Scalable Compositional Static Taint Analysis for Industrial Microservices

Zexin Zhong, Jiangchao Liu, Diyu Wu, Peng Di, Yulei Sui and Alex X. Liu

In 44nd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP’22)

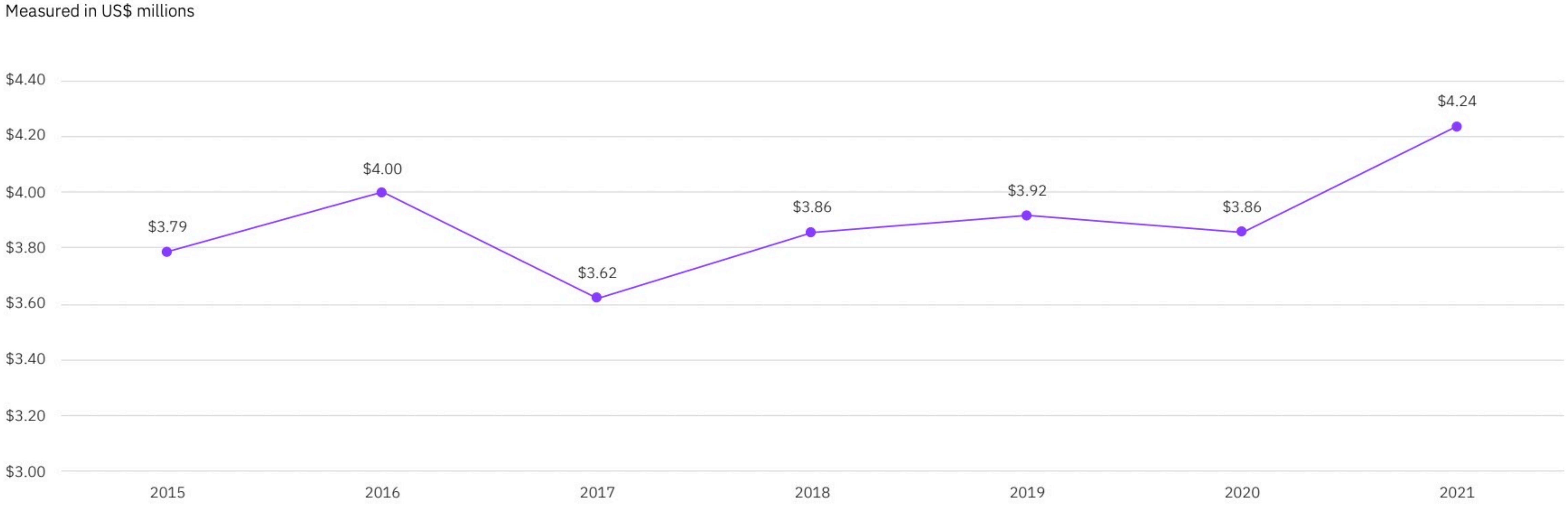
Outline

- Background & Motivation
- Challenges
- Approach
- Evaluation

UTS

Background

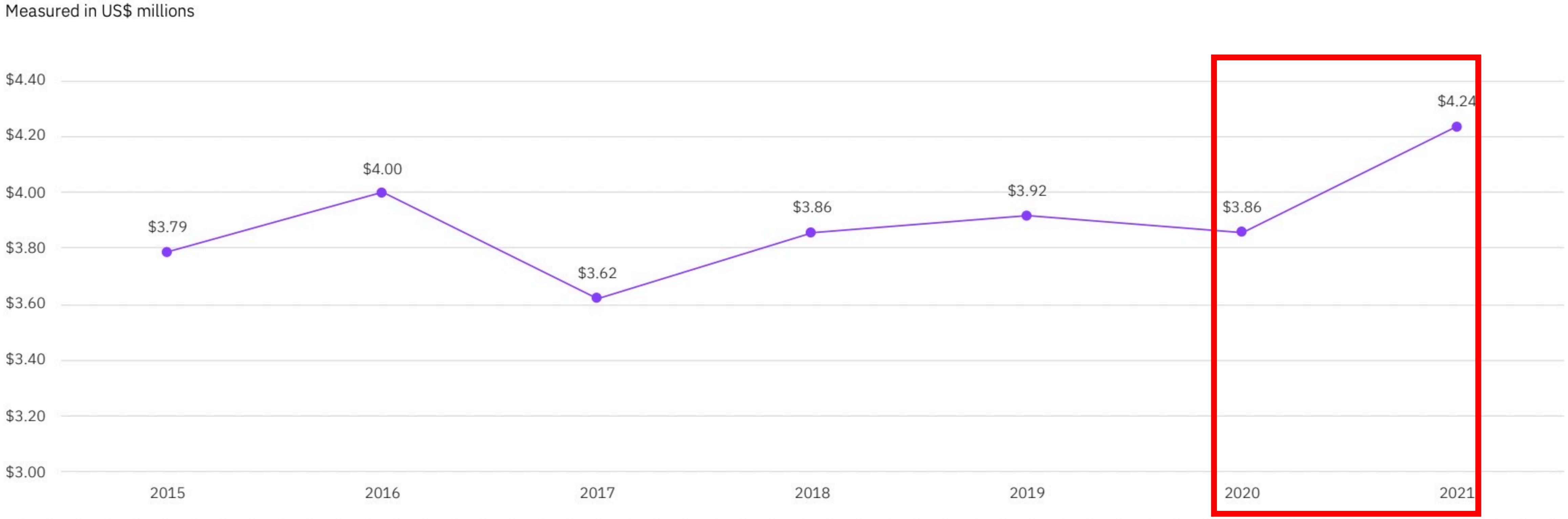
Increasing Cost of Data Breach in Industry



Average total cost of a data breach

(IBM Security, 2021)

