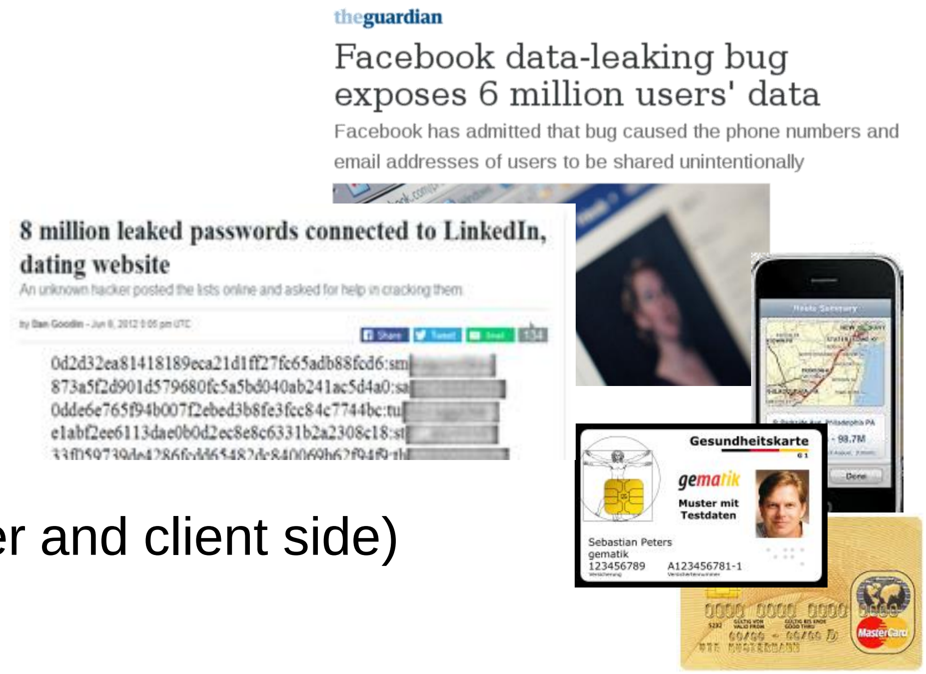
Security in Web-based Workflows

Thomas Bauereiß, Abhishek Bichhawat, Iulia Bolosteanu, Peter Faymonville, Bernd Finkbeiner, Deepak Garg, Richard Gay, Sergey Grebenshchikov, Christian Hammer, Dieter Hutter, Ondrej Kuncar, Peter Lammich, Heiko Mantel, Christian Müller, Andrei Popescu, Markus Rabe, Vineet Rajani, Helmut Seidl, Markus Tasch, Leander Tentrup

German Research Center for Artificial Intelligence
Max Planck Institute for Software Systems
Middlesex University London
Universität des Saarlandes
Technische Universität Darmstadt
Technische Universität München
Universität Bremen

Web-Based Workflows

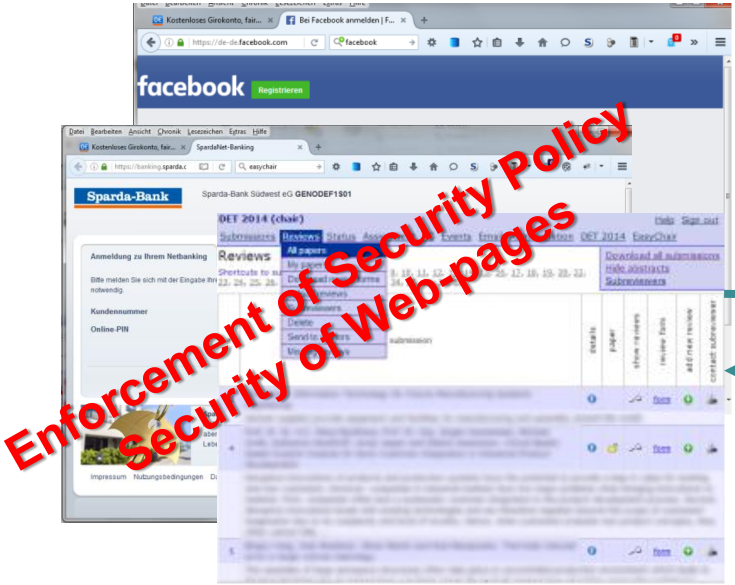
- Workflows ubiquitous on the Web
  - (e.g. Amazon, PayPal, Facebook, EasyChair, …)
- Processing sensitive data, e.g.
  - credit card details
  - health-related data
  - location information
  - private messages
  - …



