

Tracking Integrity in Updatable Security Views

Nate Foster
Princeton

Joint work with

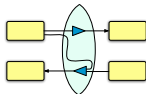
BC Pierce
Penn

S Zdancewic
Penn

V Tannen
Penn

TJ Green
UC Davis

J Vaughan
Harvard





Checking Security Policies

Table X

Id	Name	Public
●	●	F
●	●	T
●	●	F
●	●	T

Table Y

Id		
6	●	●
8	●	●

Access Control List

Usr	Y_id
42	6
42	8

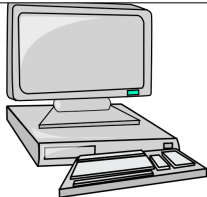
Policy:
SELECT Id, Name
FROM X
WHERE Public

Policy:
SELECT Y.*
FROM Y, Acl, User
WHERE Y_id = Y.Id
AND Usr = User.Id
AND known(User.Pass)



HTTP Request

User: 42
Password: foo



**LOOSE
LIPS**



**MIGHT
Sink Ships**



The Washington Post

"Pennsylvania yanks voter site after data leak"

THE GLOBE AND MAIL

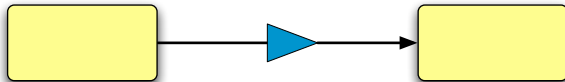
CANADA'S NATIONAL NEWSPAPER

"Passport applicant finds massive privacy breach"

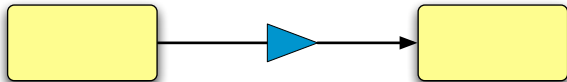
The New York Times

"Facebook glitch brings new privacy worries"

Security Views

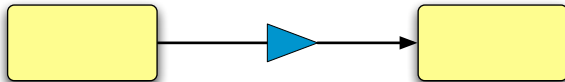


Security Views



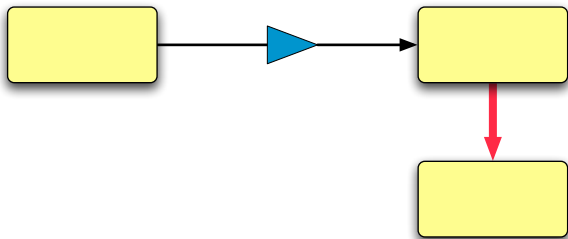
- ✓ **Robust:** impossible to leak hidden data
- ✓ **Flexible:** enforce fine-grained confidentiality policies

Security Views



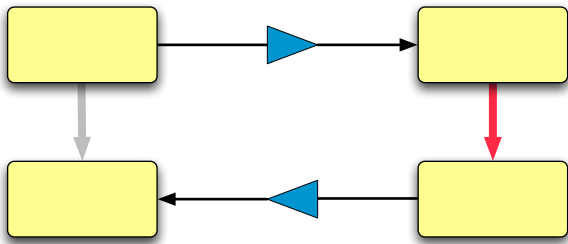
- ✓ **Robust:** impossible to leak hidden data
- ✓ **Flexible:** enforce fine-grained confidentiality policies
- ✗ Not usually **updatable**

Security Views



- ✓ **Robust:** impossible to leak hidden data
- ✓ **Flexible:** enforce fine-grained confidentiality policies
- ✗ Not usually **updatable**

Security Views



- ✓ **Robust:** impossible to leak hidden data
- ✓ **Flexible:** enforce fine-grained confidentiality policies
- ✗ Not usually **updatable**

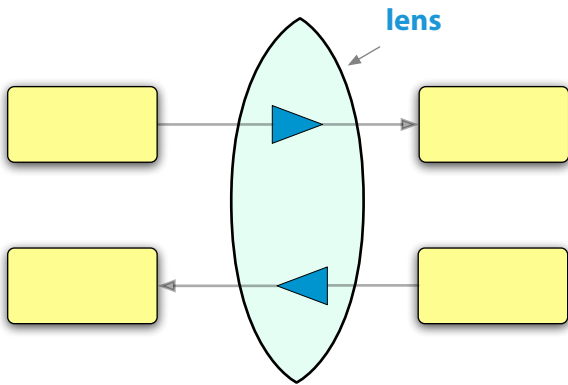
This talk: Plan

- Introduce **updatable security views**
- Describe three mechanisms for tracking **integrity**:
 1. A purely static **information-flow** analysis
 2. A dynamic analysis based on **provenance**
 3. An approach based on explicit **update operations**

Updatable Security Views

Lenses: Terminology

In recent years, we have developed a number of **programming languages** for describing **well-behaved bidirectional transformations** called **lenses**.



