

POLICY-BASED MANAGEMENT AND SHARING OF SENSITIVE INFORMATION AMONG GOVERNMENT AGENCIES

Tryg Ager, Christopher Johnson, Jerry Kiernan
IBM Almaden Research Center
San Jose, CA

ABSTRACT

We propose a set of policy-based technologies to enable increased information sharing among government agencies without compromising information security or individual privacy. Our approach includes: (1) fine-grained access controls that support deny and filter semantics to satisfy complex policy conditions; (2) a sticky policy capability that allows consolidation of information from multiple sources subject to the original disclosure policies of each source; (3) a curation organization that enables agencies to apply and manipulate item-level security classifications and disclosure policies; (4) an auditing system that accounts for the curation history of each information item; and (5) a provenance auditing method that traces derivations of information over time to support evaluations of information quality. Our goal is to present a vision for solving outstanding information sharing problems in government agencies and provide direction for the development of future government information systems.

1. Introduction

United States defense, intelligence, and law enforcement agencies face increasingly complex requirements concerning the management and sharing of classified and other sensitive information. Recent homeland security laws [1] [2] and executive orders [3] [4] [5] require that government agencies ensure appropriate access to, and sharing of, sensitive information while protecting the privacy of individuals. They also require government information systems ensure the integrity of this information [6]. However, existing systems are not sufficiently fine-grained to handle many complex scenarios. In this paper, we present an integrated set of policy-based technologies that address several real-world information sharing problems in government agencies. We intend for this work to inspire further development of policy-based technologies that enhance national security without encroaching on civil liberties.

Paper Organization The remainder of the paper is structured as follows. In Section 2, we discuss the advantages of policy-based access and disclosure controls and propose methods of integrating fine-grained filter and deny semantics. In Section 3, we introduce a

sticky policy model that allows enforcement of multiple disclosure policies over consolidated information. In Section 4, we advocate using fine-grained policy controls to raise and lower security classifications and describe an auditing system to track these changes. In Section 5, we propose a method of provenance auditing to trace all queries and operations that have accessed specific information. We conclude in Section 6.

2. Policy-Based Access and Disclosure Controls

Government information systems should employ fine-grained policy controls to manage access and disclosure of classified and other sensitive information. The row-level security systems currently used by many agencies grant access to entire records, in accordance with the user's security privileges, but do not allow agencies to enforce varying classification levels for individual cells (i.e., row-column intersections) in the same row. They also do not consider the purpose of access or intended recipient of the information, if different than the user, in deciding whether to grant access. Agencies need more flexible and granular controls to ensure that each information item is available to authorized individuals for any proper purpose, but no information is disseminated below its classification level. Policy-based controls efficiently and securely address these concerns.

Hippocratic databases (HDB) [7] are designed to limit access to, and disclosure of, sensitive information in accordance with user privileges, purpose of access, and intended recipients [8]. HDBs enforce policies by transforming user queries such that they access only policy-compliant information in the database. We suggest using similar query transformation techniques to regulate access and disclosure of classified and other sensitive information maintained by government agencies. Policy-based access and disclosure controls would allow agencies to implement multi-level security [9] down to the cell level in the database and incorporate various contextual enforcement conditions. Moreover, these policy-based controls would facilitate information sharing by appropriately protecting certain sensitive items without unduly restricting others. The remainder of this section proposes extensions to HDB enforcement necessary to handle more complex problems and scenarios in government agencies.

2.1 Integration of Filter and Deny Semantics

In some circumstances, government agencies must *filter* query results by allowing the user to access permitted information, but concealing any prohibited information sought by the query. Other situations require agencies to *deny* access by signaling to the user that the query seeks prohibited information. Thus, government information systems should support both filter and deny semantics, as suggested in [10]. We illustrate the integration of filter and deny semantics in the following scenario.

Filter/Deny Scenario Suppose that a military officer is planning a ground operation to secure a strategic target. The officer queries an intelligence database to determine the current locations of any enemy forces deployed near his intended route to the target. Pursuant to his security clearance, the officer is authorized to see the locations of enemy forces situated within twenty miles of the target route, but he must be denied access to exact locations of enemy forces located more than twenty miles away. The officer is also not entitled to access any information about enemy operatives reported by human intelligence, regardless of location, as this information is highly classified. Figure 1 shows the policy rules that should govern the officer’s access to the database.

