PURDUE UNIVERSITY

Data Protection from Insider Threats *Concepts and Research Issues*

Elisa Bertino
CS Department, Cyber Center, and CERIAS
Purdue University

CERIAS®
Center for Education and Research in Information Assurance and Security

PURDUE UNIVERSITY

Insider Threat *Motivations and Challenges*

- Mission-critical information = High-value target
- Threatens US and other Government organizations and large corporations
- Probability is low, but impact is <u>severe</u>
- Types of threat posed by malicious insiders
  - Denial of service
  - **Data leakage and compromise of confidentiality**
  - Compromise of integrity
- High complexity of problem
  - Increase in sharing of information, knowledge
  - Increased availability of corporate knowledge online
  - “Low and Slow” nature of malicious insiders

PURDUE UNIVERSITY

Some Data

2010 CyberSecurity Watch Survey (*) (CSO Magazine in cooperation with US Secret Service, CMU CERT and Deloitte)

- 26% of attacks on survey respondents' organizations were from insiders

  (as comparison: 50% from outsiders, 24%unknown)

- Of these attacks, the most frequent types are:
  - Unauthorized access to/ use of information, systems or networks 23%
  - Theft of other (proprietary) info including customer records, financial records, etc.   15%
  - Theft of Intellectual Property   16%
  - *Unintentional exposure of private or sensitive information 29%*

(*) http://www.sei.cmu.edu/newsitems/cyber_sec_watch_2010_release.cfm

PURDUE UNIVERSITY

Protection from Insider Threat - IP Theft

https://www.cert.org/blogs/insider_threat/2013/12/theft_of_ip_by_insiders.html

Based on 103 IP theft cases recorded in the MERIT Database (since 2001)

- Industry sector in which IP theft occurred more frequently

- Information Technology	35%
- Banking and Finance	13%
- Chemical	12%
- Critical Manufacturing	10%

- Majority of insider IP theft cases occurred onsite (70% onsite as opposed 18% remotely)
- Financial impact (known only for 35 of the 103 cases)
  - Over 1M USD in 48% of cases

What is an insider?

- We define an “insider” to be any individual that has currently or has previously had authorized access to information of an organization
- Other definitions do not consider individuals who no longer have access as insiders
- The advantage of the this definition includes also individuals not any longer part of the organization who may use their knowledge of the organization as part of an attack

Definitions

The President's National Infrastructure Advisory Council defines the insider threat as follows:

> "The insider threat to critical infrastructure is one or more individuals with the access or inside knowledge of a company, organization, or enterprise that would allow them to exploit the vulnerabilities of that entity's security, systems, services, products, or facilities with the intent to cause harm."
>
> "A person who takes advantage of access or inside knowledge in such a manner commonly is referred to as a "malicious insider.""

Definitions from FEMA – Emergency Management Institute
http://www.training.fema.gov/emi.aspx

The Scope of Insider Threats

Insider threats can be accomplished through either physical or cyber means and may involve any of the following:

Threat	Involves
Physical or information-technology sabotage	Modification or damage to an organization’s facilities, property, assets, inventory, or systems with the purpose of harming or threatening harm to an individual, the organization, or the organization’s operations
Theft of intellectual property	Removal or transfer of an organization’s intellectual property outside the organization through physical or electronic means (also known as economic espionage)
Theft or economic fraud	Acquisition of an organization’s financial or other assets through theft or fraud
National security espionage	Obtaining information or assets with a potential impact on national security through clandestine activities

PURDUE UNIVERSITY

Examples of Actual Incidents

Sector	Incidents
Chemical	**Theft of intellectual property.** A senior research and development associate at a chemical manufacturer conspired with multiple outsiders to steal proprietary product information and chemical formulas using a USB drive to download information from a secure server for the benefit of a foreign organization. The conspirator received $170,000 over a period of 7 years from the foreign organization.
Critical Manufacturing	**Physical sabotage.** A disgruntled employee entered a manufacturing warehouse after duty hours and destroyed more than a million dollars of equipment and inventory.
Defense Industrial Base	**National security threats.** Two individuals, working as defense contractors and holding U.S. Government security clearances, were convicted of spying for a foreign government. For over 20 years, they stole trade and military secrets, including information on advanced military technologies. **Information-technology sabotage.** A system administrator served as a subcontractor for a defense contract company. After being terminated, the system administrator accessed the system and important system files, causing the system to crash and denying access to over 700 employees.

PURDUE UNIVERSITY

Organizational Factors that Embolden Malicious Insiders

Access and Availability

- Ease of access to materials and information
- Ability to exit the facility or network with materials or information

Policies and Procedures

- Undefined or inadequate policies and procedures
- Inadequate labeling
- Lack of Training

