Enforcing Agile Access Control Policies in Relational Databases using Views

Nicolas Papernot, Patrick McDaniel, and Robert J. Walls
Department of Computer Science and Engineering, Penn State University
{ngp5056, mcdaniel, rjwalls}@cse.psu.edu

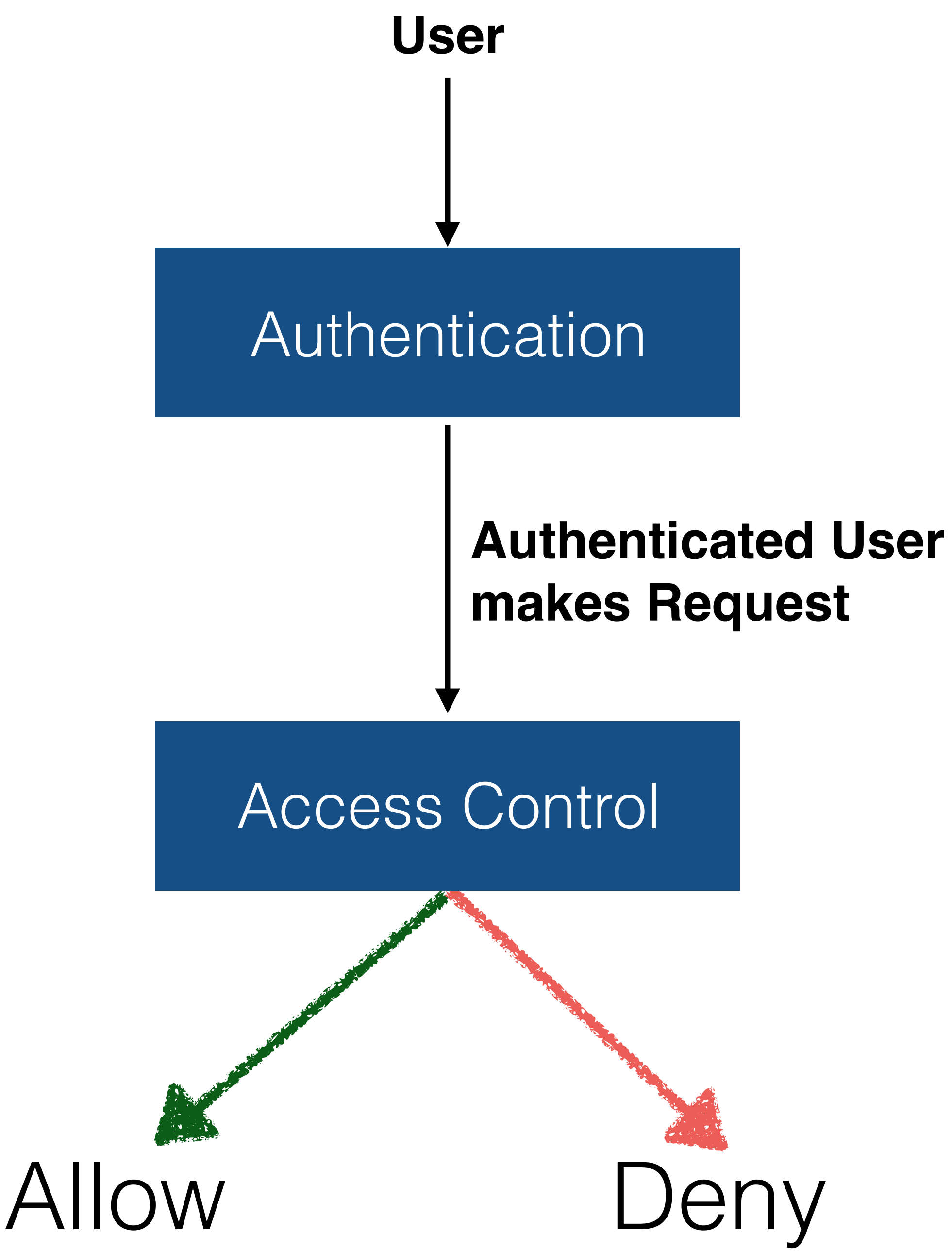
Research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-13-2-0045 (ARL Cyber Security CRA).