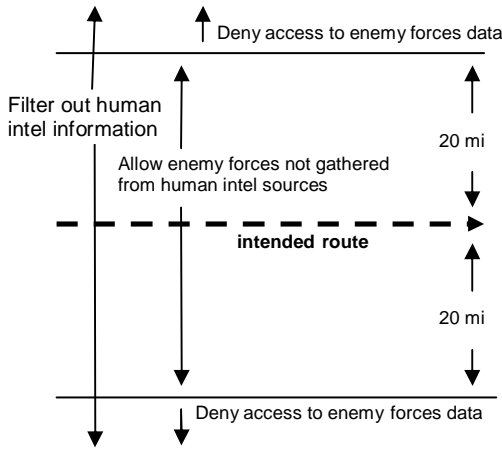


Figure 1. Integrating Filter and Deny Semantics

To answer the officer’s queries completely, without providing any misleading or prohibited information, the system must (i) *allow* access to the locations of enemy forces within twenty miles of the route that were not obtained by human intelligence; (ii) *filter* out any information about enemy operatives obtained by human intelligence; and (iii) *deny* access to the locations of enemy forces situated more than twenty miles from the officer’s intended route.

In this scenario, filtering alone may mislead the officer by suggesting that there are no enemy forces located

more than twenty miles from his intended route. Instead, the system should raise an “access denied” exception in situations where returning a filtered result would be inappropriate. On the other hand, denial alone would flatly reject queries that would return any prohibited information. In the present scenario, denial would reject the officer’s query requesting enemy forces within twenty miles of his intended route, unless he altered the predicate to seek only information not obtained by human intelligence. The system should instead use filtering when policies require that a certain subset of information be excluded from the result. We discuss our approach to supporting filter and deny semantics in the next section.

2.2 Active Enforcement of Filter and Deny Rules

Our proposed active enforcement approach precisely enforces filter and deny rules by examining a query against applicable policies and allowing the query predicate to subsume the policy predicate. We annotate each policy rule with an “f” for filter or a “d” for deny, as depicted in Figure 2. If a query involves multiple policies, and a mixture of filter and deny rules govern access to the same column, all filter rules for the column are temporarily converted into deny rules for that query.

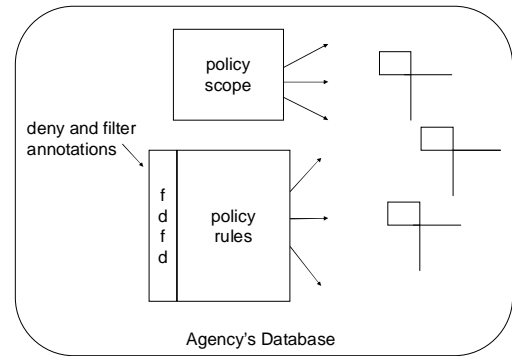


Figure 2. Annotation of Policy Rules

Filter rules allow disclosure of information that complies with policies, but return null values for any information that violates a policy. Deny rules raise a runtime exception in response to a query that would reveal information that is prohibited by policy, informing the user that he is not authorized to access that information.

2.2.1 Enforcement of Filter Rules

If access to a table’s column C is within the scope of a filter policy rule, query rewrite transformations for policy enforcement replace references to the column C in the query with a conditional expression of the form:

```
If (policy-rule-condition) then c else null
```

The internal representation of queries in our proposed system is based on the Query Graph Model (QGM) [11], which graphically captures the semantics of queries and provides convenient data structures for transforming them. QGM portrays entities as boxes and depicts entity relationships as lines between the boxes. Entities can be operators such as select, groupby, or union. Lines represent quantifiers that feed one operator by ranging over the output of others. Figure 3 shows the integration of active enforcement constraints into a query graph.

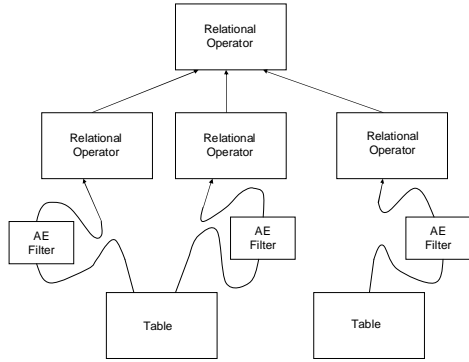


Figure 3. Active Enforcement with Filter Semantics

The active enforcement (AE) algorithm selects the tables participating in the query, represented by the two boxes at the bottom of the graph. It then introduces AE policy constraints between the output of the tables and their consumers, thus filtering all of the data accessed by the query. The AE filter propagates (i.e., allows) each column that is not governed by a policy. Otherwise, the filter conditionally propagates the column’s value depending on the policy-rule-condition. If disclosing a value would violate a policy-rule-condition, the filter replaces that value with a null value.

