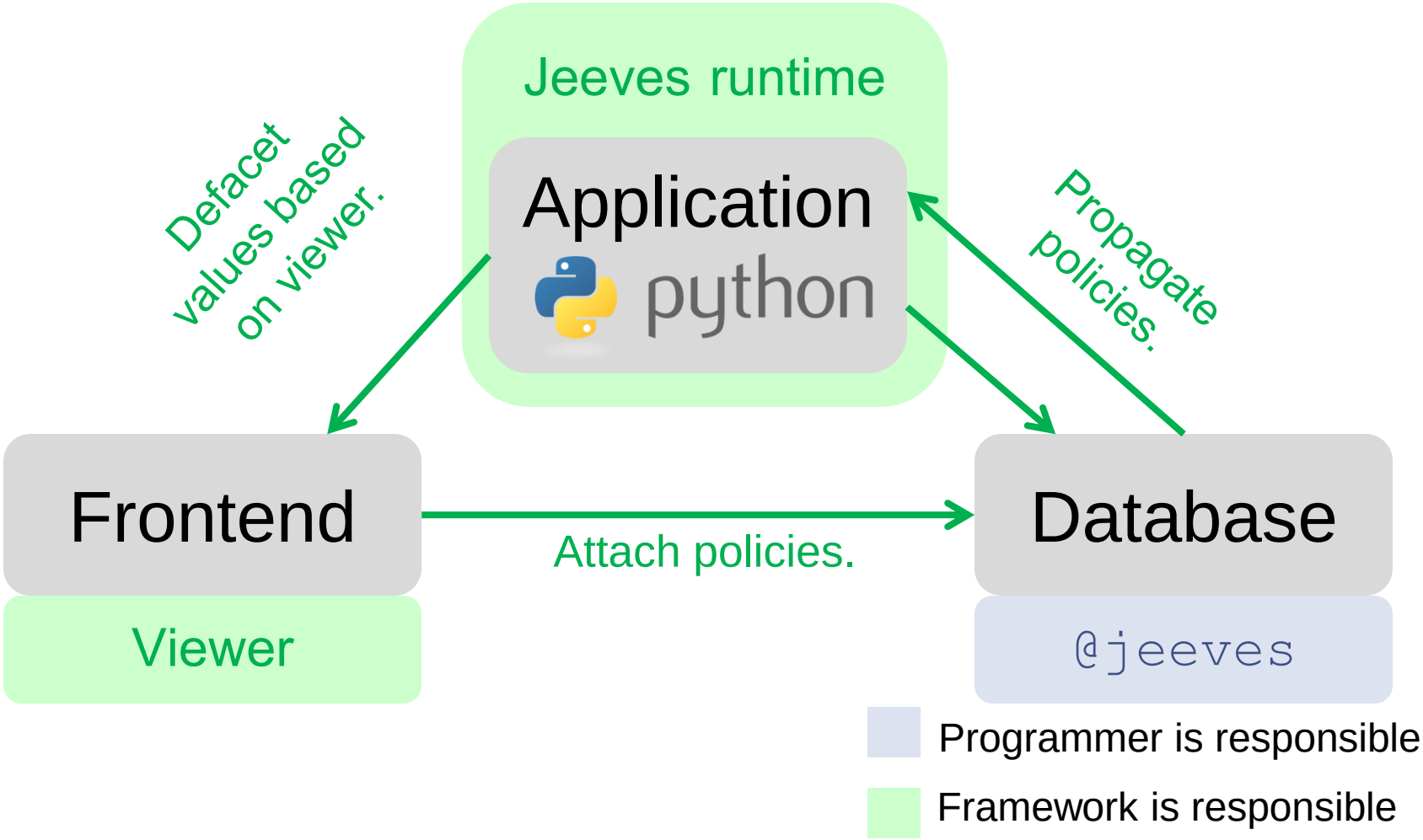
Database queries can leak information!



Impractical and potentially slow!

Solution: Facet the database. Implemented as an object-relational mapping supporting uniform facet representation.

Jacqueline Web Framework



```python
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')
```

Django-like data schema for describing fields.

```python
@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
```

Helper functions for policy include queries.

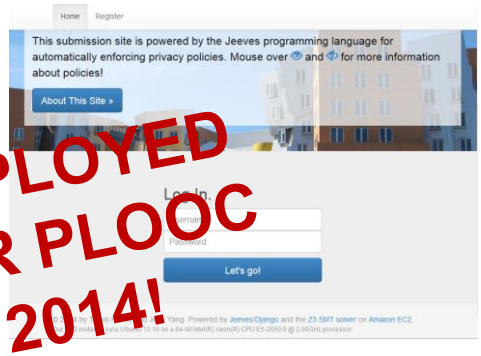
```python
@staticmethod
@label_for('name', 'location', 'time', 'description')
@jeeves
def jeeves_restrict_event(event, ctxt):
    if event.visibility == 'G':
        return event.has_host(ctxt) or event.has_guest(ctxt)
    else:
        return True
```

Policy for ‘location’ field.