Databases and Data Protection

- Databases store **valuable** and **sensitive** information
- 7 out of 10 top security threats involve databases [2]
- Deploy data protection mechanisms: authentication, access control, encryption…

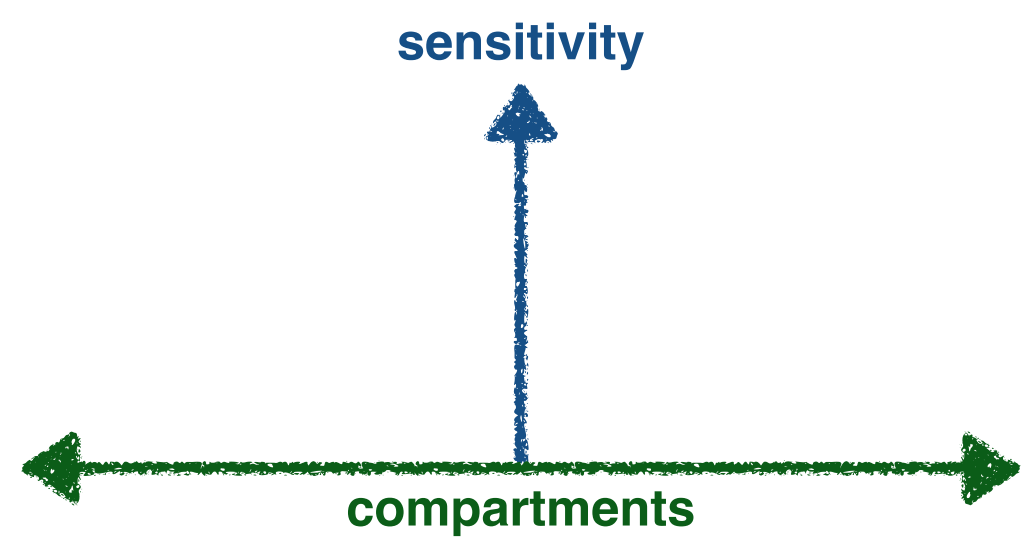
Databases and Access Control

- Access control applies to **authenticated users**
- Access control preserve **confidentiality** and **integrity**
- Most current access control workflows rely on individuals to **manually implement policy models** using access control mechanisms
- Access Control rules specified by a specific **policy**



Use Case: Multilevel Security for Relational Databases

- Implements the **need-to-know principle**
- Based on security **lattice** introduced by Bell and LaPadula
- Enforcement through two **properties**:
  - Simple security property
  - Star property
- **Static user privileges are not appropriate**:
  - Update of policy is error-prone
  - No dynamic evolution (mutual exclusion)