In most cases, null values do not reveal whether the information in null cells is unknown or prohibited. But if the query issuer has background knowledge of the database structure, and a null value conclusively indicates the presence of prohibited information (i.e., cells are never null otherwise), then returning a null value may cause unintended information leakage. In such cases, row-level filtering, in which the entire row containing the prohibited cell value is omitted, may be more appropriate than cell-level filtering.

2.2.2 Enforcement of Deny Rules

If access to a table’s column C is within the scope of a deny policy rule, we replace references to the column C in the query with a conditional expression of the form:

```
If (policy-rule-condition) then c else
raise_error("76543", "access denied")
```

The raise_error function dynamically triggers an “access denied” exception during query evaluation if the policy-rule-condition is not satisfied for any binding of the tuple variables in the query. The system only raises the error once the policy-rule-condition is no longer met, so it may generate partial results before issuing an access denied exception. Because the order in which the data is accessed depends on the query optimizer’s access plan, the sequence and number of results returned before an error is raised are non-deterministic. The system must ascertain whether a tuple has been selected by a query’s predicate before confirming whether the policy prohibits its disclosure. Otherwise, the presence of any prohibited value in a table may cause an “access denied” exception irrespective of whether the data would be selected and returned by the query. As shown in Figure 4, our query transformation scheme generates an exception only if a prohibited cell qualifies the query’s predicate.

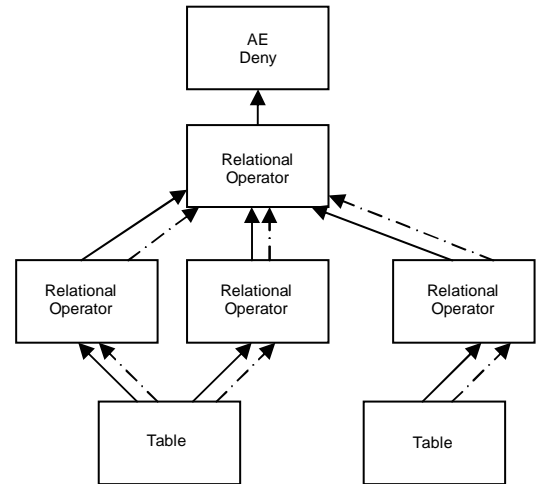


Figure 4. Active Enforcement with Deny Semantics

The active enforcement box (AE Deny) at the top of the query graph evaluates deny constraints only after all other query predicates have been qualified. This box projects all columns of the original query and contains all relevant policy-rule-conditions. Since these policy-rule-conditions might reference columns not used in the original query, these columns may not be available at the top of the query graph. We thus propagate additional columns needed by the policy-rule-conditions up the query graph, as represented by the dash-dotted lines.

Multiple Query Vulnerability Because generating “access denied” exceptions depends upon whether specific cells are accessed by a query, it may be possible to uncover the exact values of denied cells by issuing multiple queries. For example, suppose that a user is allowed to access the name of any friendly platoon leader, but not the locations of any friendly troops. The

user could first extract the names of all platoon leaders in companies he expects to be in the area and then issue additional queries that select each platoon leader by name and estimated values of location. When the location estimate is correct, the cell value would match the predicate, revealing the exact position of the platoon.

One approach to address this vulnerability is to use row-level denial semantics to increase the amount of information that is denied. That is, if disclosing a row containing any cell in a column would violate a deny rule, the system generates an “access denied” exception regardless of whether the column is referenced in the query. These row-level denial semantics would not allow the user in the above example to access the names of the platoon leaders whose locations are prohibited. If the user has no prior knowledge of the database, and the tuples in a table are independent, he cannot acquire knowledge from the database to reveal prohibited values. Deny rules should also ensure that foreign keys do not leak any information about a table’s contents, so access to tuples that have foreign keys to a denied tuple should also be denied. Finally, agencies can deter misuse by proactively auditing past database access [12].

Revealing Existence of Information Deny rules may also reveal the existence or non-existence of prohibited information, which may itself be classified [6]. A database instance-independent approach is proposed in [13] to prevent disclosure on the basis of the query and the policies, regardless of the existence of specific information in the database. Such an approach would also eliminate the non-determinism problem discussed above. However, discerning whether a particular query violates a policy rule for all possible database instances is an intractable problem [13].

3. Sticky Policy Enforcement

In many situations, government agencies are required to consolidate sensitive information from multiple sources, subject to the original disclosure policies. For instance, they must retain source classifications for information that is reproduced, extracted, or summarized [6]. Also, agencies are often required to observe source disclosure policies for information transferred from other agencies [5] or foreign governments [6]. To handle these situations, we propose a sticky policy enforcement model that transfers source disclosure policies with sensitive data and enforces relevant policies in the target database. The following scenario illustrates our model.