Time Pressure and Consequences

- Rushed employees
- Perception of lack of consequences

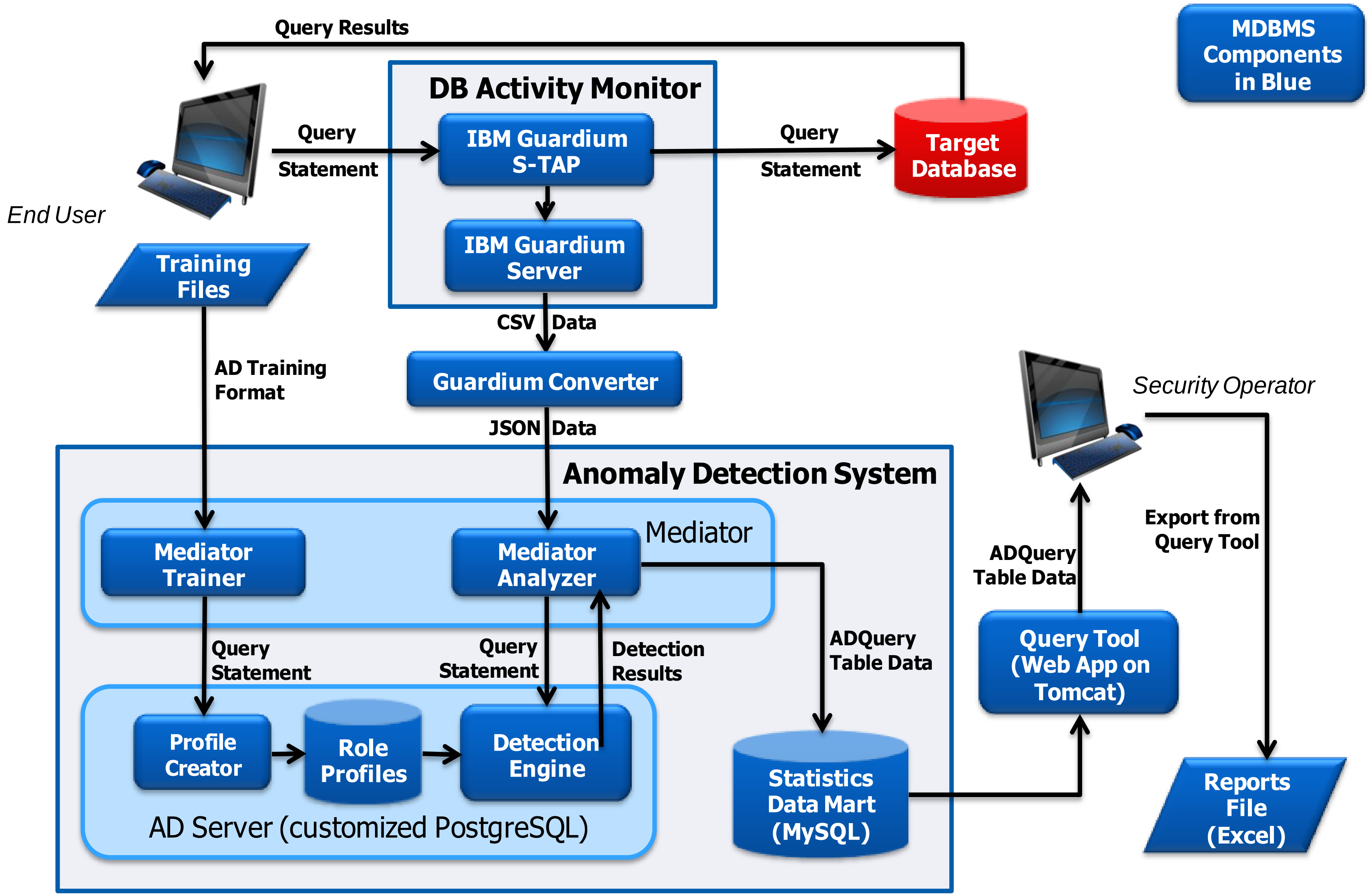
Remediation: Some Ideas

- Distribute trust amongst multiple parties to force collusion
  - Most insiders act alone
- Question trust assumptions made in computing systems
  - Treat the LAN like the WAN
- Create profiles of data access and monitor data accesses to detect anomalies

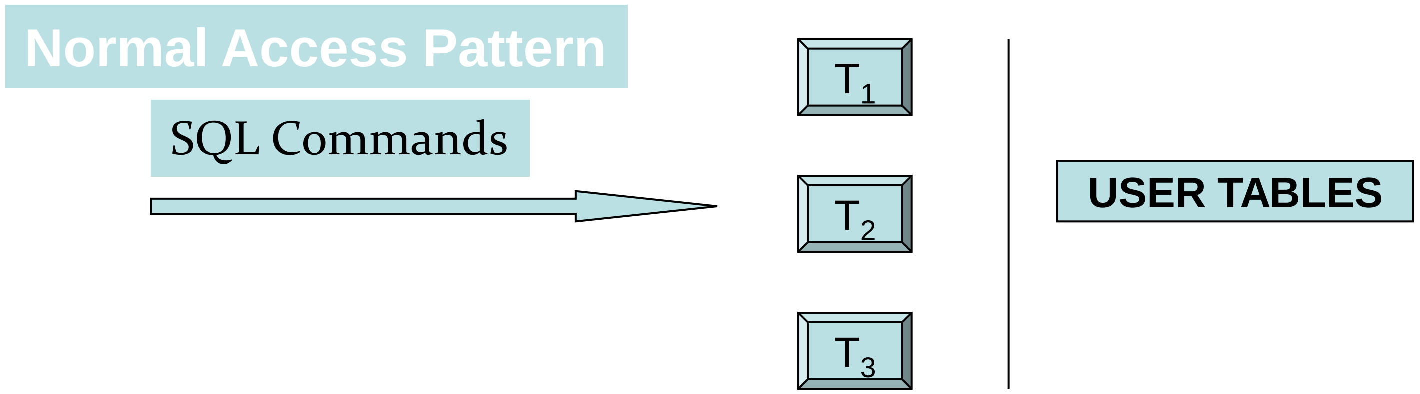
PURDUE UNIVERSITY

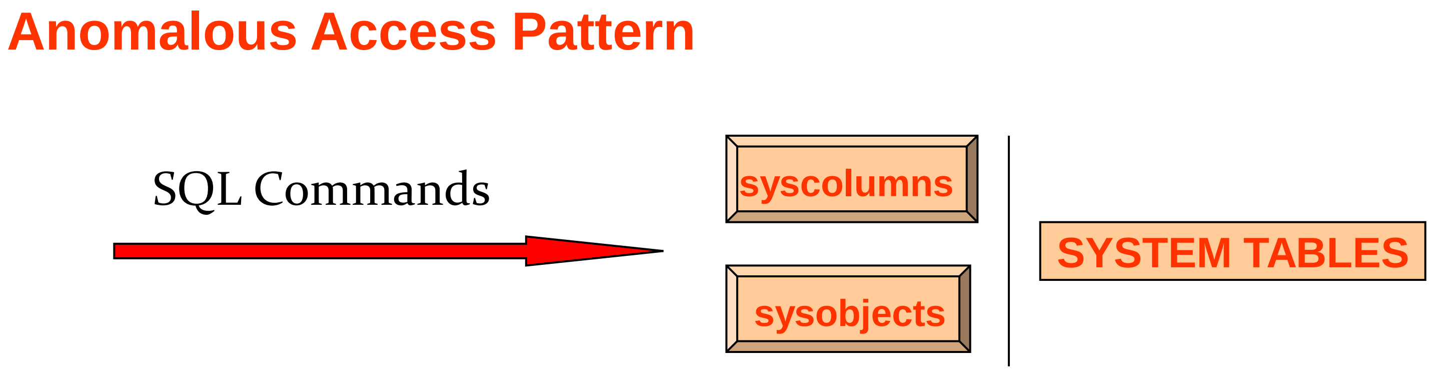
Anomaly Detection for Databases

System Architecture



Anomalous Access Pattern *Example*





PURDUE UNIVERSITY

SQL Query Representation
Key idea

- Extract access pattern from query syntax
- Build profiles at different granularity levels
  - Coarse
  - Medium
  - Fine