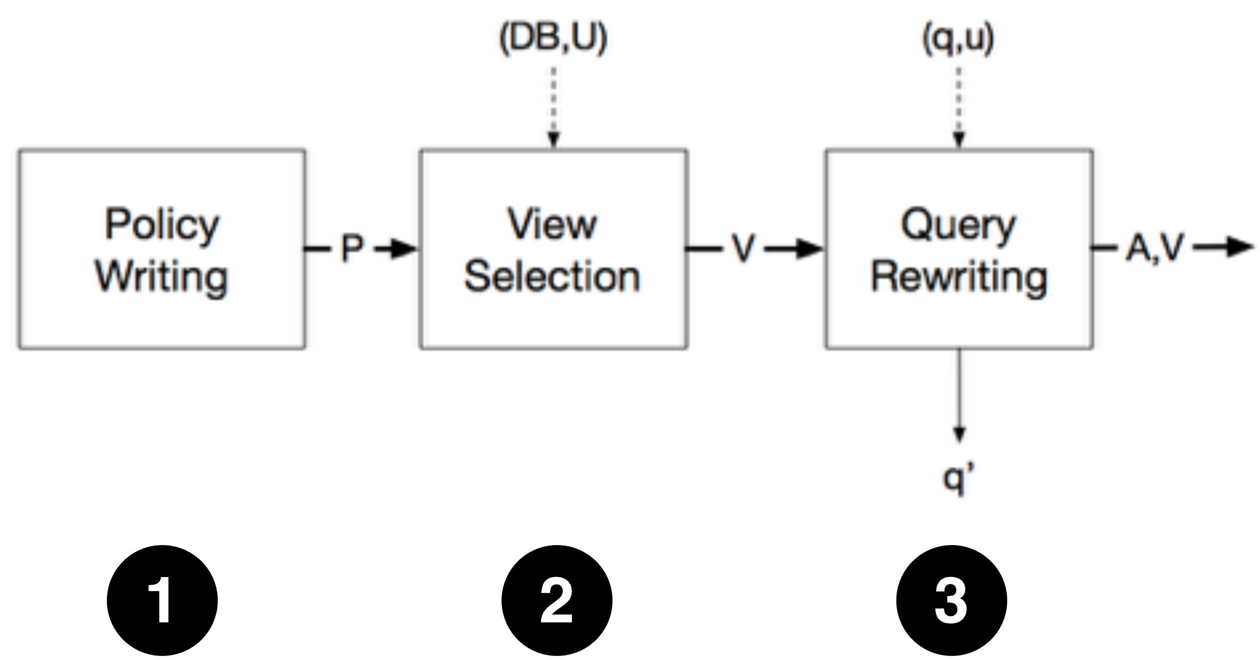


id	name	mission	destination
-	U,{Naval}	TS,{Naval}	S,{Naval}
1	Seawolf	spy	Russia
2	Roosevelt	patrol	Gulf of Aden
3	Normandy	patrol	Gulf of Oman

Example table with MLS labels

Approach to Agile Access Control Policy Enforcement

1. **Policy Writing**:
   - Define policy set **P**
   - **High-level abstraction**
2. **View Selection**:
   - Input: DB, P, U
   - **Generates** a set V reflecting P
3. **Query Rewriting**:
   - Input: DB, P, U, V, A,
   - **Rewrite query** q’ over V
   - **Update** V and A



Overview of the Approach

❶ Policy Writing

The **ASL language** is made of:

- **Predicates**: active, in, dirin, typeof (describe the users and objects)
- **Rules**: done, authorization, derivation, access control, resolution, integrity (describe the policy)

We make the following assumptions:

- **Column granularity**
- **Positive authorizations** only (no conflict resolution)

```
Require: L, DB, U
Let P = ∅
for each T ∈ DB do
    for each u ∈ U do
        if (l_{T,1} ≤ l_{u,1}) & ... & (l_{T,n} ≤ l_{u,n}) then
            add rule grant(T, u, R, SELECT) ←
              active(u, l_u) & typeof(T, l_T)
        else if (l_{T,1} ≥ l_{u,1}) & ... & (l_{T,n} ≥ l_{u,n}) then
            add rule grant(T, u, R, INSERT) ←
              active(u, l_u) & typeof(T, l_T)
        else if (l_{T,1} = l_{u,1}) & ... & (l_{T,n} = l_{u,n}) then
            add rule grant(T, u, R, UPDATE) ←
              active(u, l_u) & typeof(T, l_T)
            add rule grant(T, u, R, DELETE) ←
              active(u, l_u) & typeof(T, l_T)
        end if
    end for
end for
return P
```

Bell-LaPadula model

$$p_1 = \texttt{grant}(T, u, r, *) \leftarrow !done(T', u, *, *, *) \ \& \ \texttt{typeof}(T, A) \ \& \ \texttt{typeof}(T', B)$$

$$p_2 = \texttt{grant}(T, u, r, *) \leftarrow !done(T', u, *, *, *) \ \& \ \texttt{typeof}(T, B) \ \& \ \texttt{typeof}(T', A)$$

Chinese Wall policy

Jajodia, S., Samarati, P., & Subrahmanian, V. S.
A logical language for expressing authorizations.
IEEE Symposium on Security and Privacy, 1997