- Goal: Holistic security (both server and client side)
- Challenges
  - Complex security requirements, information flow control
  - Heterogeneous system models, security notions, languages

Web-based Applications

Client



Internet

Server

© Google

Heterogeneous system models, notions of security, languages

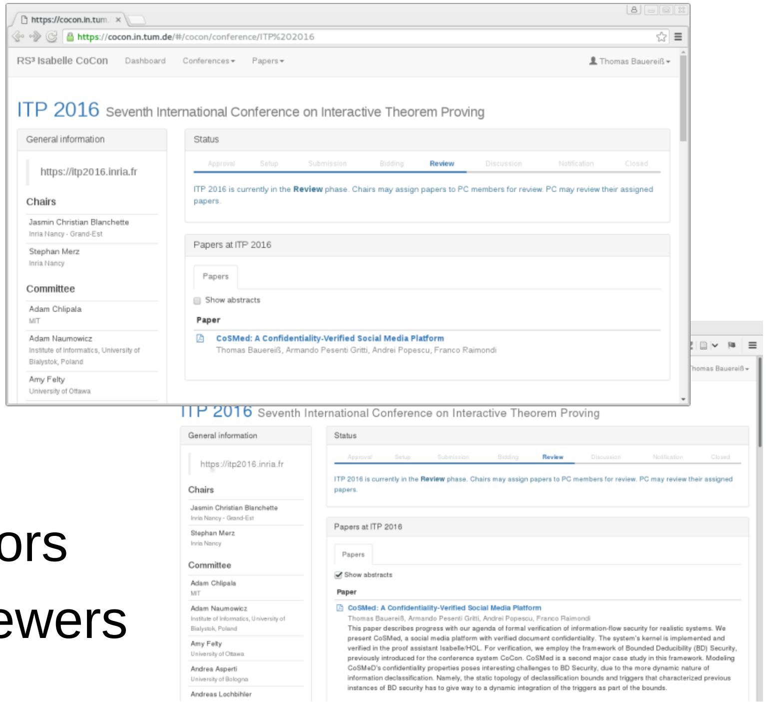
Application: CoCON

- A conference system with verified document confidentiality

CoCon VERIFIED

used for TABLEAUX '15 and ITP '16

- Confidentiality of
  - Paper Content
  - Reviews
  - Discussions
  - Decisions
  - Reviewer Assignments



- Confidentiality of information changes
  - Last version of review is sent to authors
  - Last version of paper is given to reviewers