Increasing Cost of Data Breach in Industry



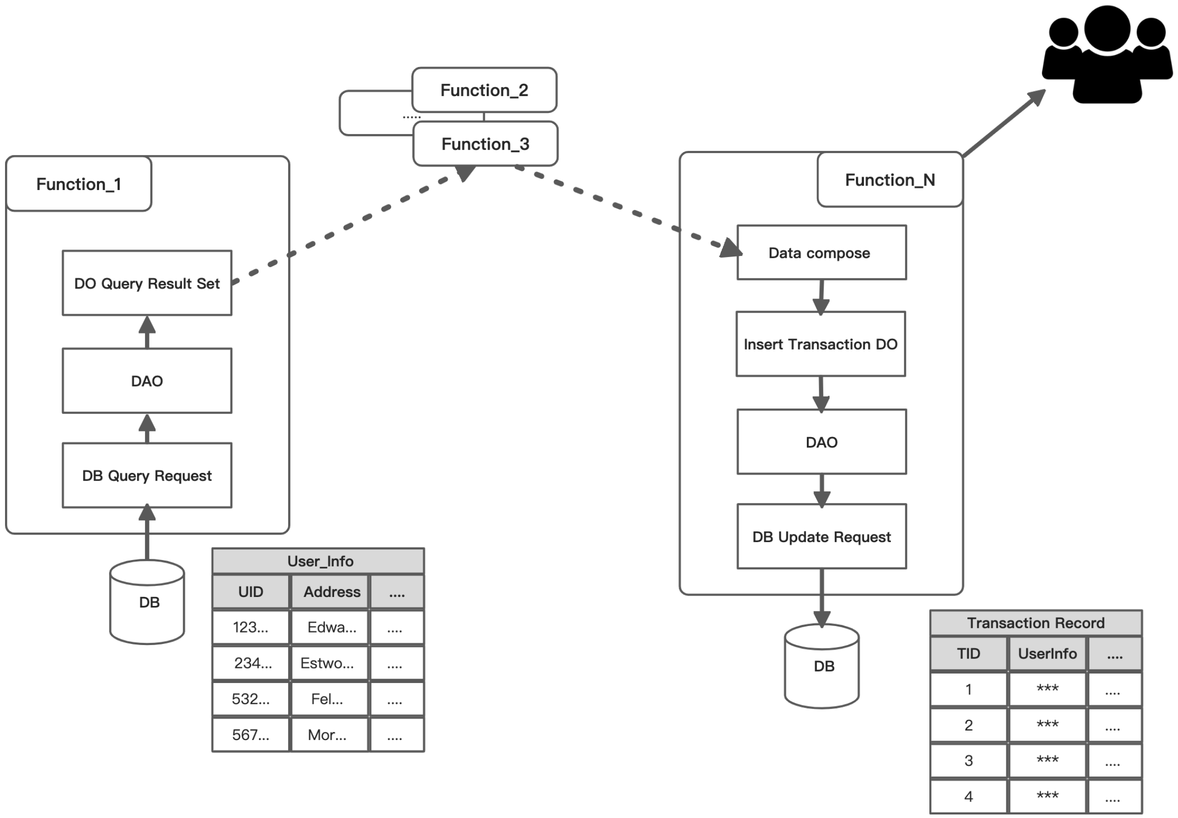
Average total cost of a data breach

(IBM Security, 2021)

- **10% Increase**
- **Largest margin in seven years**

Increasing Demand for Tracking Sensitive Data

- Security: data leak detection



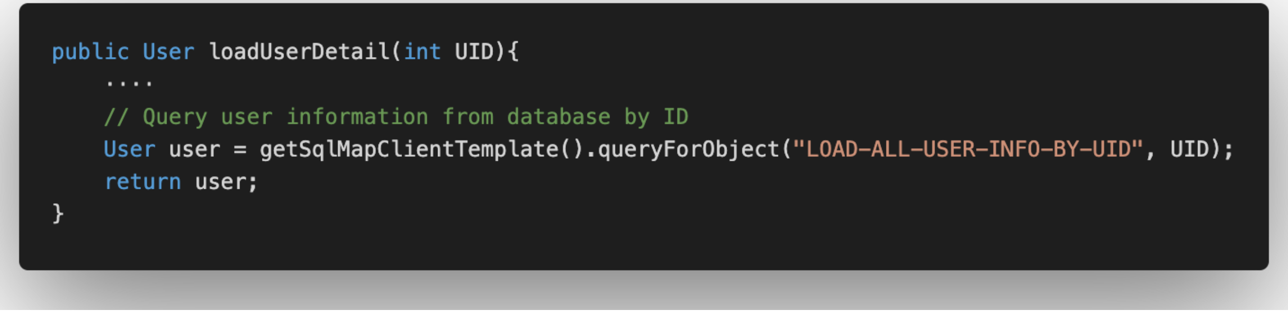
User_Info

UID	Address	....
123...	Edwa...	....
234...	Estwo...	....
532...	Fel...	....
567...	Mor...	....