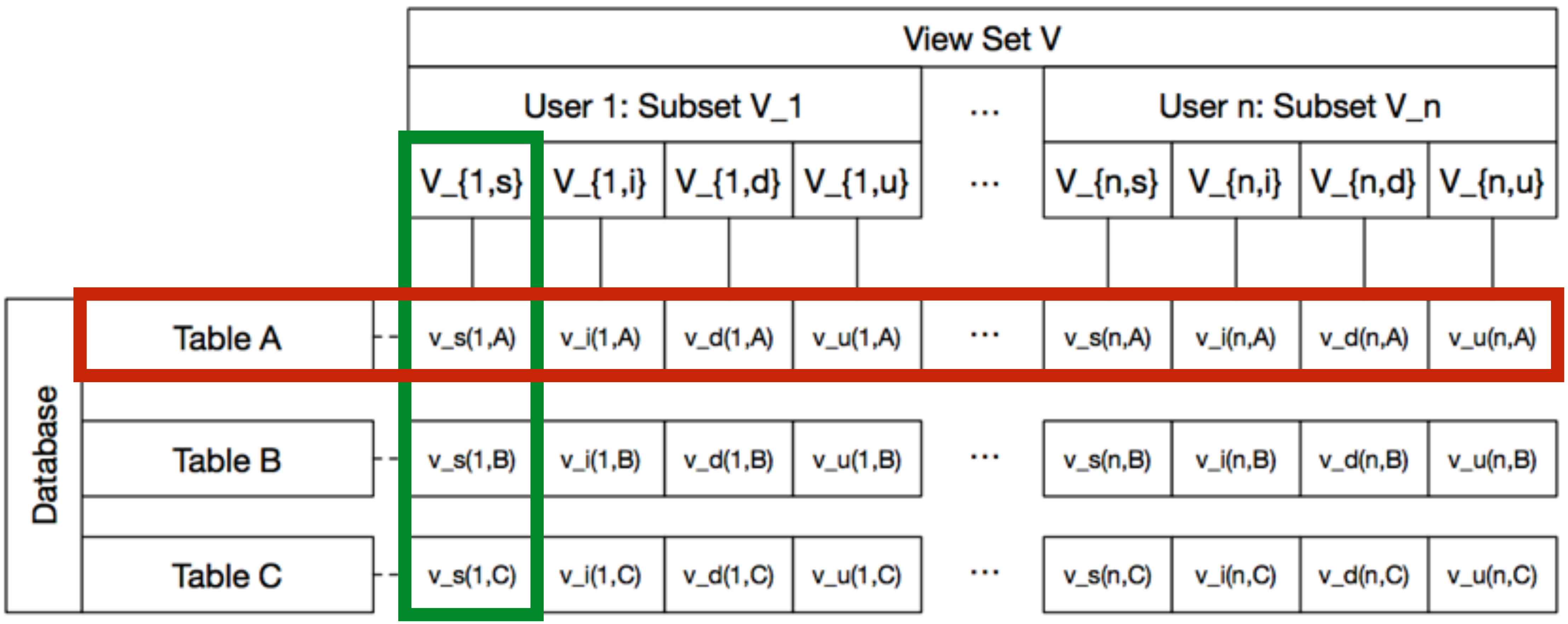
2 View Selection

- **each view corresponds to an SQL operation**: `SELECT, INSERT, DELETE,` and `UPDATE`
- a view is set for each **(user, table) pair**
- the view selection is done by considering grant, cando, and dercando **rules** and building sets `G,Gc,C,Cc,Dc`

```
Require: DB, U, P
V = V_1 ∪ ... ∪ V_|U| = ∅
for each u ∈ U do
    for each a ∈ {SELECT, INSERT, DELETE, UPDATE} do
        Create empty set V_{u,a}
        for each table T ∈ DB do
            Compute G = G(u, T, *, a)
            Compute C = C(u, T, *, a)
            Compute D = D(u, T, *, a)
            if G ∪ C ∪ D = ∅ then
                columns = ∅
                for each column c ∈ T do
                    Compute G_c = G(u, T, c, a)
                    Compute C_c = C(u, T, c, a)
                    Compute D_c = D(u, T, c, a)
                    if (G_c ∪ C_c ∪ D_c ≠ ∅) then
                        columns = columns ∪ {c}
                    end if
                end for
                v_a(u, T) = SELECT columns FROM T
            else if (G ∪ C ∪ D ≠ ∅) then
                create v_a(u, T) = SELECT * FROM T
            end if
            V_{u,a} = V_{u,a} ∪ {v}
        end for
    end for
    V_u = V_{u,s} ∪ V_{u,i} ∪ V_{u,d} ∪ V_{u,u}
end for
return V
```