PURDUE UNIVERSITY

Coarse Quiplet: example

Schema

T1 : {a1,b1,c1} T2 : {a2,b2,c2} T3 : {a3,b3,c3}

Query

```
SELECT  T1.a1, T1.c1, T2.c2 FROM T1, T2,T3
WHERE T1.a1 = T2.a2 AND T1.a1  =T3.a3
```

Field	Value
Command	SELECT
Num Projection Tables	2
Num Projection Columns	3
Num Selection Tables	3
Num Selection Columns	3

PURDUE UNIVERSITY

Medium Quiplet: example

Schema

T1 : {a1,b1,c1}   T2 : {a2,b2,c2}   T3 : {a3,b3,c3}

Query

```
SELECT  T1.a1, T1.c1, T2.c2 FROM T1, T2,T3
WHERE T1.a1 = T2.a2 AND T1.a1  =T3.a3
```

Field	Value
Command	SELECT
Projection Tables	[1 1 0]
Projection Columns	[2 1 0]
Selection Tables	[1 1 1]
Selection Columns	[1 1 1]

Fine Quiplet: example

Schema T1 : {a1,b1,c1} T2 : {a2,b2,c2} T3 : {a3,b3,c3}

Query SELECT T1.a1, T1.c1, T2.c2 FROM T1, T2,T3
WHERE T1.a1 = T2.a2 AND T1.a1 =T3.a3

Field	Value
Command	SELECT
Projection Tables	[1 1 0]
Projection Columns	[[1 0 1] [0 0 1] [0 0 0]]
Selection Tables	[1 1 1]
Selection Columns	[[1 0 0] [1 0 0] [1 0 0]]

Supervised Case
Key Ideas

- Associate each query with a role
- Build profiles per role
- Train a classifier with role as the class
- Declare a request as anomalous if classifier predicted role does not match the actual role

Supervised Case
Naïve Bayes

- Low computational complexity
- Ease of implementation
- Works surprisingly well in practice even if the attributes independence condition is not met

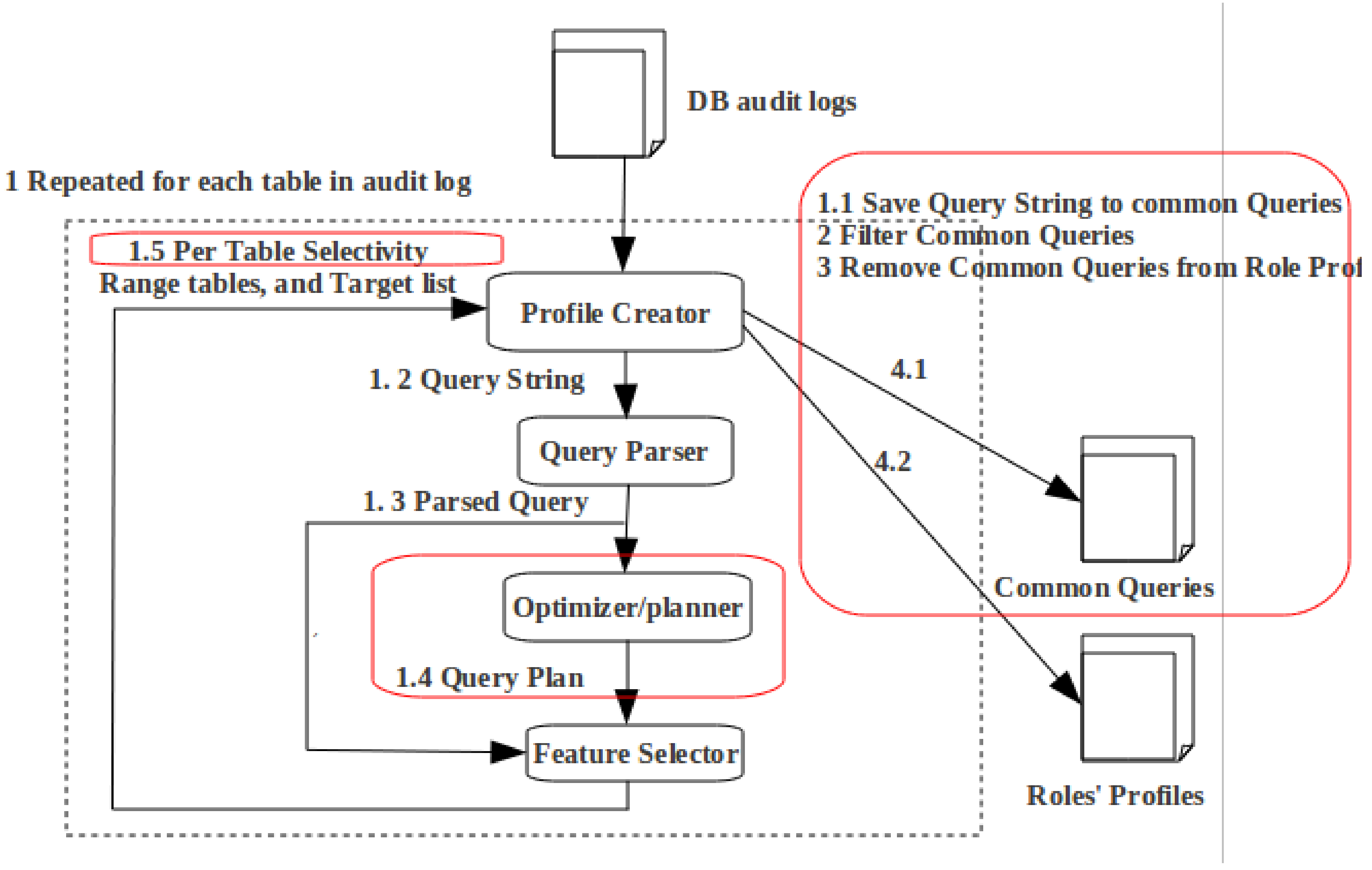
Un-supervised Case

- Associate every query with a user (not role)
- Use clustering algorithms to partition training data into clusters
- Map every training query to its representative cluster