3.1 Joint Investigation Scenario

Agencies Blue and Green are involved in a joint investigation to locate an enemy operative known as the

Red Baron. They would like to consolidate information pertaining to this investigation into a single repository maintained by Green. The terms of the collaboration allow members of the joint investigative team to view all data pooled for the investigation. However, Green agents that are not on the Blue-Green taskforce are not allowed to view any of Blue’s data. In addition, Blue agents are not entitled to access any exclusively Green data, but must have access to all Blue data. To satisfy these conditions, Blue and Green must be able to combine data with associated disclosure policies. Green must then be able to enforce these policies over the consolidated data, so that authorized agents can access all data to which they are entitled, but no prohibited data.

3.2 Sticky Policy Model

In our sticky policy model, all data transferred from the source database has policy metadata attached. Upon receiving information, the active enforcement engine of the target database records the attached policy metadata. All data is then associated with applicable policies at the time of insert. Assuming there is no heterogeneity among schemata of the source and target databases (i.e., data and table layouts are the same), a single record can be associated with multiple policies and any combination of policies can be applied to a query [14].

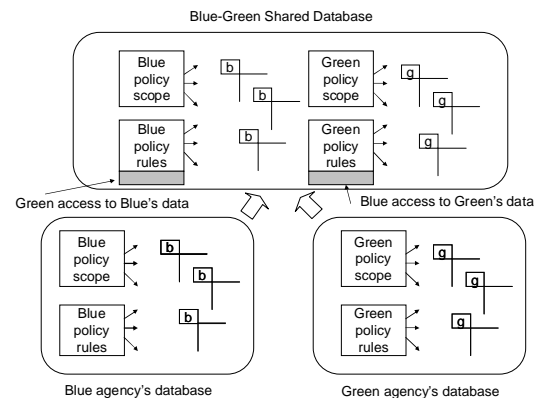


Figure 5. Data Federation Using Sticky Policies

Figure 5 illustrates a simple configuration for the consolidation of Blue and Green taskforce data and policy metadata. For example, each source table is copied into a separate replica table within the target. Some or all of the rows of a source table can be copied to the replica. Any metadata governing the usage of the information, including policy scope and policy rules, are copied along with the data. The policy *scope* identifies policies that govern access to the columns of tables, while the policy *rules* define the conditions under which access to the data is allowed. After copying the data and policy metadata to the replica, Blue’s data remains

accessible to Blue users under Blue’s policy rules and Green’s data remains accessible to Green users under Green policy rules. We create additional policy metadata, shown in grey in Figure 5, to allow each agency access to the other’s taskforce data.

Rather than storing each agency’s tables separately in the replica, we combine the source tables into a single table. We then tag the rows (or columns) contributed by each agency with an additional attribute. Generally, if multiple policies apply to the same column, all policies must agree to disclose the information. But in the present case, this would prevent Blue or Green from accessing any data in a column. For this reason, we propose policy management predicates to extend the policy scope of each agency, as illustrated in Figure 6.

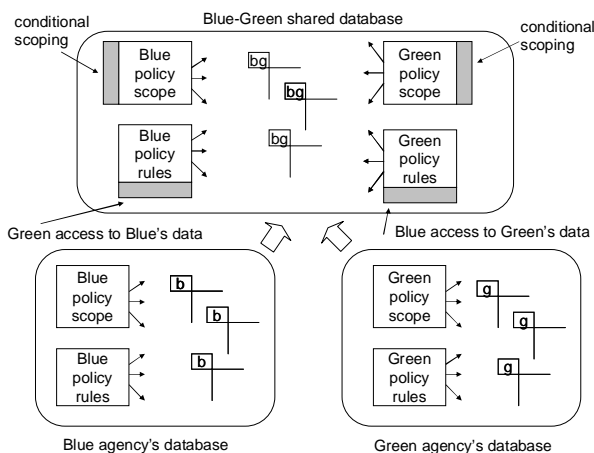


Figure 6. Merging multiple tables with sticky policies

Conditional policy scoping rules can limit the scope of a policy to a subset of the rows in a table or even to specific cells. The subset of cells identified by scoping rules need not be disjoint. We use query modification techniques with conditional scoping to implement policy enforcement. If a policy grants conditional access to cells within a column, then usage of the column in a query is modified as follows:

```
If (policy condition) then column else null
```

In the presence of scoping, the policy condition applies only if the policy applies under the scoping condition. Queries would instead be modified as follows:

```
If (policy scope condition) then if (policy condition) then column else null else column
```

This form of translation would be problematic if there are multiple scopes, so we instead use the form:

```
If (NOT(policy scope condition) or (policy condition)) then column else null
```

Multiple scoped conditions are individual conjuncts as follows:

```
If ((not (s1) or p1) and (not (s2) or p2) and ... and (not (sn) or pn)) then column else null
```

Each individual s_i, p_i are policy scoping and policy rules.