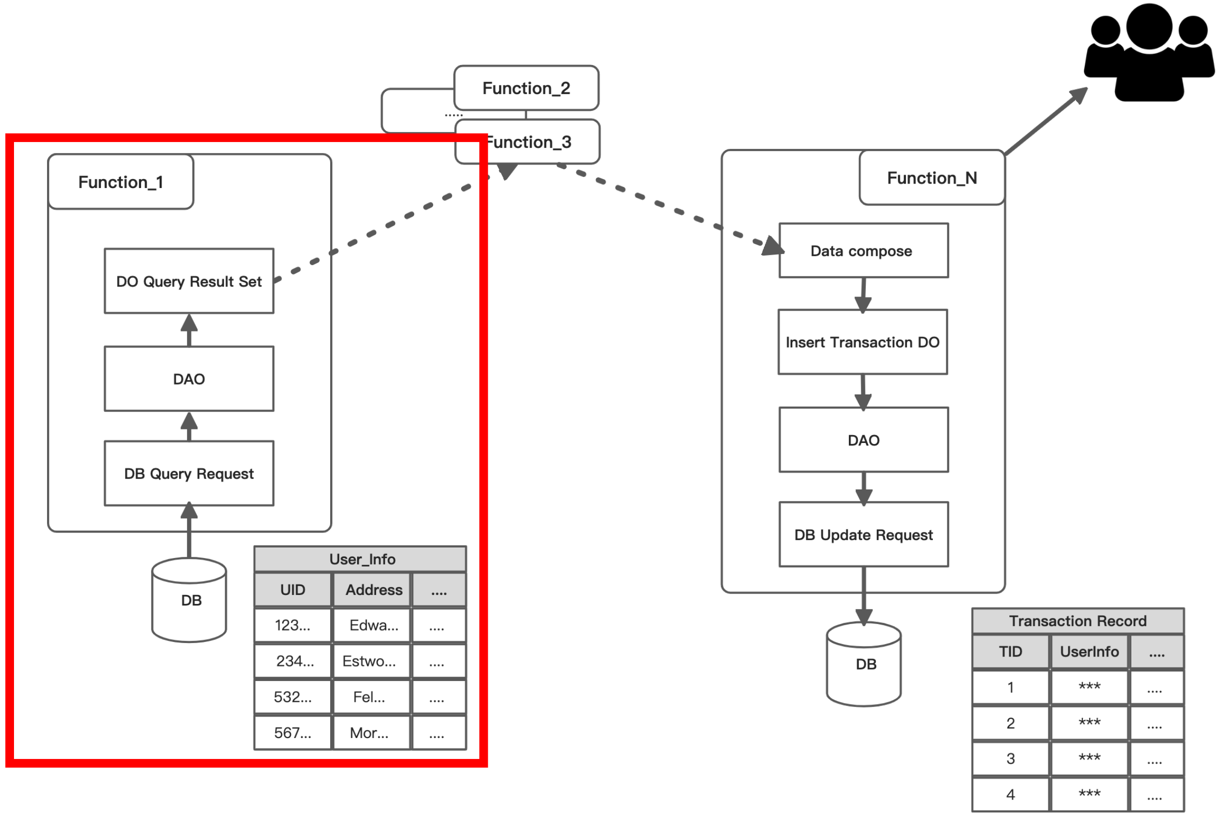
Transaction Record

TID	UserInfo	....
1	***	....
2	***	....
3	***	....
4	***	....

Increasing Demand for Tracking Sensitive Data



```
public User loadUserDetail(int UID){
    ....
    // Query user information from database by ID
    User user = getSqlMapClientTemplate().queryForObject("LOAD-ALL-USER-INFO-BY-UID", UID);
    return user;
}
```



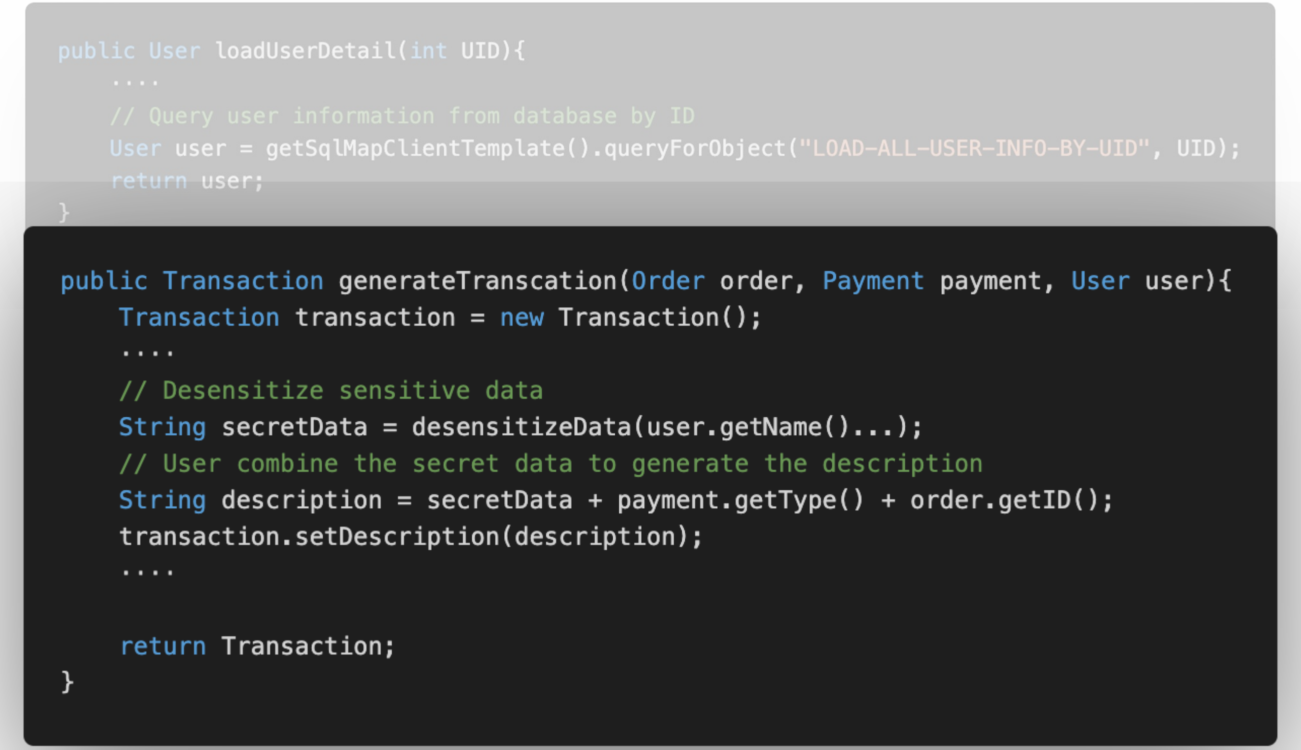
User_Info

UID	Address	....
123...	Edwa...	....
234...	Estwo...	....
532...	Fel...	....
567...	Mor...	....