Lenses, Formally

A lens l mapping between a set S of sources and a set V of views is a pair of total functions

$$l.\text{get} \in S \rightarrow V$$

$$l.\text{put} \in V \rightarrow S \rightarrow S$$

Lenses, Formally

A lens l mapping between a set S of sources and a set V of views is a pair of total functions

$$l.\text{get} \in S \rightarrow V$$

$$l.\text{put} \in V \rightarrow S \rightarrow S$$

obeying “round-tripping” laws

$$l.\text{get} (l.\text{put } v \ s) = v \quad (\text{PutGet})$$

$$l.\text{put} (l.\text{get } s) \ s = s \quad (\text{GetPut})$$

for every $s \in S$ and $v \in V$.

Example: Wiki (Get)

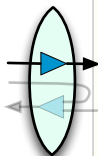
=Location=

The CIA Headquarters
is in Langley, VA.

[!On Colonial Farm Rd.]

=Employees=

* Julia Child <!1945>
! Valerie Plame 2005



=Location=

The CIA Headquarters
is in Langley, VA.

[REDACTED]

=Employees=

* Julia Child
* [REDACTED]

Example: Wiki (Update)

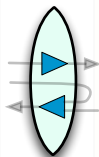
=Location=

The CIA Headquarters
is in Langley, VA.

[!On Colonial Farm Rd.]

=Employees=

- * Julia Child <!1945>
- ! Valerie Plame 2005



=Location=

The CIA Headquarters
is in McLean, VA.


Also called the Bush
Center for Intelligence.

=Employees=

- * Julia Child
- * Arthur Goldberg
- * [REDACTED]



Example: Wiki (Put)



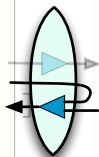
=Location=

The CIA Headquarters
is in McLean, VA.

Also called the Bush
Center for Intelligence.

=Employees=

- * Julia Child <!1945>
- * Arthur Goldberg <!1900>
- ! Valerie Plame 2005



=Location=


The CIA Headquarters
is in McLean, VA.

Also called the Bush
Center for Intelligence.

=Employees=

- * Julia Child
- * Arthur Goldberg
- * [REDACTED]

Example: Wiki (Put)

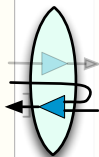
=Location=

The CIA Headquarters
is in McLean, VA.

Also called the Bush
Center for Intelligence.

=Employees=

- * Julia Child <!1945>
- * Arthur Goldberg <!1900>
- ! Valerie Plame 2005



=Location=

The CIA Headquarters
is in McLean, VA.

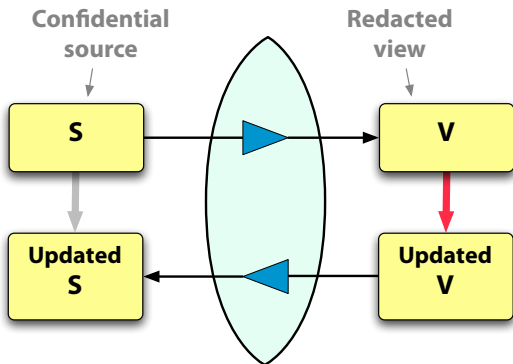
Also called the Bush
Center for Intelligence.