This sticky policy model enhances information sharing and data security by assuring government agencies that all appropriate access and disclosure controls will be applied to consolidated data subsequent to transfer.

4. Information Curation

Another significant technical challenge for government agencies involves curation – the process of applying and modifying disclosure policies to individual information items. Upon the intake of national security information, authorized government officials set classification levels (e.g., top secret, secret, confidential, unclassified) [6]. They may also attach policies establishing the purposes for which the information may be used and recipients to whom it may be disclosed. Over time, authorized officials may modify these classifications and disclosure policies. For example, agencies may lower classification levels to release information to another agency, tactical operations, or an unclassified domain. They may also raise classification levels in some situations or even reclassify information that was previously released. The following scenario shows the importance of fine-grained policy curation in preventing information leakage.

4.1 Reclassification Scenario

Gayle is an Agency Green official who determines the confidentiality levels and appropriate compartments for incoming intelligence information. She also modifies disclosure policies as a result of changing circumstances and agency objectives. Gayle receives two information items from a field agent concerning a potential security threat in Western Europe. The first item is that Lothar, an enemy operative and the Red Baron’s brother, was seen in Vienna, Austria. The second item is that an enemy organization is planning attacks on water towers in Northern France. She classifies both items as “secret.” Gayle later discovers that Agency Blue has separately declassified information about enemy threats to water towers in the city of Arras, France. Aggregation of these three pieces of information would likely reveal the identity of Saunders, an informant who met with Lothar in Vienna to discuss the planned attack on Arras. Because Saunders’s identity is “top secret,” Gayle would now like to reclassify Lothar’s Vienna location to “top secret.”

4.2 Curation Architecture

Our proposed curation system permits authorized officials to create, update, and remove item-specific disclosure policies. Policy updates can raise or lower classification levels for a specific item of information. The “sightings” table in Figure 7 contains classified identities of enemy operatives, their reported locations, and the source of each sighting. The first column of the table identifies the primary key that applies to each row.

| Sightings | | | |
|-----------|------------|----------|----------|
| ID | Operative | Location | Source |
| 1 | Lothar | Vienna | Saunders |
| 2 | Lowenhardt | Chaulnes | Bond |

Figure 7. Sightings Table

Authorized officials can change the policies applicable to information in the sightings table by modifying annotations to the sightings curation table, as shown in Figure 8. For instance, Gayle can raise the classification for the “Vienna” cell in the sightings table from “secret” to “top secret” by changing the annotation of the location column in the sightings-curation table.

| Sightings-curation | | | |
|--------------------|-----------|------------|------------|
| ID | Operative | Location | Source |
| 1 | secret | top secret | top secret |
| 2 | secret | secret | top secret |

Figure 8. Sightings Curation Table

When a user submits a query to Agency Green’s system, the active enforcement system transforms the query to comply with applicable policies. In this case, the policy for the sightings table references the sightings-curation table to make its policy decision. The system only returns information that is appropriate for the user’s clearance level, purpose of access, and intended recipient.

4.3 Auditing Curation Histories

Continuing the above scenario, Gayle is also responsible for reviewing classified information prior to releasing it to lower clearance levels. She discovers that the classified location of Lowenhardt, another Red Baron associate, in Chaulnes, France was recently reported in a European newspaper. Agency Green suspects that its source, Agent Bond, may have disclosed this information, but would like Gayle to audit the curation history of the information to determine whether the

information may have been improperly declassified and released by Green agents.

To address this problem, we propose an auditing system that tracks the curation histories of sensitive information using logs of database updates. The auditing method in [12] relies on query logs and backlog tables to track the circumstances of past database access. We recommend using a similar infrastructure for curation auditing that stores all updates to data and policy tables in curation backlog tables.

| Sightings-curation-backlog | | | | | | |
|----------------------------|-----------|------------|------------|-------|----|----|
| id | Operative | Location | Source | User | Op | Ts |
| 1 | secret | secret | top secret | Alex | I | 1 |
| 2 | secret | secret | top secret | Alex | I | 2 |
| 1 | secret | top secret | top secret | Gayle | U | 3 |
| 2 | unclass. | unclass. | secret | David | U | 4 |

Figure 9. Curation Backlog

The backlog table in Figure 9 shows two versions of the tuple having ID value 2, which concerns the Lowenhardt sighting in Chaulnes. The first version of this tuple was inserted by Alex at time 2. The second version was created at time 4, when David updated the classification of Lowenhardt’s location from secret to unclassified. The backlog table also contains two versions of the tuple having ID value 1, which concerns the Lothar sighting.