Transaction Record

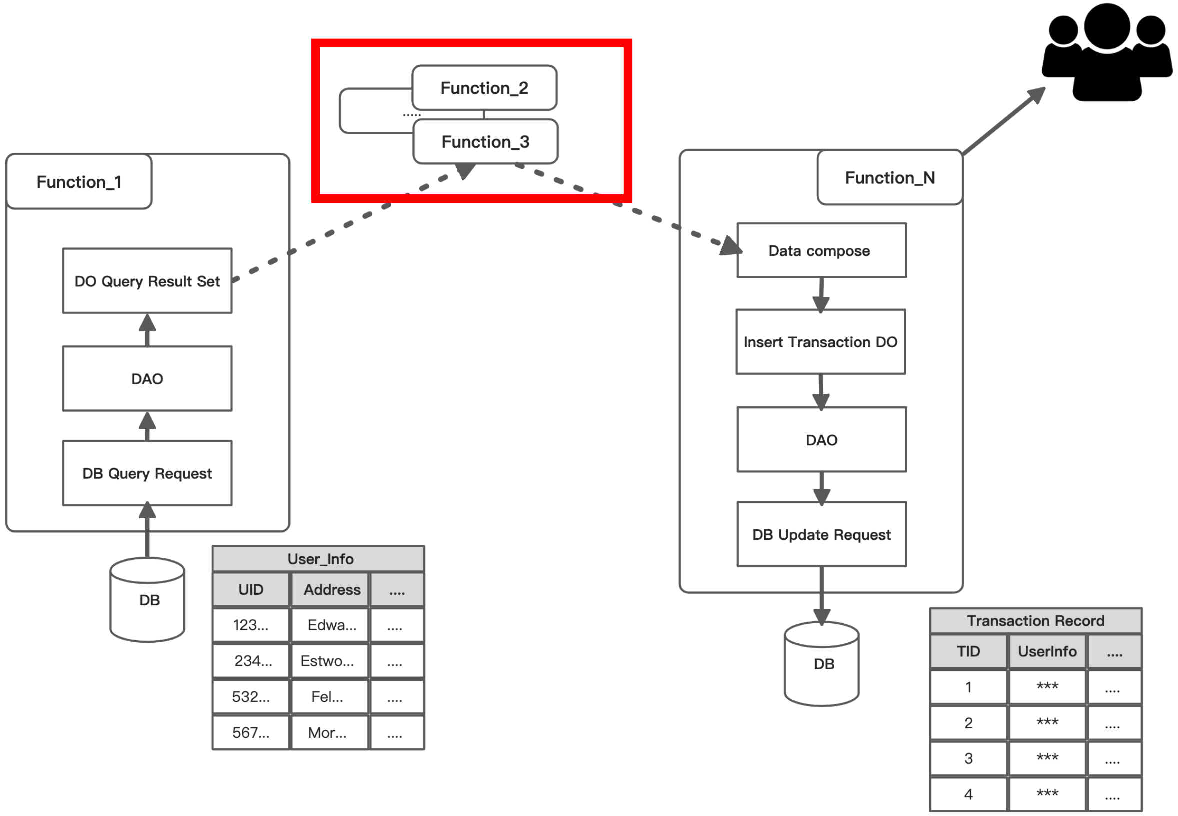
TID	UserInfo	....
1	***	....
2	***	....
3	***	....
4	***	....

Increasing Demand for Tracking Sensitive Data



```
public User loadUserDetail(int UID){
    ....
    // Query user information from database by ID
    User user = getSqlMapClientTemplate().queryForObject("LOAD-ALL-USER-INFO-BY-UID", UID);
    return user;
}
```

```
public Transaction generateTranscation(Order order, Payment payment, User user){
    Transaction transaction = new Transaction();
    ....
    // Desensitize sensitive data
    String secretData = desensitizeData(user.getName()...);
    // User combine the secret data to generate the description
    String description = secretData + payment.getType() + order.getID();
    transaction.setDescription(description);
    ....

    return Transaction;
}
```