=Employees=

- * Julia Child
- * Arthur Goldberg
- * [REDACTED]

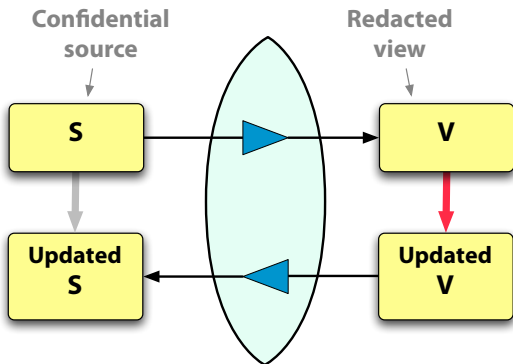
Observe that propagating the update forces the **put** function to modify hidden data in the source!

Requirements



1. Confidentiality: **get** does not leak secret data
2. Integrity: **put** does not taint trusted data

Requirements



1. Confidentiality: **get** does not leak secret data
2. Integrity: **put** does not taint trusted data

This talk



Integrity

Question: Should the (possibly untrusted) user of the view be allowed to modify hidden (possibly trusted) source data?

Integrity

Question: Should the (possibly untrusted) user of the view be allowed to modify hidden (possibly trusted) source data?

Answer: Maybe!

There are *many* alternatives, trading off which information in the source can be trusted against which information in the view can be edited.

Some Integrity Policies

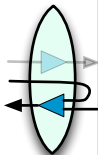
=Location=

The CIA Headquarters
is in McLean, VA.

Also called the Bush
Center for Intelligence.

=Employees=

- * Julia Child <!1945>
- * Arthur Goldberg <!1900>
- ! Valerie Plame 2005



=Location=

The CIA Headquarters
is in McLean, VA.

Also called the Bush
Center for Intelligence.

=Employees=

- * Julia Child
- * Arthur Goldberg
- * [REDACTED]

Policy: "Nothing is trusted" (whole source is tainted)

Effect: Arbitrary edits to the view are allowed but any hidden data in the source can be modified by **put**

Some Integrity Policies

=Location=

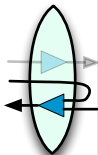
The CIA Headquarters
is in McLean, VA.

[!On Colonial Farm Rd.]

Also called the Bush
Center for Intelligence.

=Employees=

* Julia Child <!1945>
! Valerie Plame 2005



=Location=

The CIA Headquarters
is in McLean, VA.

[REDACTED]

Also called the Bush
Center for Intelligence.

=Employees=

* Julia Child
* [REDACTED]

Policy: "Confidential data trusted; public data tainted"

Effect: Cannot add or delete redacted blocks or list items

Some Integrity Policies

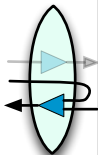
=Location=

The CIA Headquarters
is in Langley, VA.

[!On Colonial Farm Rd.]

=Employees=

* Julia Child <!1945>
! Valerie Plame 2005



=Location=

The CIA Headquarters
is in Langley, VA.

[REDACTED]

=Employees=

* Julia Child
* [REDACTED]

Policy: "Everything is trusted"

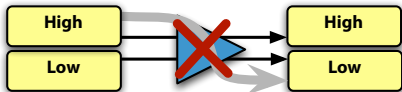
Effect: No edits are allowed

Approach #1:

Information-Flow Analysis

Non-interference

Observation: many policies can be formulated in terms of non-interference.



A transformation is **non-interfering** if the “low” parts of the output do not depend on the “high” parts of the input.

Non-interference— Integrity

Observation: many policies can be formulated in terms of **non-interference**.



A transformation is **non-interfering** if the “low” parts of the output do not depend on the “high” parts of the input.

E.g., if the data contains “**tainted**” and “**trusted**” portions



then the **tainted** parts of the input do not affect the **trusted** parts of the output.

Secure Lenses, Formally

The expectation that

"Tainted inputs to **put** should not affect *Trusted* outputs"

can be expressed by generalizing GetPut...

$$l.\mathbf{put} (l.\mathbf{get} s) s = s \quad (\text{GetPut})$$

... like this:

$$\frac{v \approx (l.\mathbf{get} s)}{l.\mathbf{put} v s \approx s} \quad (\text{GetPutSecure})$$

(...as well as a similar non-interference law for confidentiality.)

The PutPut Law, Redux

The following law can be derived:

$$\frac{v' \approx l.\text{get } s \approx v}{l.\text{put } v' (l.\text{put } v s) \approx l.\text{put } v' s}$$

This law says that **put** must have no “side-effects” on the trusted parts of the source.