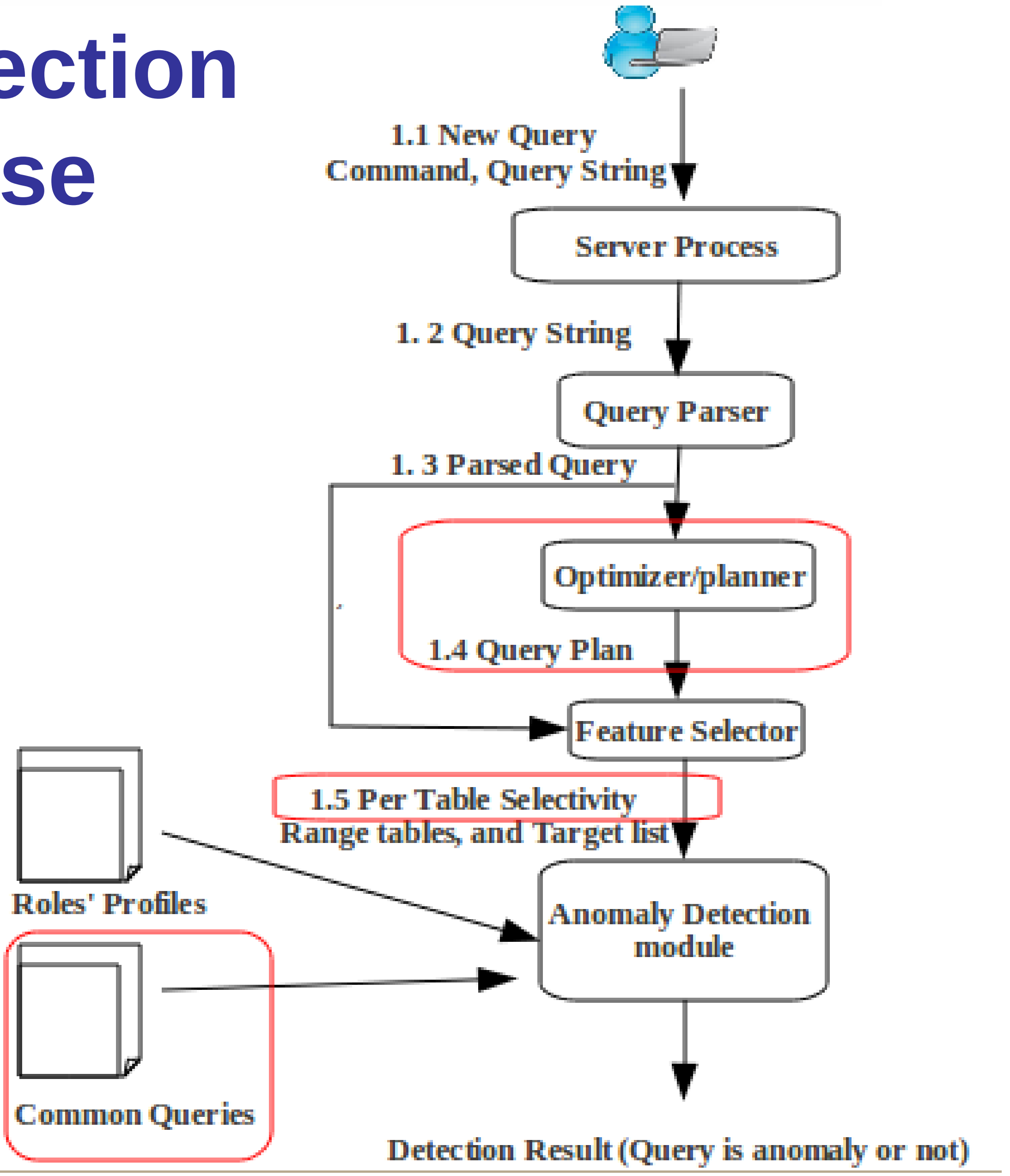
Enhancing Profiles with Data Centric Information

- Profiles can be refined by including an additional feature that keeps track of the amount of data returned by queries
- Two possible approaches
  - Execute the query and inspect the results
  - *Estimate the query selectivity before executing the query*
- We adopt the second approach and leverage the query optimizer for the estimation of the query selectivity for each table in the query
- The selectivity of the query is the portion of the table that is appear in the result
  - ❑ Range: [0 … 1]
  - ❑ e.g., query with sel = 0.2 will retrieve 20% of the table

Training Phase



Detection Phase



How to Profile and Monitor Application Programs with respect their Database Accesses?

Our Solution: DetAnom

- DetAnom consists of two phases:
  - the *profile creation phase* and the *anomaly detection phase*.
- Profile creation phase:
  - we create a profile of the application program to succinctly represent the application's normal behavior in terms of its interaction with the database.
  - for each query, we create a signature and also capture the corresponding constraints that the application program must satisfy to submit the query.
  - major issue: exploring all possible execution paths of an application program requires identifying all possible combinations of program inputs
    - to make our profiling technique close to complete and accurate, we adopt concolic testing that generates program inputs automatically to cover all execution paths.
- Anomaly detection phase:
  - whenever a query is issued,
    - mismatch in query signature or the constraint -> anomalous
    - otherwise -> legitimate
  - however, depending on the number of paths covered in concolic execution, the *anomaly detection phase* follows either `*strict*' or `*flexible*' policy.

