Hybrid Information Flow monitoring against Web tracking

Frederic Besson, Nataliia Bielova, Thomas Jensen

Inria, France

26th IEEE Computer Security Foundations Symposium

June 26-28, 2013

Alexa-top 10,000 sites [Nikiforakis et al. 12]

- 88.45% of sites have at least one remote JavaScript
- per site: up to 295 remote JavaScript

How can they track me?

- Stateful tracking: well-known and getting addressed
  - Third-party cookies blocking
  - Non-interference for JavaScript
  - EU e-Privacy directive

[Austin, Flanagan 12]
[De Groef et al. 12]
[Hedin, Sabelfeld 12]

- Stateless tracking: not addressed
  - IP address tracking
  - Web browser fingerprinting

[Eckersley'10]

Panopticlick

How Unique – and Trackable – Is Your Browser?

Your browser fingerprint **appears to be unique** among the 2,419,678 tested so far.

Currently, we estimate that your browser has a fingerprint that conveys **at least 21.21 bits of identifying information.**

- Information needed to **uniquely identify a browser**
  - *n* – number of connected devices: 5 000 000 000
  - $log_2 n$ – number of bits for a unique id: 33 bits

- **Idea: distinguish** users **by browser fingerprints**:
  - HTTP headers
  - Browser and OS features: language, plugins, fonts, screen, …

The most identifying features (via JavaScript and Flash)

Some scripts are useful

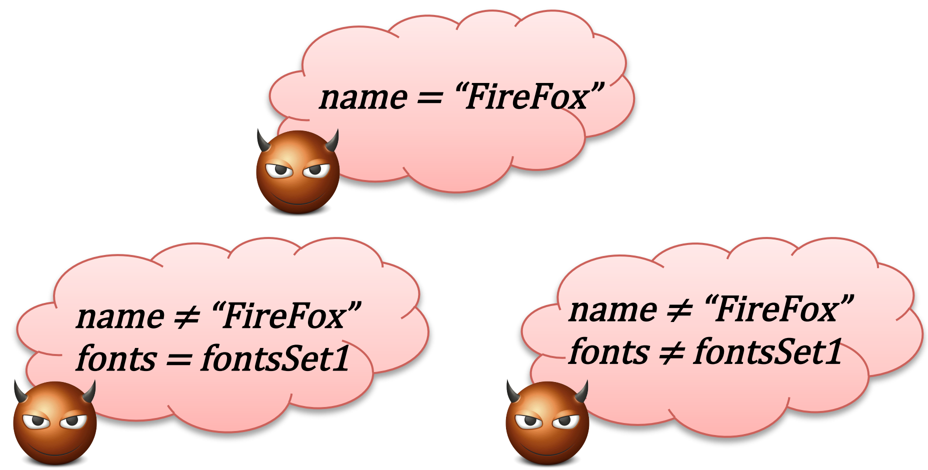
```
var x = 0;
if (name == "FireFox") {
     x = 1;
}
output x;
```

`name:` browser name
`output x:` request containing `x` sent

Non-interference is too restrictive:
`x` depends on name

What does tracker learn?

```
var x = 0;
if (name == "FireFox") {
     x = 1;
}
else {
     if (fonts == fontsSet1) {
          x = 2;
     }
}
output x;
```



Depending on user's browser, **different executions** of this script **leak different quantity** of information!

Quantitative information flow

- Traditional model:
  - Decrease in uncertainty: entropy-based [Smith'09]
  - Increase in accuracy: belief-based [Clarkson, Myers, Schneider'07]

- Traditional analysis:
  - Static analysis for all program executions
    [Clark, Hunt, Malacaria'07] [Mardziel, Magill, Hicks, Srivatsa'11]

- Our approach:
  - Monitor **one program execution** and **quantify leakage**

Quantification of leakage

- **Self-information, or "surprisal"**
  - "amount of information about the identity" [Eckersley'10]
  - = beliefs for deterministic programs [Clarkson, Myers, Schneider'07]

$$I(A) = -\log_2 P(A)$$

```
var x = 0;
if (name == "FireFox"){
     x = 1;
}
output x;
```