It generalizes the “**constant complement**” condition, the gold standard for correct view update in databases.

Labels

Fix a lattice of **integrity labels**, e.g.



Annotated Regular Expressions

Mark up the **source schema** (a regular expression) to indicate which data is *Tainted* and which is *Trusted*.

$$\mathcal{R} ::= \emptyset \mid u \mid \mathcal{R} \cdot \mathcal{R} \mid \mathcal{R} | \mathcal{R} \mid \mathcal{R}^* \mid \mathcal{R} : k$$

The annotation k is drawn from the lattice of integrity labels.

Annotated Regular Expressions

Mark up the **source schema** (a regular expression) to indicate which data is *Tainted* and which is *Trusted*.

$$\mathcal{R} ::= \emptyset \mid u \mid \mathcal{R} \cdot \mathcal{R} \mid \mathcal{R} | \mathcal{R} \mid \mathcal{R}^* \mid \mathcal{R} : k$$

The annotation k is drawn from the lattice of integrity labels.

For example:

$$S \triangleq \left(\begin{array}{l} "*" \cdot \text{TEXT} : \text{Tainted} \cdot "<!" \cdot \text{YEAR} \cdot ">\backslash n" \\ | \\ "! " \cdot \text{TEXT} \cdot " " \cdot \text{YEAR} \cdot "\backslash n")^* \end{array} \right.$$

Note: no annotation is equivalent to the least annotation

Equivalences

From an annotated schema, we can read off an equivalence relation \approx_k , for each k in the lattice of integrity labels.

- $\approx_{Trusted}$ — “ s and s' agree on trusted data”

```
* Julia Child, the chef <!1945>  
! Valerie Plame 2005
```

$\approx_{Trusted}$

```
* Julia Child <!1945>  
! Valerie Plame 2005
```

- $\approx_{Tainted}$ — “ s and s' agree on tainted and trusted data” (i.e., they are identical)

```
* Julia Child <!1945>  
! Valerie Plame 2005
```

$\approx_{Tainted}$

```
* Julia Child <!1945>  
! Valerie Plame 2005
```

Approach #2:

Data Provenance

Data Provenance

Observation: information-flow analysis is an effective but coarse way to track integrity.

Typing rules mark large parts of the source as “potentially tainted” → need a finer-grained mechanism.

Data Provenance

Observation: information-flow analysis is an effective but coarse way to track integrity.

Typing rules mark large parts of the source as “potentially tainted” → need a finer-grained mechanism.

Definition (provenance, n.)

1. the source or origin of an object; its history and pedigree; a record of the ultimate derivation and passage of an item through its various owners [Oxford English Dictionary]
2. the description of the origins of a piece of data and the process by which it arrived in a database [Buneman, Khanna, Tan '01]

Provenance Semirings

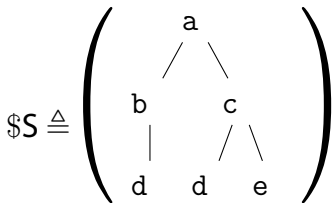
Plan:

- Work with annotations from a **commutative semiring** [Green, Karvounarakis, Tannen PODS '07]
- Decorate source structures with annotations
- Refine language semantics to propagate annotations

Intuition: (+) represents alternate use while (·) represents joint use of data

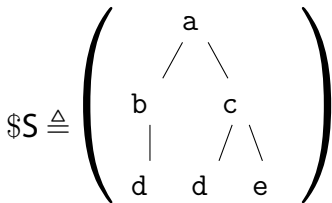
Provenance Semirings for XQuery

Intuition: (+) represents alternate use while (.) represents joint use of data

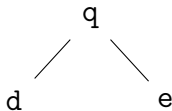
$$q \triangleq \text{element } q \{ \text{for } \$t \text{ in } \$S \text{ return} \\ \text{for } \$x \text{ in } (\$t)/* \text{ return} \\ (\$x)/* \}$$