We suggest extending the audit expression language used in [12] to declaratively specify the curation information to be audited. Because curation auditing targets general updates, the *before* and *after* values of update operations should be features of the language. The following is an informal example of the syntax and semantics of this language:

Who lowered the classification level of Lowenhardt’s location during the past 12 months?

During current date - 1 year to current date

Audit-curation Sightings-curation s

Where s.id = 2 and before s.location > after.location

The *during* clause of the audit expression specifies a time period for the audit. The *audit-curation* clause stipulates that the sightings-curation table is to be audited. The *where* clause examines who has updated the policy for Agent Bond’s Lowenhardt sighting by decreasing the classification level for the location column of the tuple with id=2. The *before* and *after* images of updated tuples can be accessed using special *before* and *after* keywords. The audit returns the IDs of

logged commands having performed updates that qualify the audit expression. This command log is organized similarly to the query log in [12], which records all queries submitted to the database along with annotations such as the ID of the user submitting the query and the time the query was submitted (see Figure 11). In response to the above audit expression, the system would reveal that David lowered the classification of Lowenhardt’s location to “unclassified” at time 4.

Curation auditing can be accomplished strictly using temporal database extensions. The backlog tables record all versions of a tuple along with the user IDs that generated new versions of each tuple. Although not required for curation auditing, the actual command that caused the update may also be of interest to the auditor since it may reveal patterns of misuse.

4.4 Protection of Metadata

The foregoing policy enforcement and curation auditing technologies are only useful to government agencies if their policy repositories, audit logs, service registries, and directories cannot be modified by unauthorized parties. This metadata should therefore carry the highest levels of security protection. We suggest using the policy enforcement infrastructure presented in this paper to limit access and disclosure of metadata. Indeed, deny and filter rules can be defined over the policy metadata tables themselves. Updates to the policy metadata table can also be recorded in backlog tables and be subject to auditing. Agencies should further employ methods such as those described in [15] [16] to detect tampering with audit logs and accurately restore corrupted metadata.

5. Information Provenance

Government agencies are required to evaluate the quality of intelligence information prior to making decisions or disseminating reports. The provenance of this information, which includes the source and modification history [17] [18], is critical to this evaluation. We suggest using a provenance infrastructure similar to that used in Trio [19] [20] to assist in evaluations of information quality. On top of this infrastructure, we propose a provenance auditing method that enables agencies to track all queries and operations that have accessed a specific information item. This type of auditing allows an agency to determine the extent to which derivative information, assumptions, and conclusions rely on this information. We illustrate these features in the following intelligence scenario.

5.1 Provenance Tracking Scenario

Barbara is an Agency Blue analyst who discovers an enemy threat to an airfield in Southern France. The

credibility of the threat depends the quality and integrity of a few critical items of foreign intelligence. The most important of these is a recent sighting of the Red Baron in Avignon, France. Agency Blue requires agents to verify the accuracy of source material thoroughly before issuing threat warnings. Thus, Barbara would like to evaluate the quality of her intelligence information prior to submitting a final report.

5.2 Provenance Infrastructure

Agency Blue maintains a table of all sightings of enemy operatives reported by friendly spies. The table in Figure 10 contains a list of enemy operatives who have been sighted in France. We borrow the x-tuple format from [20] to present the data. The table lists the name of the person sighted, the location of the sighting, and the source of the information. Each tuple also contains a confidence rating concerning the reliability of the data.

| ID | Sightings (name,location,source) | |
|-----|---------------------------------------|---|
| #1 | (Red-baron, Paris, Ins. Gadget):0.4 | ? |
| #2 | (Red-baron, Paris, Ins. Clouseau):0.4 | ? |
| #3 | (Red-baron, Avignon, Mata Hari):1.0 | ? |
| ... | ... | |

Figure 10. Sightings Table

Figure 10 shows the first three x-tuples in the sightings table. The first tuple states that Inspector Gadget reported seeing the Red Baron in Paris and has 40% confidence in his observation. The question mark (?) beside the tuple represents the alternative unknown value, which by default, has the remaining 60% confidence. The second tuple shows a similar sighting by Inspector Clouseau, also with 40% confidence. The third represents Mata Hari’s sighting of the Red Baron in Avignon with 100% confidence. This contradicts the two Paris sightings.