Popularity of "FireFox" is 21%

$I\,(name = \text{"FireFox"}) = -\log_2 0.21 = 2.25$ bits

$I\,(name \neq \text{"FireFox"}) = -\log_2 0.79 = 0.34$ bits

- **Entropy-based definition = average leakage for all browsers!**

$H(name) - H(name \mid x) = 0.74$ bits

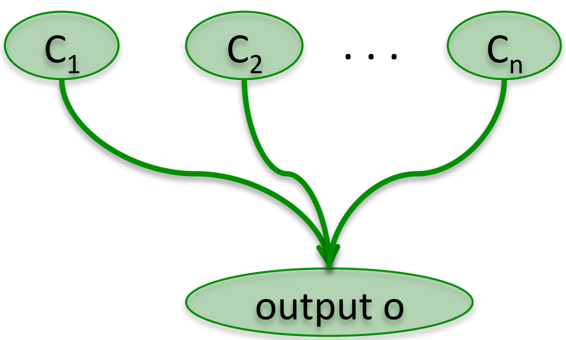
The rest of this talk

- Hybrid monitoring for quantitative information flow
  - Knowledge representation
  - Labeling propagation
- Soundness and precision
- Hierarchy of hybrid monitors ordered by precision

Knowledge of tracker: configurations

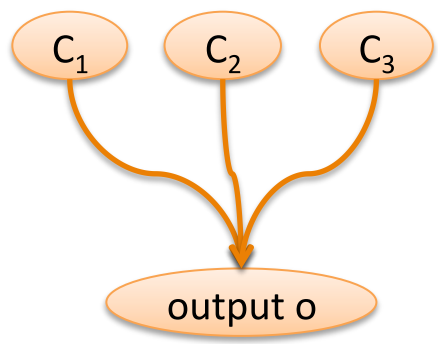
- Browser configuration $C : Features \rightarrow Val$
- $Features = \{name, fonts, \ldots\}$ and $C(name) = \text{“FireFox”}$
- Leakage by **self-information**: $I(A) = -\log_2 P(A)$

Noninterference
All configurations



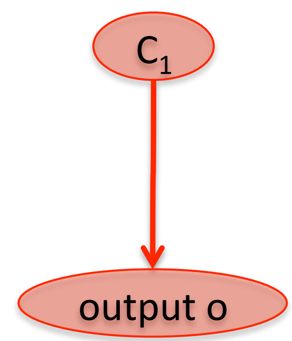
$-\log_2(P(C_1) + \ldots + P(C_n)) = -\log_2 1 = 0$ *bits*

Partial leakage
Some configurations



$-\log_2(P(C_1) + P(C_2) + P(C_3))$ *bits*

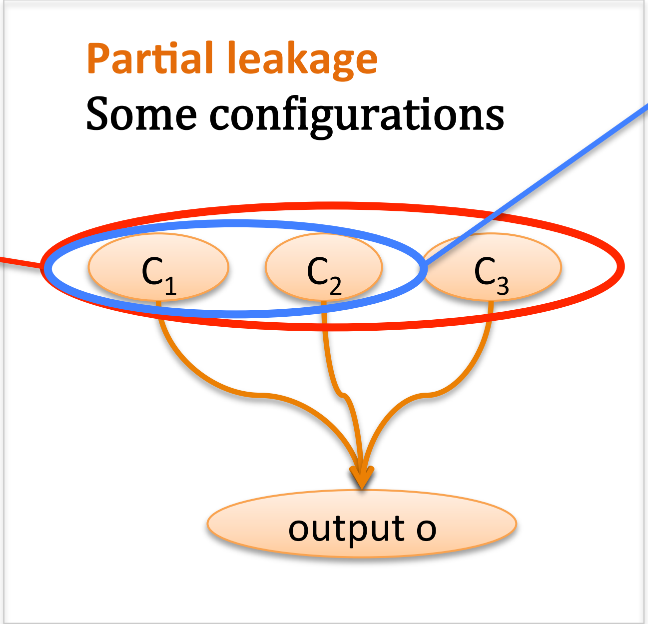
Complete leakage
One configuration



$-\log_2 P(C_1)$ *bits*

Knowledge of tracker: configurations

Actual knowledge of tracker is a set of equivalent configurations *Eq(P,C)*



We over-approximate knowledge by a set of configurations

Smaller set induces a bigger leakage:

$$-\log_2(P(C_1)+P(C_2)+P(C_3)) \leq -\log_2(P(C_1)+P(C_2))$$

Knowledge of tracker: formula

- Set of configurations represented by a formula

$$B ::= tt \mid ff \mid f = v \mid f \neq v \mid B \wedge B \mid B \vee B$$

f: browser feature
v: value

Noninterference
All configurations

$\{C_1, C_2, \ldots, C_n\}$

tt

Partial leakage
Some configurations

$\{C_i \mid C_i(\text{name})=\text{“FireFox”} \wedge C_i(\text{fonts}) \neq \text{fontsSet}\}$

$name = \text{“FireFox”} \wedge fonts \neq fontsSet$

Dynamic knowledge propagation

- Dynamic labeling *K: Vars → Formula*
  - for browser features: *K(name): name = "FireFox"*

```
x = name;
```

K(x): name = "FireFox"

```
x = 0;
if (name == "FireFox") {
    x = 1;
}
output x;
```

K(x): tt (after `x = 0;`)

K(x): name = "FireFox" (after `x = 1;`)

Dynamic knowledge propagation

```
x = 1;
if (name == "FireFox") {
     x = 1;
}
output x;
```

$K(x)$: *tt*

$K(x)$: *name = "FireFox"*

Dynamic analysis **is not very precise!**

Let's statically analyze non-executed branches!

Hybrid Monitoring

name = "FireFox" OR fonts = fontsSet

- Dynamic analysis: *env: Var → Val*
- Static analysis: *env: Var → Val ∪ {T}*

```
var x = 1;
var y = fonts;

if (name == "FireFox") {
    x = 1;
}
else {
    if (y != fontsSet) {
        x = 2;
    }
}
output x;
```

env(x) = 1 (after `var x = 1;`)

K(y): fonts = fontsSet (after `var y = fonts;`)

env(x) = 1 *K'(x): tt* (after `x = 1;`)

env(x) = 1 (after the inner `if` block)

Combination of knowledge in $K(x)$

Static		Dynamic
env(x) = 1	=	*env(x) = 1*

(name = "FireFox" => K'(x)) ∧

(name ≠ "FireFox" => K'(x))

Static analysis

name = "FireFox" OR fonts = fontsSet

- Dependency analysis $D: Var \to 2^{Var}$

```
var x = 1;              env(x) = 1
var y = fonts;          K(y): fonts = fontsSet2

if (name == "FireFox") {
     x = 1;             env(x) = 1     K'(x): tt
}
else {
     if (y != fontsSet) {
          x = 2;                       D(x) = {y}
     }                  env(x) = 1
}
output x;
```

$(name = \text{"FireFox"} \Rightarrow K'(x)) \wedge$
$(name \neq \text{"FireFox"} \Rightarrow K'(x))$

↓

$(name = \text{"FireFox"} \Rightarrow tt) \wedge$
$(name \neq \text{"FireFox"} \Rightarrow \bigwedge_{y \in D(x)} K(y))$

↓

$name \neq \text{"FireFox"} \Rightarrow fonts = fontsSet$

↓

$name = \text{"FireFox"} \vee fonts = fontsSet$

Soundness and Precision

Actual knowledge of tracker is a set of equivalent configurations $Eq(P,C)$

Definition (Soundness)

A hybrid monitor is **sound** if for all variables x, $K(x)$ over-approximates the knowledge of the tracker

$$Models(K(x)) \subseteq Eq(P,C)$$

Theorem (Soundness)

A **sound** static analysis induces a **sound** hybrid monitor.