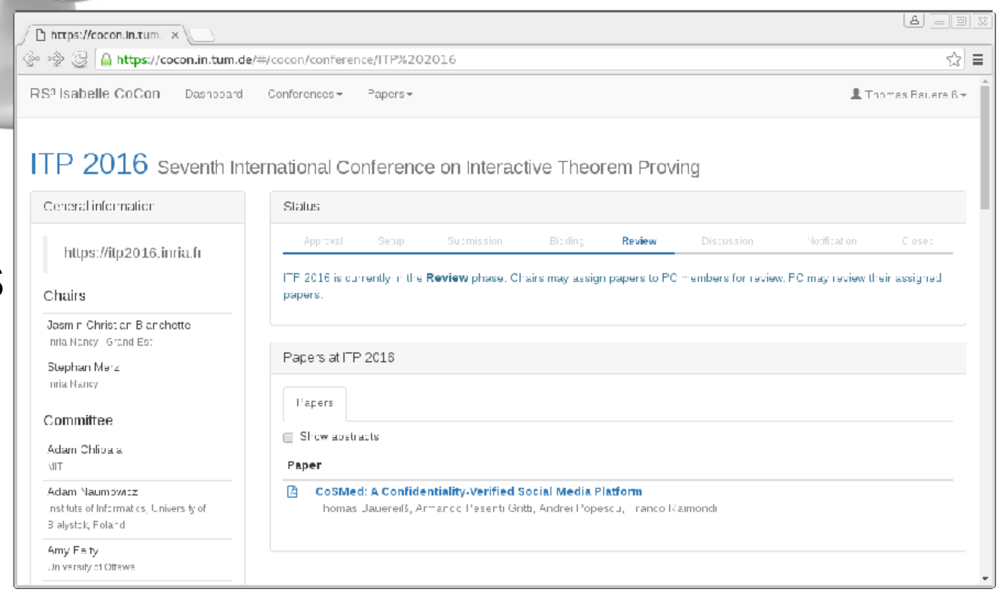
Admissible Information Flows

Information	Role	Restrictions
Paper Content	Author	no restrictions
	PC member	Last uploaded version
Review	Reviewer	no restrictions
	non-conflict PC member	Last edited version before discussion and all the later versions
	PC member or paper author	Last edited version before notification
Discussion	Non-conflict PC member	no restrictions
Decision	Non-conflict PC member	no restrictions
	PC member or author	Last edited version
Reviewer Assignment	Non-conflict PC member	no restrictions
	Author	Number of reviewers

The Client

- Web interface to enter
  - Papers (as a autor)
  - Reviews (as a PC-member)
  - Discussions (as a PC-member)
  - News (as a PC-member)

Monitoring information flow in the browser



- Threats
  - E.g. reviews are leaked into a discussion or news
- Security Policy
  - Provided by the server

Security Enforcement - Client

- Information flow control inside the browser
- E.g. news sent to the server must not depend on review
- Program analysis:
  - How is the output (news) constructed?
  - Does its computation use the review?
- Example:

```
news = "Reviews nearly finished"
if (review[78] == accept)
    news = "Reviewers did a great job"
```

Value of `news` *(public) is computed with the help of* `review[78]` *(confidential)*

Indirect information flow

- Information flow depends on execution model (covert channels)
  - Event based: concurrency / interleaving, blocking / non-blocking
  - DOM based: live collections

Support For Event Handling

- The capture phase executes all capture and target handlers associated with all nodes from the root to the target's parent, starting from the root.
- The target phase executes all the handlers associated with the target.
- The bubble phase executes all target and bubble handlers associated with all nodes from the target's parent to the root, starting from the target's parent.

```
var p = document.getElementById('para');
p.onclick = function() {
    alert('In click');
    p.innerHTML += 'click';
};
window.onresize = function() {
    p.innerHTML = 'resize ';
};
```

Example: preemption (while one API call is waiting for user input the execution of another API call is scheduled/executed)

```
function foo() {
   ...
   pub = true;
   if (sec)
      preemption-point
   ...
   pub = false;
}

function bar() {
   conf = pub;
}
```

Implicit leak via preemption

Security Enforcement (Client)

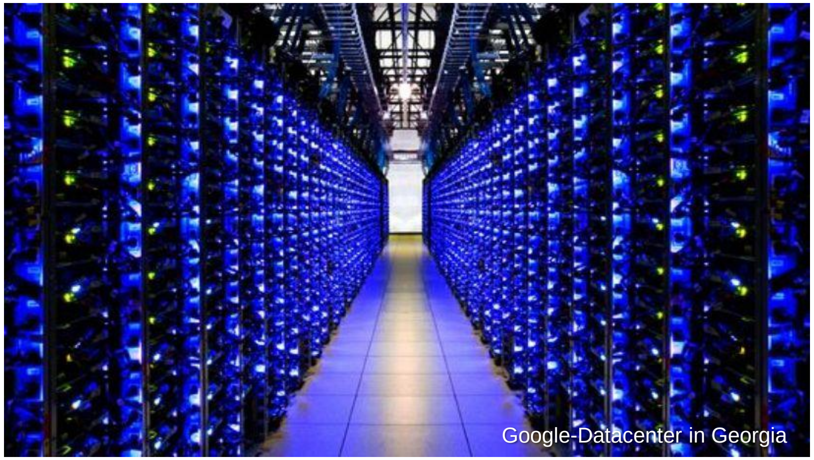
Covering the leaks caused by

- Handler preemption
- Event phases
- Live collections
- Browser optimizations

Runtime Enforcement

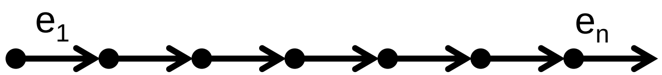
- No illegal input is used to compute a specific output
- “Source-Sink” relation is provided by the server
- Sound enforcement of Reactive Noninterference
- Full implementation for Safari/WebKit