We track the provenance of information in the database by recording all SQL commands in a command log.

| SQL-Commands | | | | | |
|--------------|-------|--------|--------|-----|----------------------------------|
| c-id | user | ts-beg | ts-end | ... | command-string |
| ... | | | | | |
| #7 | UserA | T1 | T1.7 | | insert into sightings values ... |
| #8 | UserB | T2 | T2.5 | | insert into sightings values ... |
| #9 | UserB | T3 | T3.6 | | insert into sightings values ... |
| #10 | UserA | T4 | T4.5 | | insert into sightings values ... |
| #11 | UserX | T5 | T5.4 | | insert into suspects select ... |
| #12 | UserY | T6 | T6.2 | | update suspects set location ... |
| ... | | | | | |

Figure 11. SQL Command Log

Figure 11 illustrates how the command log records individual commands (c-id), the user issuing each command, the time the command began and ended (ts-beg and ts-end), and the actual command string submitted to the system. For provenance tracking, we also use a similar backlog infrastructure to the one presented in section 4.3 for curation auditing.

Agency Blue can then use the sightings reports and confidence levels to generate a table of most likely suspects. Figure 12 displays how this suspects table is computed. The insertion selects sightings with the highest levels of confidence. We use the language extensions presented in [20] to extract the confidence associated with each tuple. The *conf* function returns the confidence level of an x-tuple alternative. Given the selection in command #11, the Suspects table reflects that Red Baron is located in Avignon because that location has a confidence level of 1.0. The confidence he is in Paris is $0.64 = 0.4 + 0.4 - (0.4 * 0.4)$ and computed from the confidence levels of two tuples confirming that location [19].

| ID | Suspects (name, location) |
|------|---------------------------|
| #101 | (Red-baron, Avignon): 1.0 |
| ... | ... |

```
SQL-COMMAND - #11
insert into suspects
select      gen-id(), s.name, s.location
from        Sightings s
groupby    s.name, s.location
having conf(s) >= all(
select      conf (s2)
from        Sightings s2
where       s.name = s2.name
groupby    s2.location
```

Figure 12. Insert Command Followed by an Update

5.3 Provenance Auditing

Some time later, Agency Blue discovers that Mata Hari is a double agent and has been providing false information. Aside from having used Mata Hari’s information in command #11 to update the Suspects table, Blue may have used misleading information provided by Mata Hari in many other queries and operations. This may affect the validity of further assumptions and conclusions. The agency would like to audit usage of information supplied by Mata Hari so it can assess the damage and take corrective measures.

We propose to extend the auditing infrastructure presented in [12], and discussed with regard to curation in Section 4.3, to also audit information provenance. The audit will track all logged queries or commands that access tuples containing information specified by an

audit expression. It will also track logged queries and commands having accessed tuples in any table derived from these tuples. The following is an example of an audit expression for information provenance:

```
Who has accessed any of the information supplied
by Mata Hari?

Audit-provenance Sightings s
where s.source = 'Mata Hari'
```

The *audit-provenance* clause stipulates that the Sightings table is being audited, and that the provenance of information stemming from the sightings table must also be considered in the audit. The *where* clause limits the scope of the provenance to all queries and commands that accessed information for which Mata Hari was the source. The audit returns the IDs of logged commands (c-id attribute values in Figure 11) having directly or indirectly accessed information she provided. Direct access means that a logged query or command referenced the sightings table itself, including those tuples for which source=‘Mata Hari’. Indirect access refers to any tuple in any relation that was generated by a logged query or command that had direct or indirect access to tuples specified by the audit. In response to the above audit expression, the system would disclose all commands and queries that have directly or indirectly relied on misinformation given by Mata Hari.

Our audit algorithm first selects the set of candidate logged queries and commands and finds those that have directly accessed the information specified by the audit. Thereafter, the algorithm iteratively selects the set of candidate logged queries and commands that accessed tuples generated by commands at the previous iteration. Once visited, commands are only revisited in subsequent iterations if they are not already in the suspicious set of commands. The iterative process ends when no new suspicious commands can be derived.

The scope of tuples generated by a logged command may be larger than necessary if we consider all tuples resulting from a general update command. For example, suppose that an insert command has selected all tuples, including those representing information supplied by Mata Hari, from the sightings table and inserted them into another table R. Only those tuples in R that were generated from tuples in the sightings table corresponding to Mata Hari should be considered for indirect access. The audit algorithm must therefore consider the logged command that generated the tuples when auditing provenance. Although we have described the basic features of a provenance auditing algorithm, developing efficient algorithms remains a research topic.

6. Conclusion

We have outlined a vision for policy-based management and sharing of information among defense, intelligence and law enforcement agencies. We demonstrated how enforcement of fine-grained access and disclosure policies can ensure appropriate access to, and sharing of, sensitive information. We also described methods of curating item-level disclosure policies and auditing updates to these policies. Finally, we proposed a system of provenance auditing to evaluate the quality of source information and any derivative information. We trust that our approach will be useful in addressing important information sharing problems in government agencies.