Provenance Semirings for XQuery

Intuition: (+) represents alternate use while (.) represents joint use of data

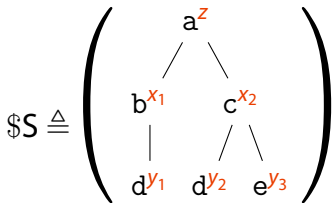
$$q \triangleq \text{element } q \{ \text{for } \$t \text{ in } \$S \text{ return} \\ \text{for } \$x \text{ in } (\$t)/* \text{ return} \\ (\$x)/* \}$$


$\longrightarrow q \longrightarrow$

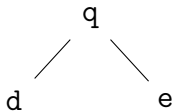


Provenance Semirings for XQuery

Intuition: (+) represents alternate use while (.) represents joint use of data

$$q \triangleq \text{element } q \{ \text{for } \$t \text{ in } \$S \text{ return} \\ \text{for } \$x \text{ in } (\$t)/* \text{ return} \\ (\$x)/* \}$$


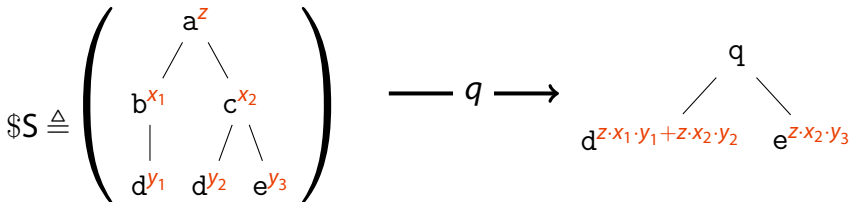
$\longrightarrow q \longrightarrow$



Provenance Semirings for XQuery

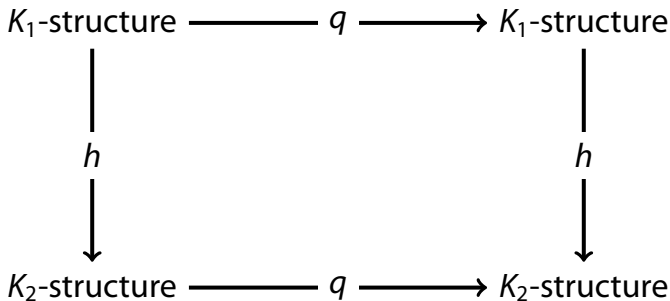
Intuition: (+) represents alternate use while (.) represents joint use of data

```
 $q \triangleq \text{element } q \{ \text{for } \$t \text{ in } \$S \text{ return}$   
     $\text{for } \$x \text{ in } (\$t)/* \text{ return}$   
     $(\$x)/* \}$ 
```



Fundamental Property

For every query q and homomorphism of commutative semirings $h \in K_1 \rightarrow K_2$, the following diagram commutes:



Application of Fundamental Property

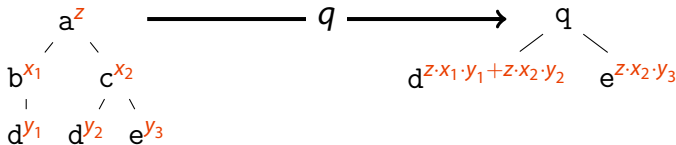
Let K be $\{\text{U}ntouchable, \text{Tru}Sted, \text{Tai}Nted\}$

Let h be the homomorphism that maps x_2 to **N** and every other (non-zero) label to **S**

Application of Fundamental Property

Let K be $\{\text{U}ntouchable, \text{T}ru\text{S}ted, \text{T}ai\text{N}ted\}$

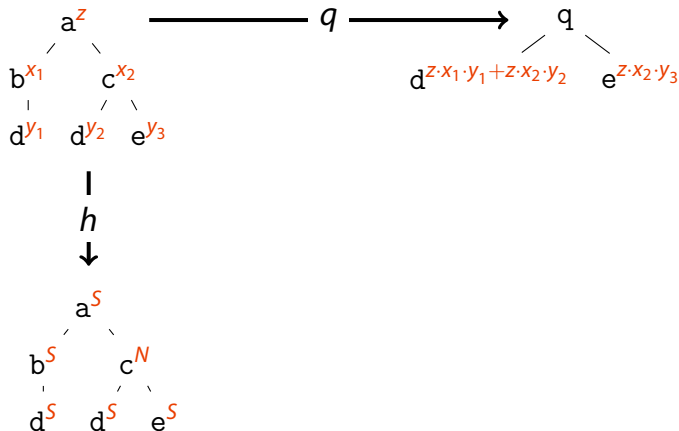
Let h be the homomorphism that maps x_2 to **N** and every other (non-zero) label to **S**