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

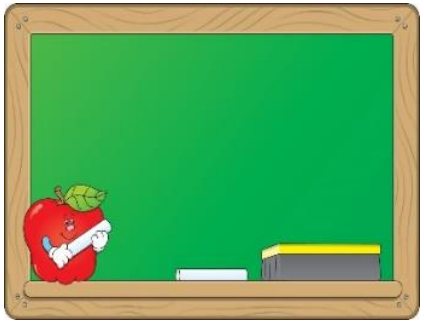
Public value for ‘location’ field.

Implemented in Jeeves



DEPLOYED FOR PLOOC 2014!

Conference management system



Course manager

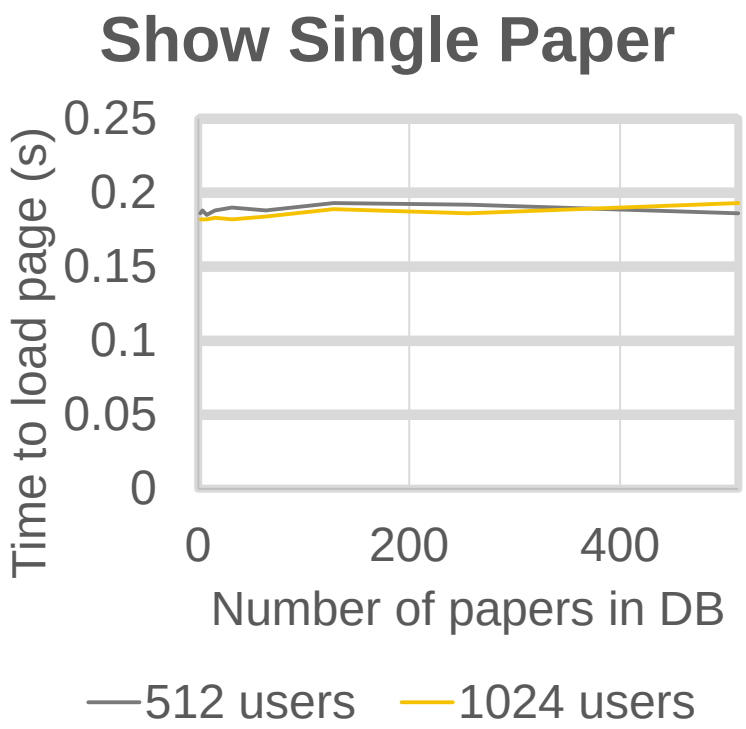
Medical records (HIPAA)

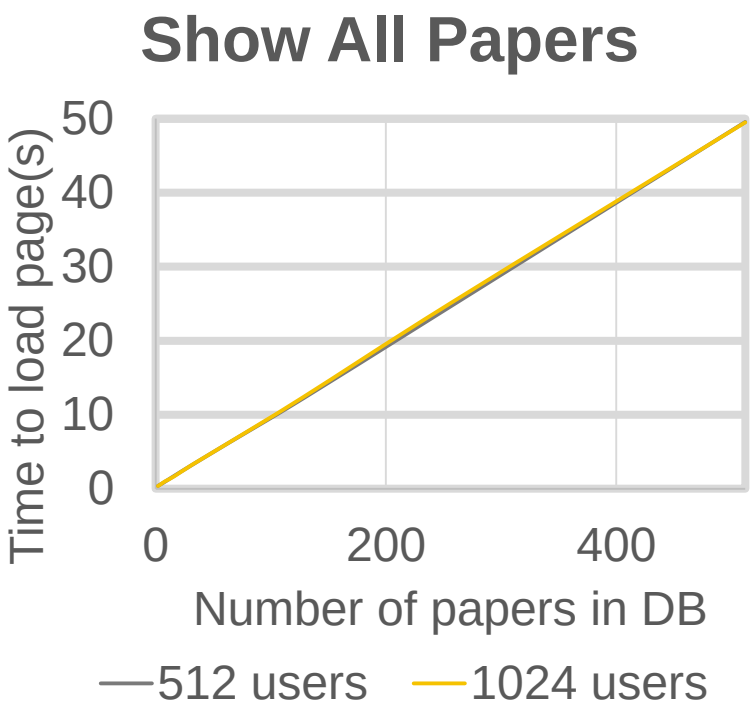
Questions

1. How does faceted execution scale?
2. How does faceting the database affect queries?

Demo

CMS Running Times





Database size does not affect query speed.

Showing sensitive data grows linearly.