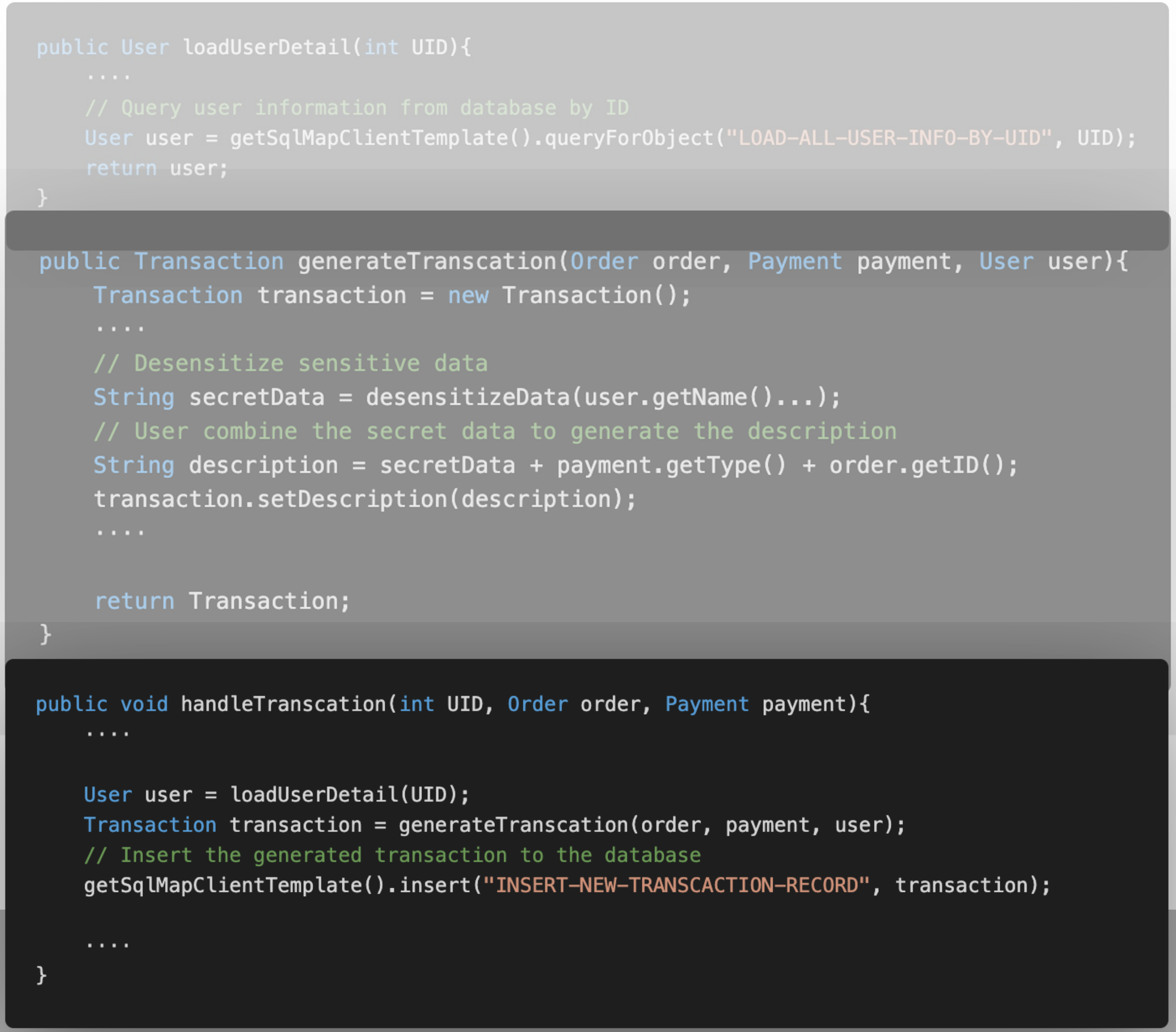
User_Info

UID	Address	....
123...	Edwa...	....
234...	Estwo...	....
532...	Fel...	....
567...	Mor...	....

Transaction Record

TID	UserInfo	....
1	***	....
2	***	....
3	***	....
4	***	....

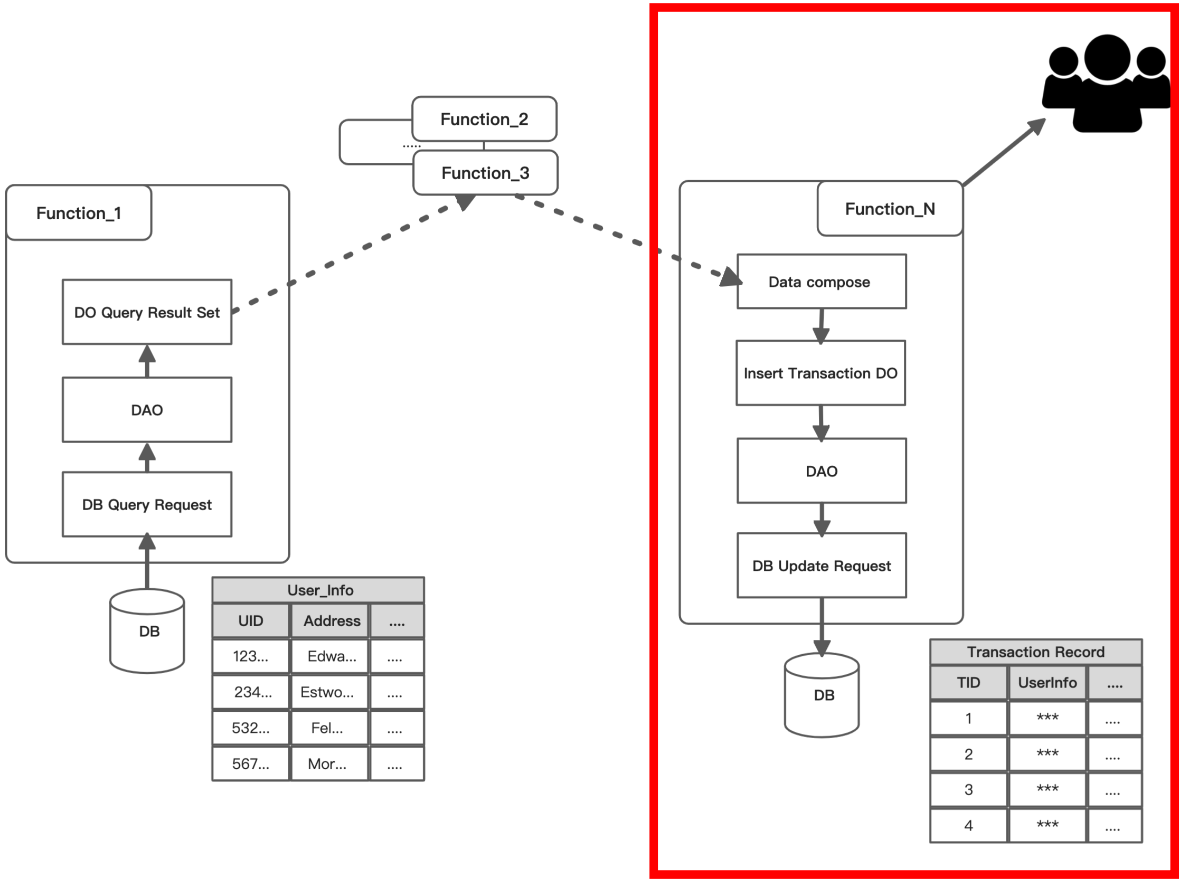
Increasing Demand for Tracking Sensitive Data



```
public User loadUserDetail(int UID){
    ....
    // Query user information from database by ID
    User user = getSqlMapClientTemplate().queryForObject("LOAD-ALL-USER-INFO-BY-UID", UID);
    return user;
}
```

```
public Transaction generateTranscation(Order order, Payment payment, User user){
    Transaction transaction = new Transaction();
    ....
    // Desensitize sensitive data
    String secretData = desensitizeData(user.getName()...);
    // User combine the secret data to generate the description
    String description = secretData + payment.getType() + order.getID();
    transaction.setDescription(description);
    ....

    return Transaction;
}
```

```
public void handleTranscation(int UID, Order order, Payment payment){
    ....

    User user = loadUserDetail(UID);
    Transaction transaction = generateTranscation(order, payment, user);
    // Insert the generated transaction to the database
    getSqlMapClientTemplate().insert("INSERT-NEW-TRANSCACTION-RECORD", transaction);

    ....
}
```