Application of Fundamental Property

Let K be $\{\text{U}ntouchable, \text{Tru}Sted, \text{Tai}Nted\}$

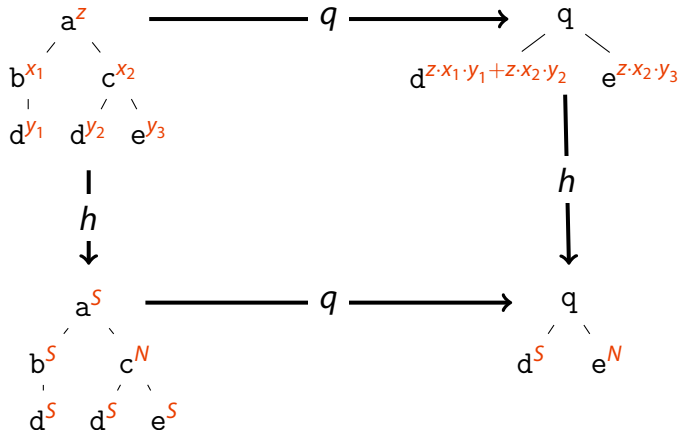
Let h be the homomorphism that maps x_2 to **N** and every other (non-zero) label to **S**



Application of Fundamental Property

Let K be {**U**ntouchable, Tru**S**ted, Tai**N**ted}

Let h be the homomorphism that maps x_2 to **N** and every other (non-zero) label to **S**



Approach #3:

Audit

[work-in-progress with J Vaughan]

Audit Logs

Motivation: want to allow untrusted users to modify hidden information in the source but also provide its owner with a way to audit their changes

To to this, we need:

1. a representation for audit logs
2. a semantics for lenses that generates logs
3. tools for examining and manipulating logs

Δ -Lenses

$$o ::= \text{nop} \mid o; o \mid \rightsquigarrow v \mid (o, o) \mid \text{inl } o \, v \mid \text{inr } o \, v \\ \mid \text{swap } i \, j \mid \text{insert } v \mid \text{delete} \mid \text{hd } o \mid \text{tail } o$$

A Δ -lens l is a pair of total functions

$$l.\text{get} \in O_S \rightarrow O_V$$

$$l.\text{put} \in O_V \rightarrow O_S$$

Δ -Lenses

$$o ::= \text{nop} \mid o; o \mid \rightsquigarrow v \mid (o, o) \mid \text{inl } o \, v \mid \text{inr } o \, v \\ \mid \text{swap } i \, j \mid \text{insert } v \mid \text{delete} \mid \text{hd } o \mid \text{tail } o$$

A Δ -lens l is a pair of total functions

$$l.\text{get} \in O_S \rightarrow O_V$$
$$l.\text{put} \in O_V \rightarrow O_S$$

Can use operations

- characterize update quality
- implement efficient, fine-grained rollbacks
- handle concurrent updates
- maintain views incrementally

Thank You!

Collaborators: Benjamin Pierce, Steve Zdancewic, Val Tannen, TJ Green, Jeff Vaughan

Other Contributors: Aaron Bohannon, Davi Barbosa, Julien Cretin, Ravi Chugh, Malo Deniélou, Michael Greenberg, Michael Greenwald, Christian Kirkegaard, Stéphane Lescuyer, Adam Magee, Jon Moore, Alexandre Pilkiewicz, Danny Puller, Alan Schmitt



Want to play? Boomerang is available for download.

- Source code (LGPL)
- Precompiled binaries
- Papers, tutorial, and demos

<http://www.cs.princeton.edu/~jnfoster/>