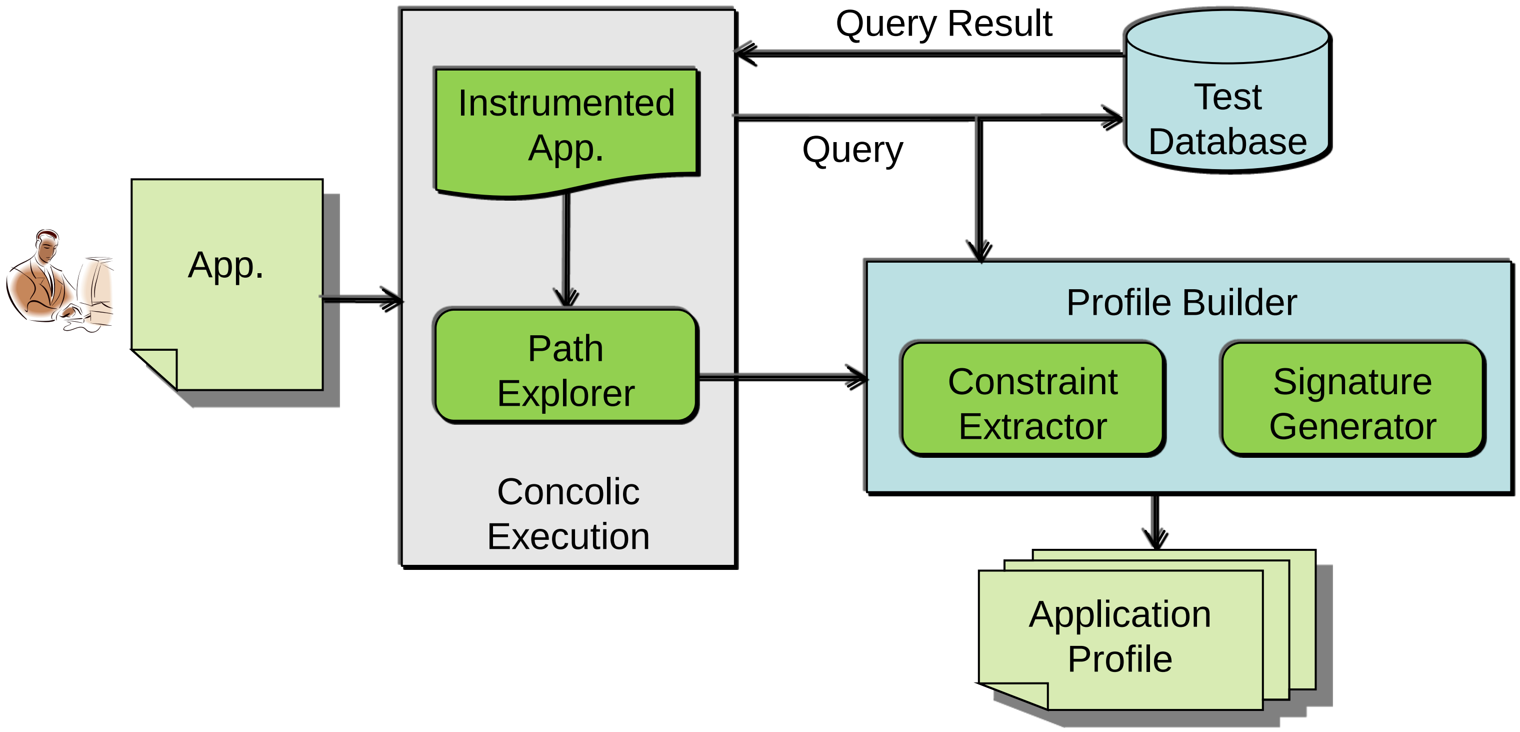
PURDUE UNIVERSITY

Concolic Testing

- Concolic testing is a program analysis technique that explores all possible execution paths by running the program both symbolically and concretely.
- The program to be tested is first concretely executed with some initial random inputs.
- Then the concolic execution engine examines the branch conditions along the executed path’s control-flow and uses a decision procedure to find inputs that reverse the branch conditions.
- This process is repeated to discover more inputs that trigger new control-flow paths, and thus more program states are tested.
- The concolic execution uses a bounded depth-first search (bounded DFS) to explore the execution paths.
  – tradeoff between the exploration of more execution paths and termination of the current path if its length is significantly long.