User_Info

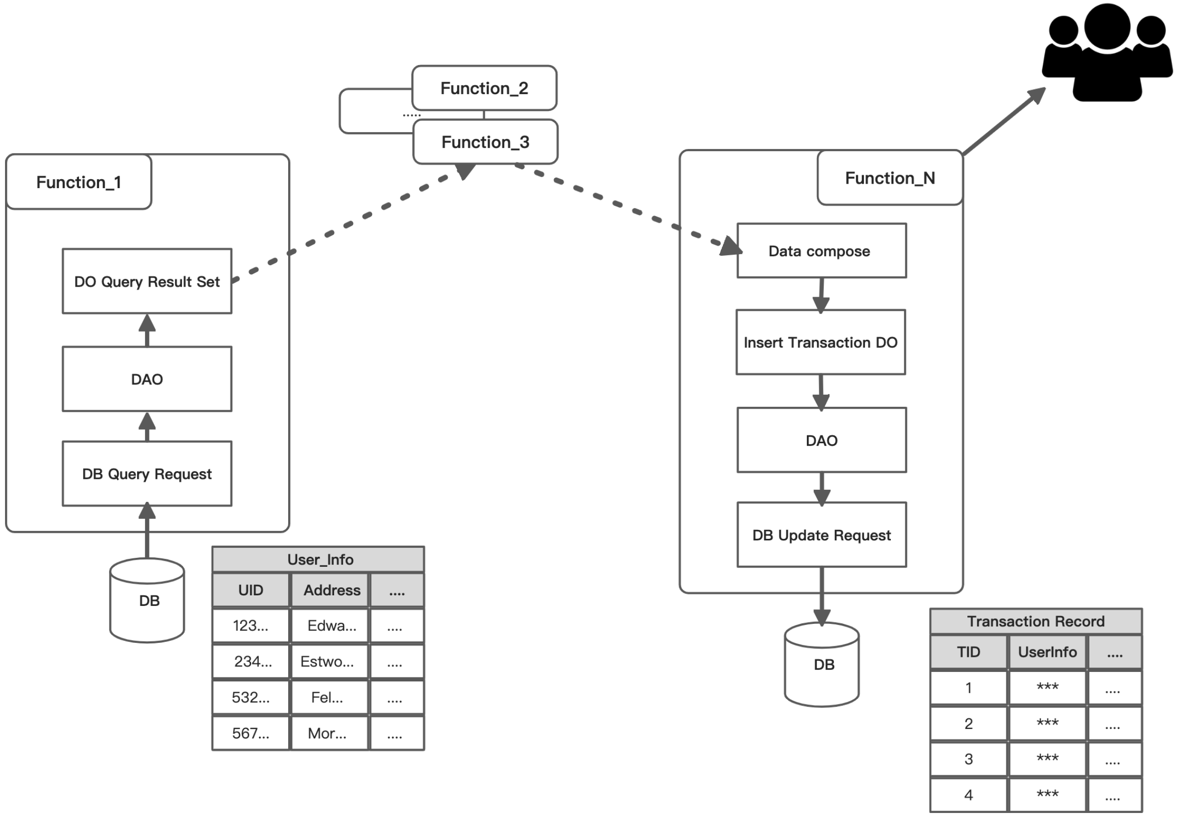
UID	Address	....
123...	Edwa...	....
234...	Estwo...	....
532...	Fel...	....
567...	Mor...	....

Transaction Record

TID	UserInfo	....
1	***	....
2	***	....
3	***	....
4	***	....

Increasing Demand for Tracking Sensitive Data

1. Where will the user information be loaded in the microservices applications?



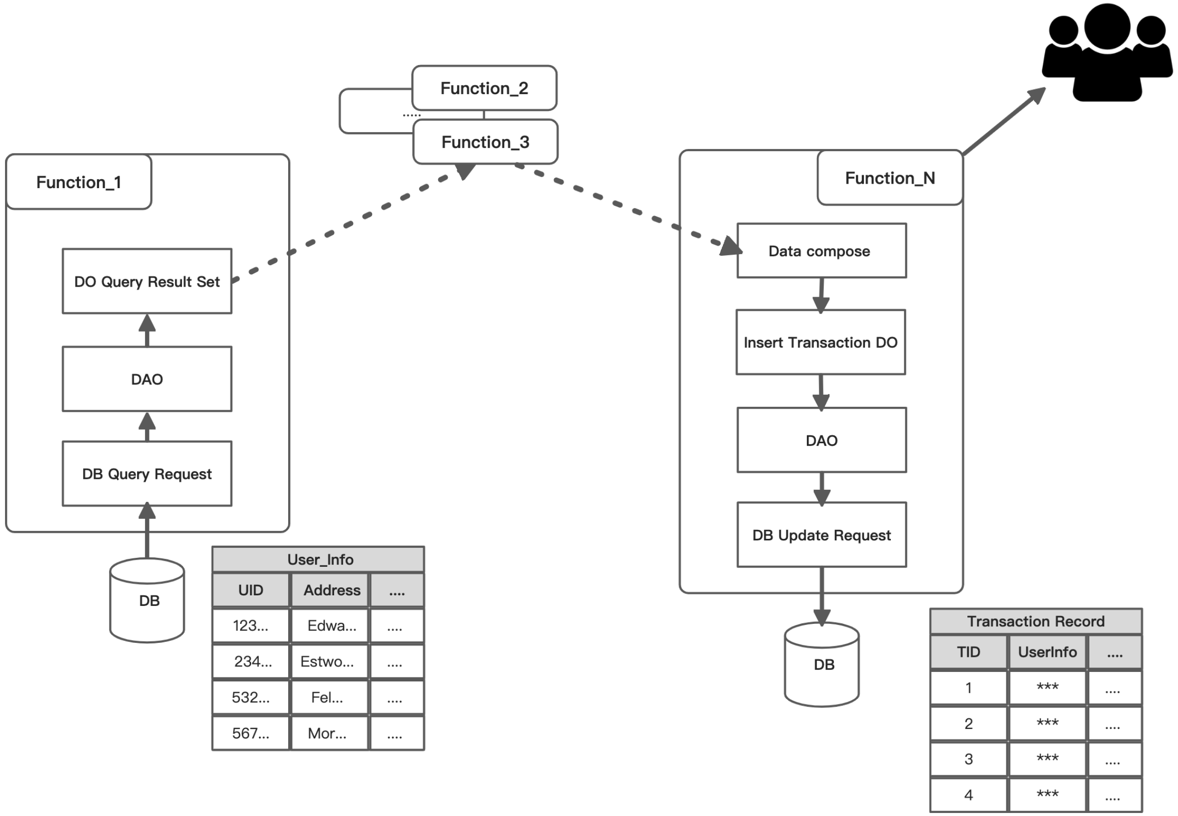
User_Info		
UID	Address	....
123...	Edwa...	....
234...	Estwo...	....
532...	Fel...	....
567...	Mor...	....