Modeling the Server of CoCON



© Google

- Database maintaining papers, reviews, news, …
- IO-automata (event-based system)
  - An event is an atomic operation
  - A trace τ is a list of events $e_1 \dots e_n$

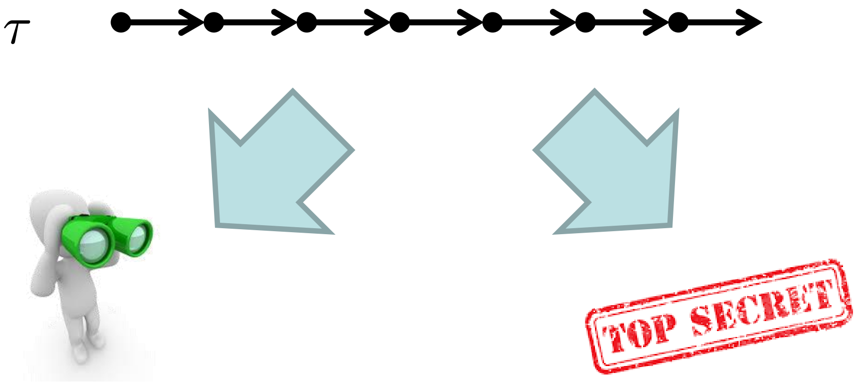


  - A system (behavior) is a set of traces
    (i.e. each of these traces correspond to a run of the system
  - There are special Input/Output-events for composing automata
- Model is automatically translated to SCALA (ISABELLE)

Trace-Based Information Flow Control

- System behavior is given as a set of traces $\tau \in S$

Trace τ



Observations $o(\tau)$

Secrets $v(\tau)$

- Typically, o and v are homomorphic extensions of corresponding functions on events: i.e. $o(e \cdot \tau) = \phi(e, o(\tau))$

BD-Security and Declassification

Security means that each observation can be caused by various traces (with different secrets)

i.e. an observation $o(\tau)$ must be equal to observations $o(\tau')$, $o(\tau'')$… caused by other traces τ', τ'' …
… but these traces have different secrets: $v(\tau) \neq v(\tau') \neq v(\tau'')$

Information flow control denotes a closure property of S:

“ if there is a trace $\tau \in S$ …

… then there are other traces $\tau' \in S$ with (specific) different secrets but causing the same observations ”

BD-Security – The Formalities

- An observer cannot learn anything about the secret if its observations o(τ) does not provide any clues about possible secrets v(τ) :

$$\forall\ \tau \in S.\ \forall\ v \in V.\ \exists\ \tau' \in S.\ \ o(\tau) = o(\tau) \wedge v(\tau') = v$$

- An observer cannot learn partly about the secret if

$$\forall\ \tau \in S.\ \forall\ v \in V.\ \exists\ \tau' \in S.$$
$$B(v(\tau), v) \Rightarrow o(\tau) = o(\tau) \wedge v(\tau') = v$$

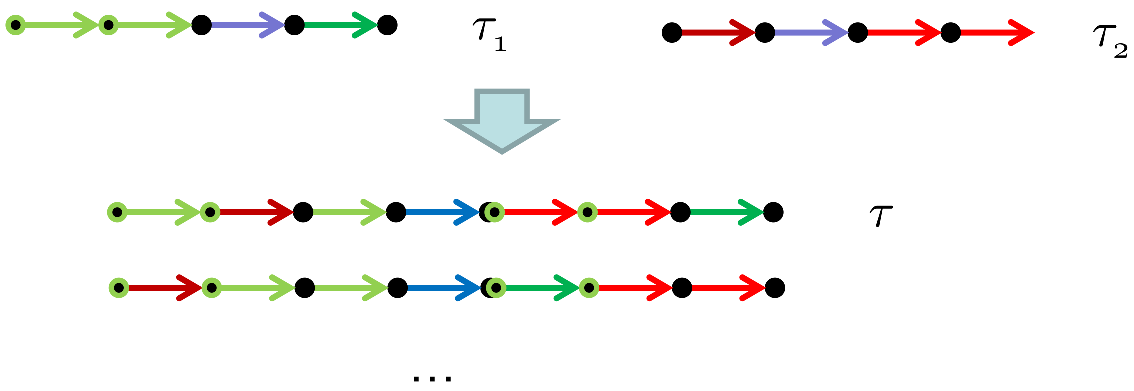
declassification

Verification of BD-Security

- Unwinding proofs (schematic inductive proofs)
- Machine generated
  - BD-security properties are formalized and verified by Isabelle/HOL (interactive higher order theorem prover)
  - Ongoing work: modeling the properties as HyperCTL*-properties and doing model checking
- Verification in the large
  - Structured specification & composition theorems