Fig. 6. ASL View Selection algorithm

2 View Selection



Overview of the View Set

❷ View Selection

- **each view corresponds to an SQL operation**: `SELECT, INSERT, DELETE,` and `UPDATE`
- a view is set for each **(user, table) pair**
- the view selection is done by considering grant, cando, and dercando **rules** and building sets `G,Gc,C,Cc,D,Dc`

Require: DB, U, P
$V = V_1 \cup ... \cup V_{|U|} = \emptyset$
for each $u \in U$ **do**
  for each $a \in \{$SELECT, INSERT, DELETE, UPDATE$\}$ **do**
    Create empty set $V_{u,a}$
    for each table $T \in DB$ **do**
      Compute $G = G(u, T, *, a)$
      Compute $C = C(u, T, *, a)$
      Compute $D = D(u, T, *, a)$
      if $G \cup C \cup D = \emptyset$ **then**
        $columns = \emptyset$
        for each column $c \in T$ **do**
          Compute $G_c = G(u, T, c, a)$
          Compute $C_c = C(u, T, c, a)$
          Compute $D_c = D(u, T, c, a)$
          if $(G_c \cup C_c \cup D_c \neq \emptyset)$ **then**
            $columns = columns \cup \{c\}$
          end if
        end for
        $v_a(u, T)$ = SELECT $columns$ FROM T
      else if $(G \cup C \cup D \neq \emptyset)$ **then**
        create $v_a(u, T)$ = SELECT * FROM T
      end if
      $V_{u,a} = V_{u,a} \cup \{v\}$
    end for
  end for
  $V_u = V_{u,s} \cup V_{u,i} \cup V_{u,d} \cup V_{u,u}$
end for
return V

Fig. 6. ASL View Selection algorithm

❸ Query Rewriting

1. **rewrite query**: straight forward thanks to the view-table-user correspondance

2. **update view set**: update of view set is done by identifying relevant rules (using subroutine find-done) and removing views conflicting with the new accesses made in the query

3. **update past accesses**: append the accesses made in the query

```
Require: $DB$, $P$, $A$, $V$, $u$, $q$
  $q' = q$
  $A' = \emptyset$
  let $a$ be the operation of $q$
  for each table column $T.c$ mentioned in $q$ do
      if $\exists v \in V_{u,a}, T.c \subset v$ then
          rewrite $q'$ with $v.c$ instead of $T.c$
          if $v.c \neq \emptyset$ then
              $A' \leftarrow A' \cup \texttt{done}(T.c, u, R, a, t)$
          end if
      else
          return "query not compliant"
      end if
  end for
  for each $d \in A'$ do
      $\{p_j\}_j = \texttt{find-done}(P \cup A \cup A', d)$
      for each $p_j$ do
          if $p_j == \texttt{error}()$ then
              return "query not compliant"
          else if $p_j == \texttt{dercando}(T.c, s, +a)$ then
              empty c from $v \in V_{u,a}$ such that $T.c \subset v$
          else if $p_j == \texttt{do}(T.c, s, +a)$ then
              empty c from $v \in V_{u,a}$ such that $T.c \subset v$
          else if $p_j == \texttt{grant}(T.c, s, r, +a)$ then
              empty c from $v \in V_{u,a}$ such that $T.c \subset v$
          end if
      end for
  end for
  return $\{q', V', A' \cup A\}$
```

Fig. 7. ASL Query Rewriting algorithm

Conclusion

- Tractable algorithms capable of **enforcing policies** expressed using **ASL language**
  - Capable of enforcing **static access control, information flow, mutual exclusion**
  - Support for all **SQL queries** (SELECT, INSERT, DELETE, UPDATE)
- **No modification** required of:
  - Database schema
  - User queries (no need to change application code)
- Future work:
  - Implementing the technique within a database
  - Evaluate performance impact
  - Design technique allowing policy set P and view set V updates on the fly
    -> View management is main factor of complexity

Questions?

Research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-13-2-0045 (ARL Cyber Security CRA). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.