All the theorems are proven in Coq: http://www.irisa.fr/celtique/ext/QIF/

Soundness and Precision

Definition (Precision)

A hybrid monitor A is **more precise than** a hybrid monitor B, if for all variables x:

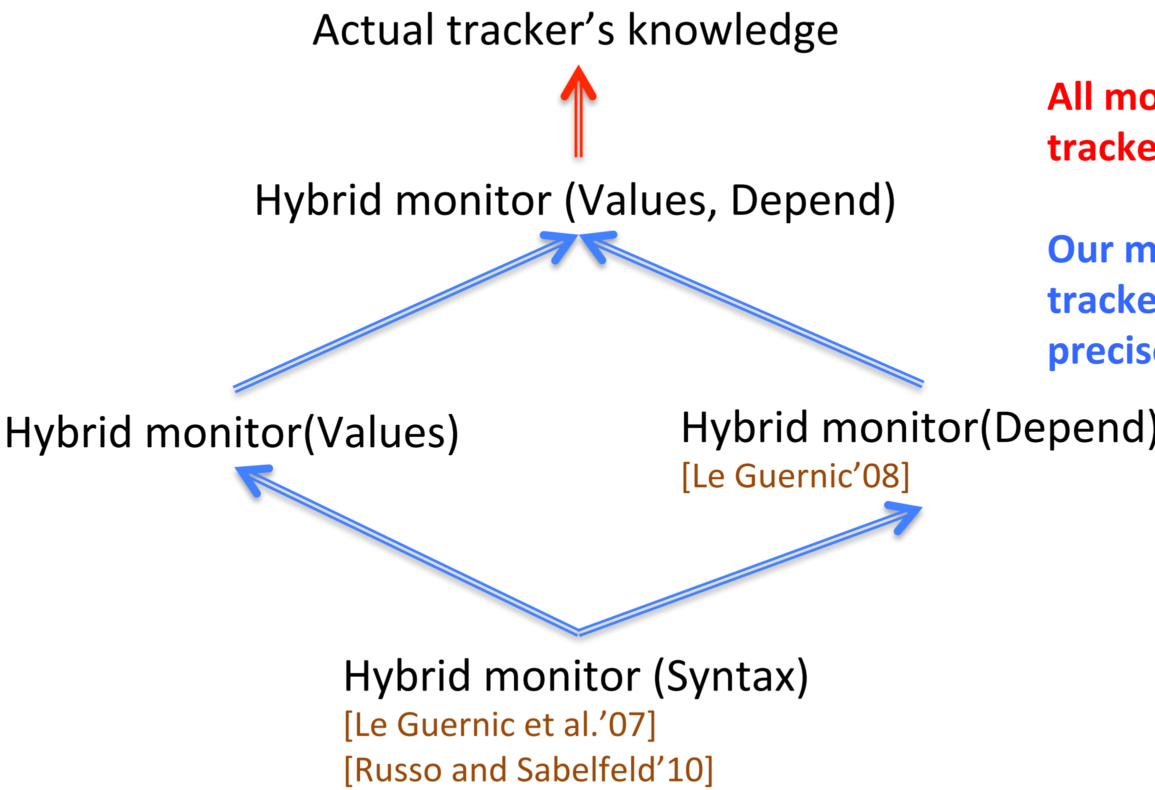
$$Models(K_B(x)) \subseteq Models(K_A(x))$$

Theorem (Precision)

A **more precise** static analysis induces a **more precise** monitor.

All the theorems are proven in Coq: http://www.irisa.fr/celtique/ext/QIF/

Hierarchy of hybrid monitors parameterized by static analysis



All monitors over-approximate tracker's knowledge

Our monitor approximates tracker's knowledge more precisely that other monitors

All the relations are proven in Coq: http://www.irisa.fr/celtique/ext/QIF/

Future work

- Support for enforcement
  - threshold-based enforcement
  - possible leakage due to enforcement action
- Extension to Java-like language
  - and, eventually, to JavaScript-like language

Our results

- Hybrid information flow monitoring
  - Labeling with knowledge
  - Knowledge => quantitative leakage
  - Parameterization by static analysis
- Soundness and precision
  - Requirements for static analysis
  - Easy comparison of hybrid monitors
- Hierarchy of hybrid monitors ordered by precision
  - Constant propagation + dependency analysis => more precise monitor