BD-Security and Composition

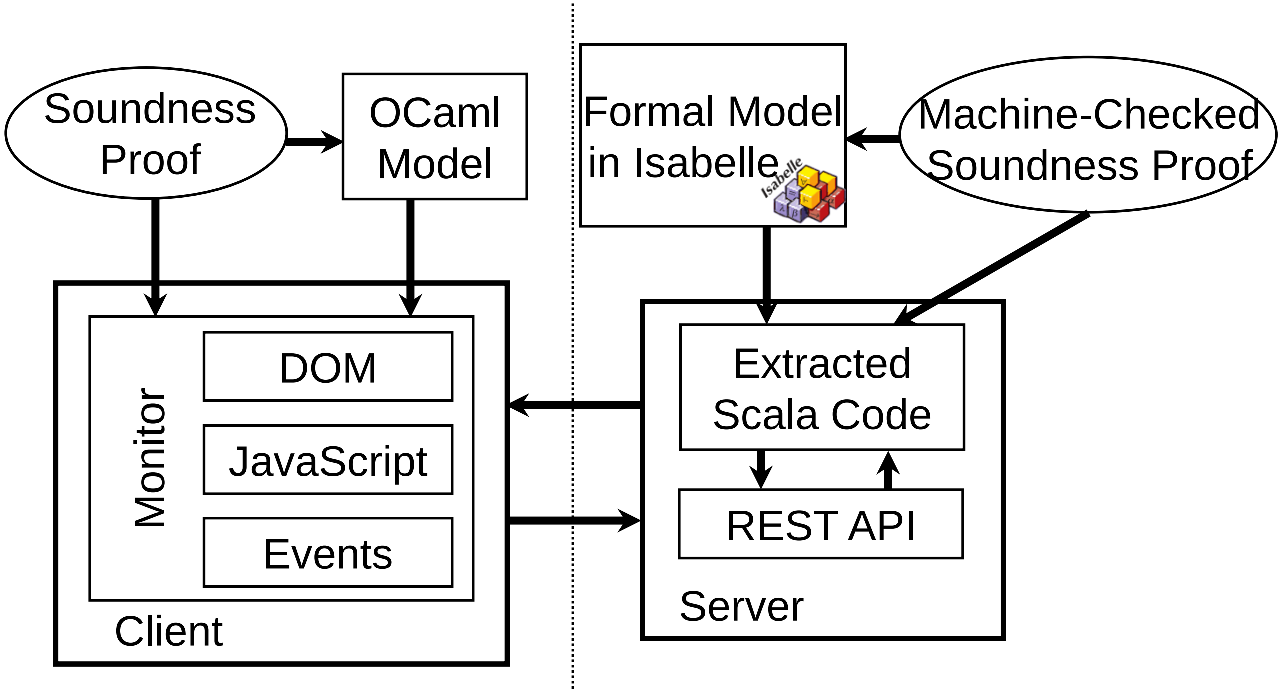
- Composition of IO-automata via shared events and interleaving of traces



- Composition of secure IO-automata is not secure in general
- Development of conditions to ensure secure composition
- Refinement as a special case of composition
- Encoding safety-properties (e.g. Separation of Duty) as composition problems

Approach

- Formal verification of server application using a theorem prover
- Runtime monitoring of untrusted client-side JavaScript
- Policy for JavaScript code sent from server to client



Future Work

- Formal end-to-end security guarantee integrating client- and server-side properties

- More compositionality results for Bounded Deducibility Security

- Improved automation of proofs by integrating model checking
  - Sound abstraction to finite-state verification problem
  - Integration of model checker for HyperCTL*

Summary

- Client: Runtime Enforcement
  - Sound enforcement of Reactive Noninterference
  - Full implementation for Safari/WebKit
- Server: Formal Verification
  - Case study: Conference management system (CoCon)
  - Novel security notion: Noninterference-like property with declassification support (Bounded Deducibility Security)
  - Mechanically verified properties, for example ``authors learn nothing about reviews beyond the last version after notification''
  - CoCon used for TABLEAUX '15 and ITP '16