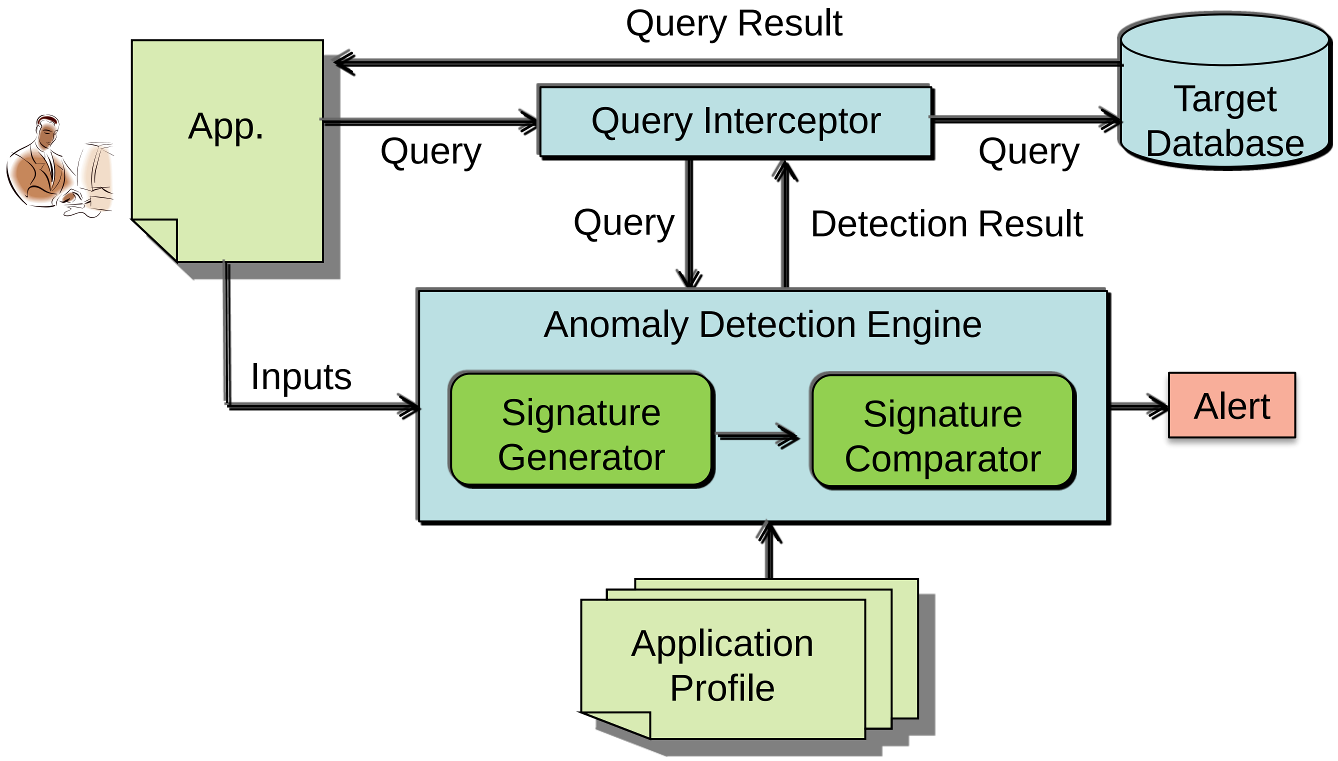
Profile Creation Phase

The application program is given as input to the concolic execution module



Anomaly Detection Phase

Queries issued by the application program are first verified by the anomaly detection engine (ADE) and then forwarded to the target database



Signature Generation

SQL query structure:

```
SELECT[DISTINCT] {TARGET-LIST} FROM {RELATION-LIST} WHERE
{QUALIFICATION}
```

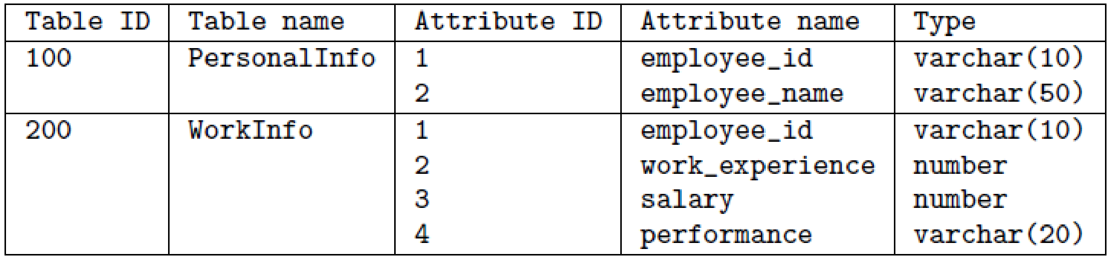


Table ID	Table name	Attribute ID	Attribute name	Type
100	PersonalInfo	1	employee_id	varchar(10)
		2	employee_name	varchar(50)
200	WorkInfo	1	employee_id	varchar(10)
		2	work_experience	number
		3	salary	number
		4	performance	varchar(20)

Example: SELECT employee_id, work_experience FROM WorkInfo
WHERE work_experience > 10

Signature: {1, {{200, 1}, {200, 2}}, {200}, {{200, 2}}, 1}

- The leftmost 1 represents the SELECT command.
- {200, 1}, and {200, 2} represent the IDs of attributes employee_id and work_experience, respectively.
- 200 represents the ID of the table WorkInfo.
- {200, 2} represents the attribute used in the WHERE clause, i.e, work_experience.
- The rightmost 1 corresponds to the number of predicates in WHERE clause.

PURDUE UNIVERSITY

Constraint Extraction

```
public static void salaryAdjustment(int profit, int
     investment){
    Statement s;
    ...
    int employee_count = 0;
    if(profit >= 0.5 * investment){
        String query1 = "SELECT employee_id,
             work_experience FROM WorkInfo WHERE
             work_experience > 10";
        resultSet1 = s.executeQuery(query1);
        resultSet1.last();
        if(resultSet1.getRow() > 100){
        String query3 = "SELECT employee_id FROM
             WorkInfo WHERE work_experience > 10 AND
             performance = 'good'";
        resultSet3 = s.executeQuery(query3);
        ... // do other operations
        } else{
            String query2 = "UPDATE WorkInfo SET salary
                 = salary * 1.2";
            s.executeUpdate(query2);
    }else{
        String query4 = "SELECT p.employee_name FROM
             PersonalInfo p, WorkInfo w WHERE
             performance = 'poor' AND p.employee_id =
             w.employee_id";
        resultSet2 = s.executeQuery(query4);
        ... // do other operations
    }
}
```

c_1: `1.0 x1 - 0.5 x2 >= 0.0,`
Here, x1 and x2 correspond to variables `profit` and `investment`, respectively.

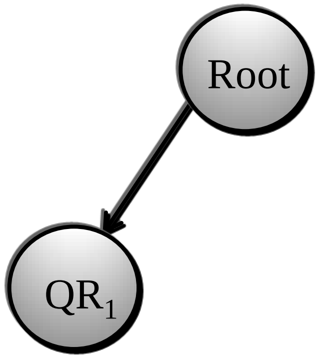
PURDUE UNIVERSITY

Profile Creation

```
public static void salaryAdjustment(int profit, int
     investment){
    Statement s;
    ...
    int employee_count = 0;
    if(profit >= 0.5 * investment){
        String query1 = "SELECT employee_id,
             work_experience FROM WorkInfo WHERE
             work_experience > 10";
        resultSet1 = s.executeQuery(query1);
        resultSet1.last();
        if(resultSet1.getRow() > 100){
        String query3 = "SELECT employee_id FROM
             WorkInfo WHERE work_experience > 10 AND
             performance = 'good'";
        resultSet3 = s.executeQuery(query3);
        ... // do other operations
        } else{
            String query2 = "UPDATE WorkInfo SET salary
                 = salary * 1.2";
            s.executeUpdate(query2);
    }else{
        String query4 = "SELECT p.employee_name FROM
             PersonalInfo p, WorkInfo w WHERE
             performance = 'poor' AND p.employee_id =
             w.employee_id";
        resultSet2 = s.executeQuery(query4);
        ... // do other operations
    }
}
```

c_1: 1.0 $x1$ - 0.5 $x2$ >= 0.0
sig($query_1$)= {1, {{200, 1}, {200, 2}}, {200}, {{200, 2}}, 1}
QR_1 = <sig($query_1$), c_1>