Transaction Record		
TID	UserInfo	....
1	***	....
2	***	....
3	***	....
4	***	....

Increasing Demand for Tracking Sensitive Data

1. Where will the user information be loaded in the microservices applications?

2. How will the user information be used in the microservices applications ?



User_Info		
UID	Address	....
123...	Edwa...	....
234...	Estwo...	....
532...	Fel...	....
567...	Mor...	....

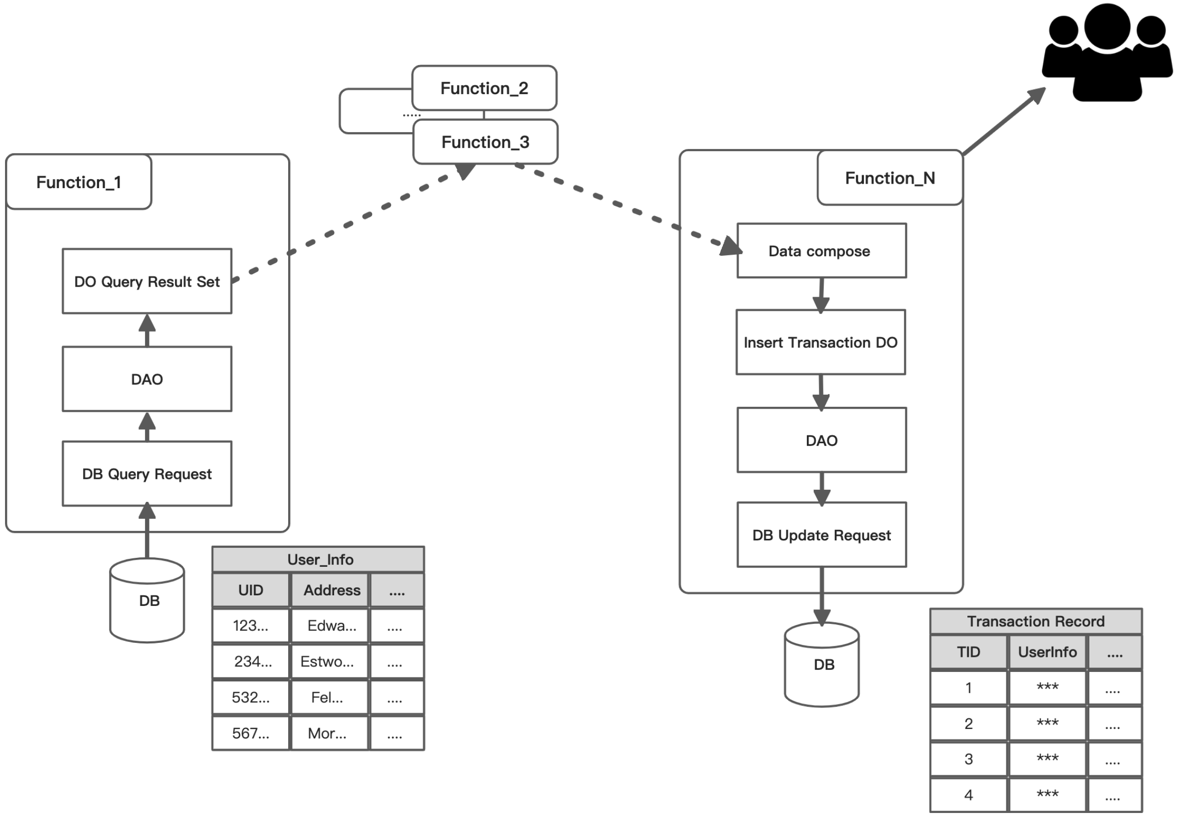
Transaction Record		
TID	UserInfo	....
1	***	....
2	***	....
3	***	....
4	***	....

Increasing Demand for Tracking Sensitive Data

1. Where will the user information be loaded in the microservices applications?

2. How will the user information be used in the microservices applications ?

3. How to identify the propagation paths of the user information in the microservices applications?



User_Info

UID	Address	....
123...	Edwa...	....
234...	Estwo...	....
532...	Fel...	....
567...	Mor...	....

Transaction Record

TID	UserInfo	....
1	***	....
2	***	....
3	***	....
4	***	....

Taint Analysis

- What is taint analysis?
  - Information/data flow tracking analysis
  - Aims to reason about the control and data dependence from a source to sink

Taint Analysis

- What is taint analysis?
  - Information/data flow tracking analysis
  - Aims to reason about the control and data dependence from a source to sink

- What can taint analysis do?
  - Detect sensitive data leak

Taint Analysis

- What is taint analysis?
  - Information/data flow tracking analysis
  - Aims to reason about the control and data dependence from a source to sink
- What can taint analysis do?
  - Detect sensitive data leak
  - Detect code vulnerability

Taint Analysis

- What is taint analysis?
  - Information/data flow tracking analysis
  - Aims to reason about the control and data dependence from a source to sink
- What can taint analysis do?
  - Detect sensitive data leak
  - Detect code vulnerability
  - Use for change governance

Taint Analysis

◆What is taint analysis?

- ◆ Information/data flow tracking analysis
- ◆ Aims to reason about the control and data dependence from a source to sink

◆What can taint analysis do?