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

Evaluation Lessons

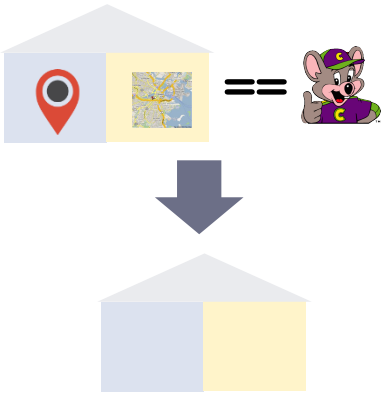
- Faceted execution is expensive.
- Policy-agnostic programming does not always need faceted execution.
- Many opportunities for optimization in web frameworks.

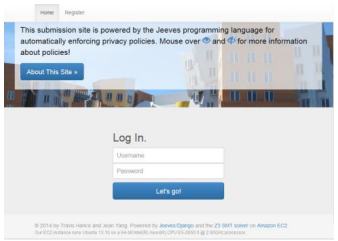
Policy-Agnostic Programming in Jeeves

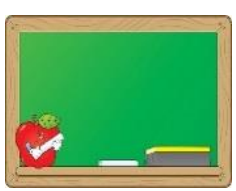
Sensitive values

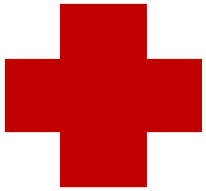
Policies

Other functionality







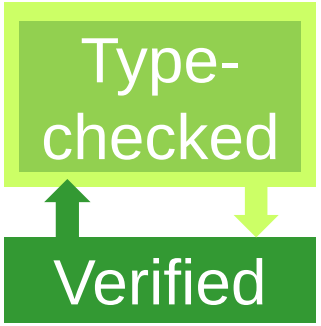


Design of a policy-agnostic programming language **[POPL '12]**

Semantics and guarantees **[PLAS '13]**

Implementation, web framework, and case studies **[in submission]**

My Interests in Provable Guarantees



Verve: a provably type-safe OS.
[PLDI '10 (Best Paper); CACM '10]

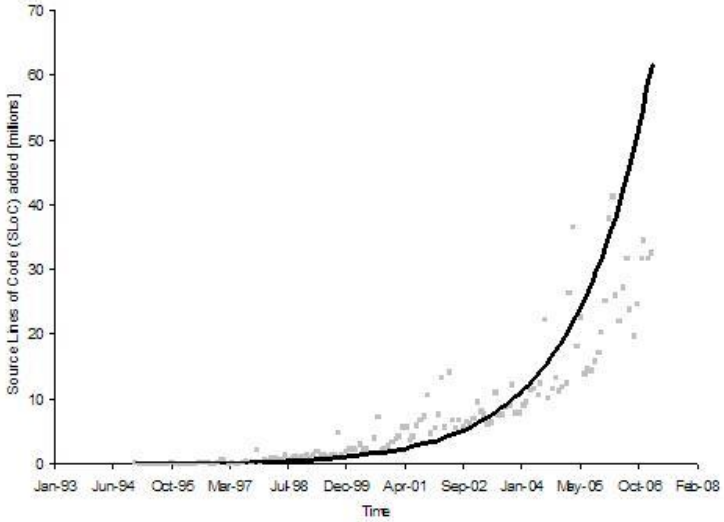
F*: type-checking security properties.
[ICFP '11; JFP '13]

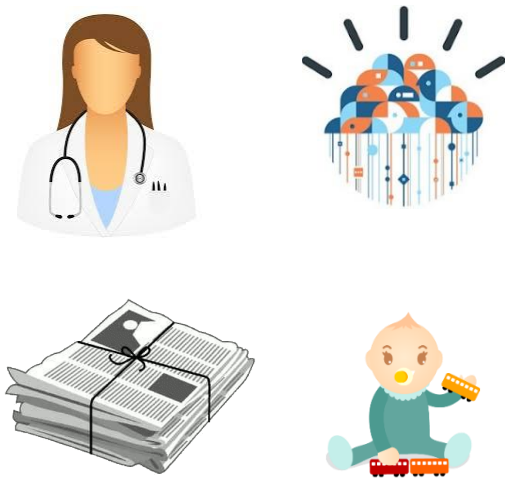
Ask Reeves: verified privacy policies.
[patent filed]

Lessons

- Difficult to write secure and private programs.
- Even more difficult to prove them correct.
- Easier if we can get privacy by construction.

The Future of Policy-Agnostic Programming

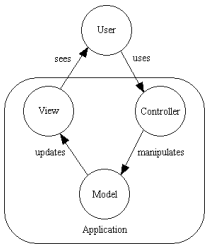




- Factor out policies from other code.
- Rely on compiler and runtime to differentiate behavior.

Making Policy-Agnostic Programming Work

- **Programmability.**
  - Languages for different kinds of policies.
  - Tools for verifying and visualizing policies.
- **Scaling.**
  - Policy-agnostic semantics with minimal faceting.
  - Building foundations with policy management in mind.



Web frameworks

Databases

Web browsers

Parting Thoughts

By **reducing opportunity** for programmer error, we can **eliminate** whole classes of **privacy leaks**.