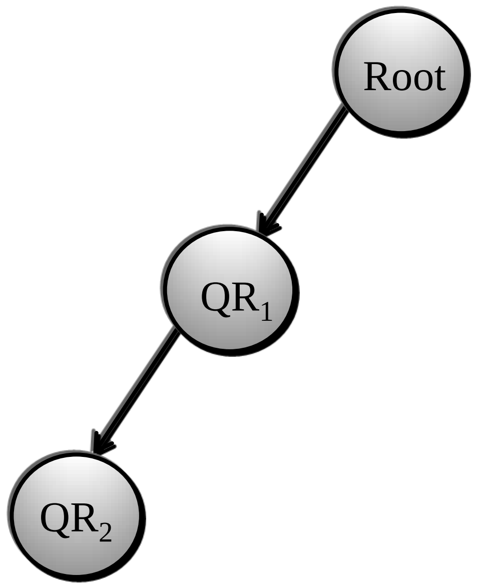
Application Profile



Profile Creation

```
public static void salaryAdjustment(int profit, int
     investment){
    Statement s;
    ...
    int employee_count = 0;
    if(profit >= 0.5 * investment){
        String query1 = "SELECT employee_id,
             work_experience FROM WorkInfo WHERE
             work_experience > 10";
        resultSet1 = s.executeQuery(query1);
        resultSet1.last();
        if(resultSet1.getRow() > 100){
        String query3 = "SELECT employee_id FROM
             WorkInfo WHERE work_experience > 10 AND
             performance = ’good’";
        resultSet3 = s.executeQuery(query3);
        ... // do other operations
        } else{
            String query2 = "UPDATE WorkInfo SET salary
                 = salary * 1.2";
            s.executeUpdate(query2);
    }else{
        String query4 = "SELECT p.employee_name FROM
             PersonalInfo p, WorkInfo w WHERE
             performance = ’poor’ AND p.employee_id =
             w.employee_id";
        resultSet2 = s.executeQuery(query4);
        ... // do other operations
    }
}
```

Application Profile



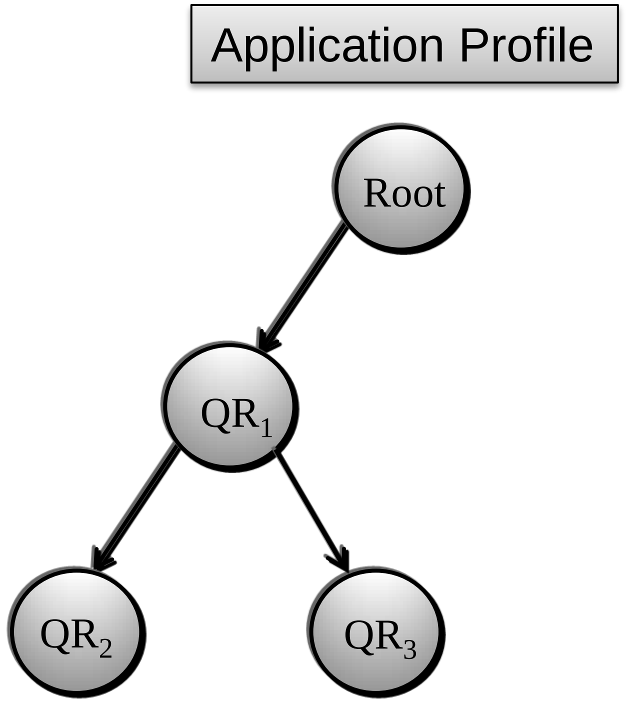
c_2: x3 ≤ 100.0
sig(query2)={2, {{200, 3}}, {200}, {∅}, 0}
QR_2 = <sig($query_2$), c_2>

PURDUE UNIVERSITY

Profile Creation

```
public static void salaryAdjustment(int profit, int
     investment){
    Statement s;
    ...
    int employee_count = 0;
    if(profit >= 0.5 * investment){
        String query1 = "SELECT employee_id,
             work_experience FROM WorkInfo WHERE
             work_experience > 10";
        resultSet1 = s.executeQuery(query1);
        resultSet1.last();
        if(resultSet1.getRow() > 100){
        String query3 = "SELECT employee_id FROM
             WorkInfo WHERE work_experience > 10 AND
             performance = 'good'";
        resultSet3 = s.executeQuery(query3);
        ... // do other operations
        } else{
            String query2 = "UPDATE WorkInfo SET salary
                 = salary * 1.2";
            s.executeUpdate(query2);
    }else{
        String query4 = "SELECT p.employee_name FROM
             PersonalInfo p, WorkInfo w WHERE
             performance = 'poor' AND p.employee_id =
             w.employee_id";
        resultSet2 = s.executeQuery(query4);
        ... // do other operations
    }
}
```

c_3: x3 > 100.0
$sig(query_3) = \{1, \{\{200, 1\}\}, \{200\}, \{\{200, 2\}, \{200, 4\}\}, 2\}$
$QR_3 = \langle sig(query_3), c_3 \rangle$