Acknowledgements

We thank Arnon Rosenthal for his invaluable help in identifying research problems and scenarios in the government sector to motivate our work. We also thank Tyrone Grandison and Rakesh Agrawal for their constructive discussions and suggestions.

References

-
- [1] Homeland Security Act of 2002 (<http://files.findlaw.com/news.findlaw.com/cnn/docs/terrorism/hsa2002.pdf>).
 - [2] Intelligence Reform and Terrorism Prevention Act of 2004, Section 1016 (<http://files.findlaw.com/news.findlaw.com/cnn/docs/terrorism/irtpa2004.pdf>).
 - [3] Executive Order Strengthening the Sharing of Terrorism Information to Protect Americans, August 24, 2004 (<http://www.whitehouse.gov/news/releases/2004/08/20040827-.html>).
 - [4] Executive Order Further Strengthening the Sharing of Terrorism Information to Protect Americans, October 25, 2005 (<http://www.whitehouse.gov/news/releases/2005/10/20051025-.html>).
 - [5] Executive Memorandum for the Heads of Executive Departments and Agencies: Guidelines and Requirements in Support of the Information Sharing Environment, December 16, 2005 (www.whitehouse.gov/news/releases/2005/12/20051216-10.html).
 - [6] Further Amendment to Executive Order 12958, As Amended, Classified National Security Information, March 25, 2003, (www.whitehouse.gov/news/releases/2003/03/20030325-11.html).
 - [7] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu, Hippocratic Databases. *Proc. of the 28th Int'l Conf. on Very Large Databases*. Hong Kong, China, August 2002.
 - [8] K. LeFevre, R. Agrawal, V. Ercegovac, R. Ramakrishnan, Y. Xu, and D. DeWitt, Limiting Disclosure in Hippocratic Databases. *Proc. of the 30th Int'l Conf. on Very Large Databases*, Toronto, Canada, August 2004.
 - [9] Multi-level Security in the Department of Defense: The Basics (<http://nsi.org/Library/Compsec/sec0.html>).
 - [10] A. Rosenthal, M. Winslett, Security of Shared Data in Large Systems: State of the Art and Research Directions. *Proc. of the 30th Int'l Conf. on Very Large Databases*, Toronto, Canada, August 2004.
 - [11] H. Pirahesh, J. Hellerstein, and W. Hasan, Extensible/Rule Based Query Rewrite Optimization in Starburst. *Proc. of the ACM SIGMOD Int'l Conf. on Management of Data*, San Diego, California, USA, June 1992.
 - [12] R. Agrawal, R. Bayardo, C. Faloutsos, J. Kiernan, R. Rantzaou, and R. Srikant, Auditing Compliance with a Hippocratic Database.

-
- Proc. of the 30th Int'l Conf. on Very Large Databases*, Toronto, Canada, August 2004.
 - [13] G. Miklau and Dan Suciu, A Formal Analysis of Information Disclosure in Data Exchange. *Proc. of the ACM SIGMOD Int'l Conf. on Management of Data*, Paris, France, June 2004.
 - [14] IBM Hippocratic Database Active Enforcement Installation Guide, Version 1.0 (<http://www.almaden.ibm.com/software/projects/iis/hdb/Publications/papers/HDBEnforcementUserGuide.pdf>).
 - [15] R. Snodgrass, S. Yao, and C. Collberg, Tamper Detection in Audit Logs. *Proc. of the 30th Int'l Conf. on Very Large Databases*, Toronto, Canada, August 2004.
 - [16] Markle Foundation, Implementing a Trusted Information Sharing Environment: Using Immutable Audit Logs to Increase Security, Trust and Accountability, New York, February 2006.
 - [17] P. Buneman, S. Khanna, and W. Tan, Why and Where: A Characterization of Data Provenance. *Proc. of the Int'l Conf. on Database Theory*, London, UK, January 2001.
 - [18] Z. Ives, D. Halevy, D. Weld, Adapting to Source Properties in Processing Data Integration Queries. *Proc. of the ACM SIGMOD Int'l Conf. on Management of Data*, Paris, France, June 2004.
 - [19] O. Benjelloun, A. Das Sarma, A. Halevy, and J. Widom, ULDBs: Databases with Uncertainty and Lineage. Stanford University Technical Report, December 2005.
 - [20] O. Benjelloun, A. Das Sarma, C. Hayworth, and J. Widom, An Introduction to ULDBs and the Trio System. *IEEE Data Engineering Bulletin*, March 2006.