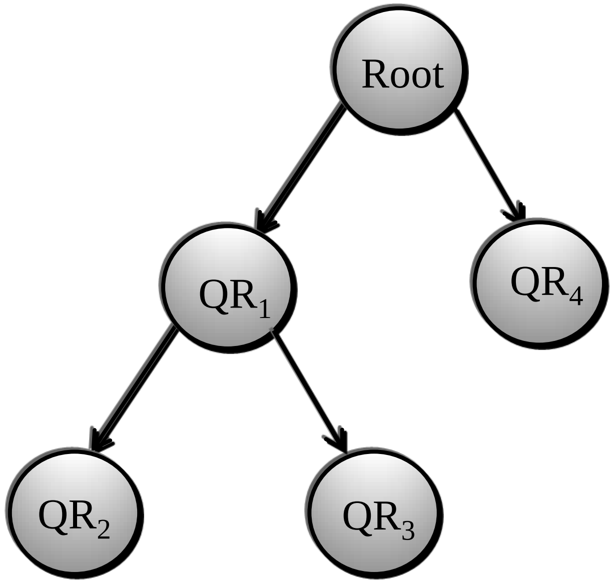


PURDUE UNIVERSITY

Profile Creation

```
public static void salaryAdjustment(int profit, int
     investment){
    Statement s;
    ...
    int employee_count = 0;
    if(profit >= 0.5 * investment){
       String query1 = "SELECT employee_id,
            work_experience FROM WorkInfo WHERE
            work_experience > 10";
       resultSet1 = s.executeQuery(query1);
       resultSet1.last();
       if(resultSet1.getRow() > 100){
       String query3 = "SELECT employee_id FROM
            WorkInfo WHERE work_experience > 10 AND
            performance = 'good'";
       resultSet3 = s.executeQuery(query3);
       ... // do other operations
       } else{
          String query2 = "UPDATE WorkInfo SET salary
               = salary * 1.2";
          s.executeUpdate(query2);
    }else{
       String query4 = "SELECT p.employee_name FROM
            PersonalInfo p, WorkInfo w WHERE
            performance = 'poor' AND p.employee_id =
            w.employee_id";
       resultSet2 = s.executeQuery(query4);
       ... // do other operations
    }
}
```

Application Profile



c_4: 1.0 $x1$ − 0.5 $x2$ < 0.0
sig($query_4$)= {1, {{100, 2}}, {100, 200}, {{200, 4}, {100, 1}, {200, 1}}, 2}
QR_4 = <sig($query_4$), c_4>

Anomaly Detection

- When the application program starts executing, the ADE module sets the root node of the AP as the parent node (v_p).
- Upon receiving a query along an execution path of the program, the ADE:
  - considers all the children of v_p as candidate nodes
  - takes the inputs from the executing application
  - verifies for each candidate node whether the inputs satisfy the constraint in QR_i.
  - lets the SG sub-module generate the signature of the received query and the SC sub-module compare it with the signature stored in QR_i, i.e., sig($query_i$).
  - checks if the inputs satisfy constraint c_i of a candidate QR_i
  - expects the program to execute the query associated with the satisfied c_i.
- If the signatures match, the query is considered as legitimate.
  - the verification outcome is then passed to the QI module which then sends the legitimate query to the target database for execution.
- If the signatures mismatch, the query is considered as anomalous.
  - the *SC* sub-module raises a flag and the *ADE* takes next steps based on either `*strict*' or `*flexible*‘ policies.

Strict & Flexible Policies

- If the length of an execution path exceeds the depth limit (i.e., bound) of DFS set by the concolic execution module:
  - the concolic execution stops that particular execution at that depth level, and searches for new paths.
  - it may leave some large execution paths unexplored that may contain queries.
- Strict Policy:
  - we set the bound of DFS high enough so that the concolic execution can explore almost all possible paths of the program and cover all the branches that are estimated statically.
  - as a result, the profile of the application program gets close to be complete
  - the ADE module becomes confident enough to distinguish between legitimate and anomalous queries.
  - when the signature of an input query does not match:
    - the ADE module identifies that query as anomalous with high confidence and raises an alert signal.
    - this information is then forwarded to the QI module.

PURDUE UNIVERSITY

Strict & Flexible Policies

- Flexible Policy:
  - if the bound of DFS for concolic execution is not high enough:
    - the profile creation phase may leave some large paths unexplored.
    - in this case, if a query is issued by the program along an execution path, and the SC does not find a match for its signature, the ADE raises a flag for that query.
  - now if a query is flagged for more than k times (k is a threshold set in the ADE module):
    - this module raises an alert signal, and requests the security officer (or some other trusted user) to check if the query is actually anomalous or legitimate.
  - if the query is assessed as anomalous:
    - it is kept in the blacklist of the QI so that future occurrences of such query are blocked automatically.
  - if the query is assessed as legitimate, the AP is updated accordingly with its QR.

Conclusion

- We have designed and implemented an anomaly detection mechanism that is able to identify anomalous queries resulting from previously authorized applications.
- Our mechanism builds close to accurate profile of the application program and checks at run-time incoming queries against that profile.
- In addition to anomaly detection, our DetAnom mechanism is capable of detecting any injections or modifications to the SQL queries, e.g., SQL injection attacks.
- DetAnom results in low run-time overhead and high accuracy in detecting anomalous database accesses.

Questions???

Case Studies

```
public static void salaryAdjustment(int profit, int
     investment){
    Statement s;
    ...
    int employee_count = 0;
    if(profit >= 0.5 * investment){
        String query1 = "SELECT employee_id,
             work_experience FROM WorkInfo WHERE
             work_experience > 10";
        resultSet1 = s.executeQuery(query1);
        resultSet1.last();
        if(resultSet1.getRow() > 100){
        String query3 = "SELECT employee_id FROM
             WorkInfo WHERE work_experience > 10 AND
             performance = 'good'";
        resultSet3 = s.executeQuery(query3);
        ... // do other operations
        } else{
            String query2 = "UPDATE WorkInfo SET salary
                 = salary * 1.2";
            s.executeUpdate(query2);
    }else{
        String query4 = "SELECT p.employee_name FROM
             PersonalInfo p, WorkInfo w WHERE
             performance = 'poor' AND p.employee_id =
             w.employee_id";
        resultSet2 = s.executeQuery(query4);
        ... // do other operations
    }
}
```

Assume, profit and investment variables are set to 60000 and 100000.

Issues query$_1$: c$_1$ is satisfied and the signature is matched with that of the QR$_1$. query1 is

Assume that number of rows returned by

Issues query$_3$: c$_3$ is not satisfied and the signature is matched. Considered as anomalous query.

Issues query$_2$: c$_2$ is satisfied and the signature is matched. Considered as normal query.