- ◆Detect sensitive data leak
- ◆Detect code vulnerability
- ◆Use for change governance

- FlowDroid
- ANTaint
- F4F
- DroidInfer

Challenges

Recall

Scalability

Precision

Challenges

Recall

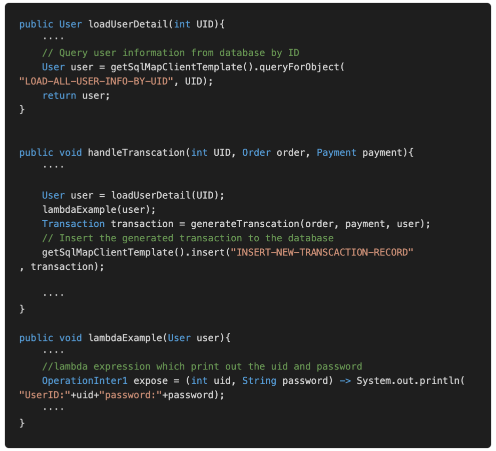
Challenges

Recall

Challenge 1: Low Recall Rate

Unsound call-graph due to framework behaviours

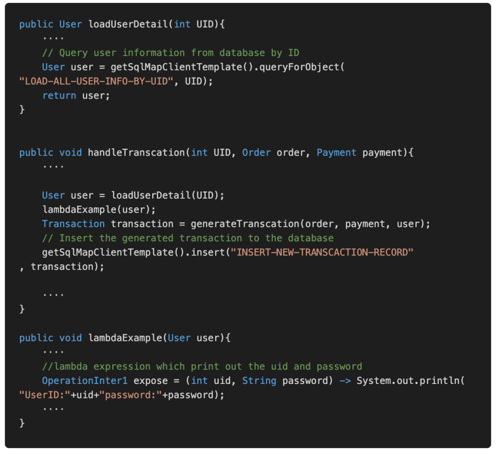
Challenge1: Recall



```java
public User loadUserDetail(int UID){
    ....
    // Query user information from database by ID
    User user = getSqlMapClientTemplate().queryForObject(
"LOAD-ALL-USER-INFO-BY-UID", UID);
    return user;
}


public void handleTranscation(int UID, Order order, Payment payment){
    ....

    User user = loadUserDetail(UID);
    lambdaExample(user);
    Transaction transaction = generateTranscation(order, payment, user);
    // Insert the generated transaction to the database
    getSqlMapClientTemplate().insert("INSERT-NEW-TRANSCACTION-RECORD"
, transaction);

    ....
}

public void lambdaExample(User user){
    ....
    //lambda expression which print out the uid and password
    OperationInter1 expose = (int uid, String password) -> System.out.println(
"UserID:"+uid+"password:"+password);
    ....
}
```

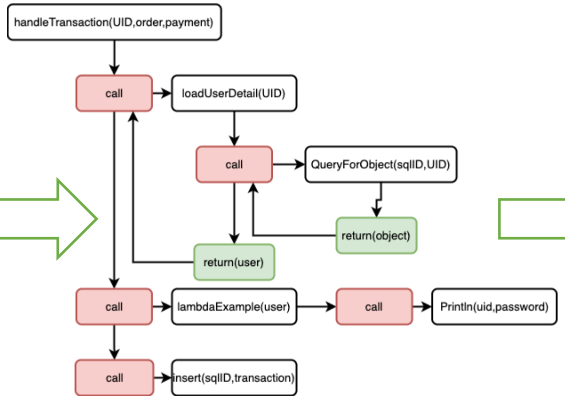
Code Fragment

Challenge1: Recall



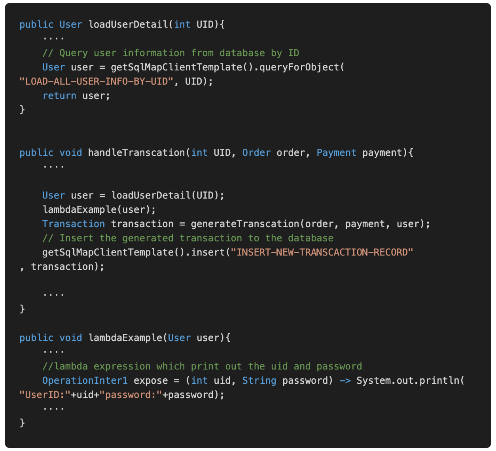
```
public User loadUserDetail(int UID){
    ....
    // Query user information from database by ID
    User user = getSqlMapClientTemplate().queryForObject(
"LOAD-ALL-USER-INFO-BY-UID", UID);
    return user;
}


public void handleTranscation(int UID, Order order, Payment payment){
    ....

    User user = loadUserDetail(UID);
    lambdaExample(user);
    Transaction transaction = generateTranscation(order, payment, user);
    // Insert the generated transaction to the database
    getSqlMapClientTemplate().insert("INSERT-NEW-TRANSCACTION-RECORD"
, transaction);

    ....
}

public void lambdaExample(User user){
    ....
    //lambda expression which print out the uid and password
    OperationInter1 expose = (int uid, String password) -> System.out.println(
"UserID:"+uid+"password:"+password);
    ....
}
```

Code Fragment



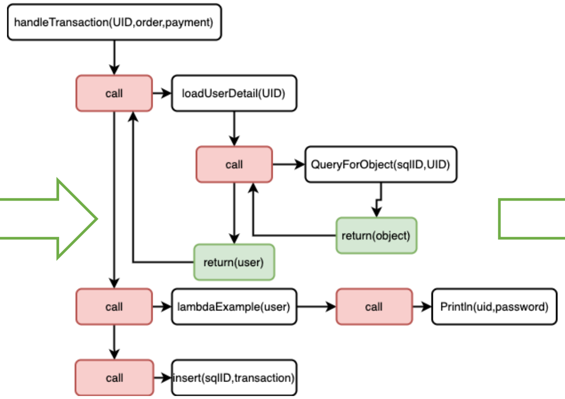
Call Graph

Challenge1: Recall

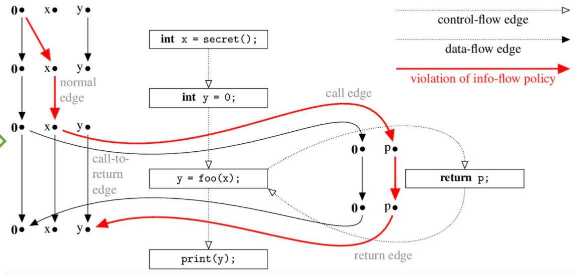


```
public User loadUserDetail(int UID){
    ....
    // Query user information from database by ID
    User user = getSqlMapClientTemplate().queryForObject(
"LOAD-ALL-USER-INFO-BY-UID", UID);
    return user;
}


public void handleTranscation(int UID, Order order, Payment payment){
    ....

    User user = loadUserDetail(UID);
    lambdaExample(user);
    Transaction transaction = generateTranscation(order, payment, user);
    // Insert the generated transaction to the database
    getSqlMapClientTemplate().insert("INSERT-NEW-TRANSCACTION-RECORD"
, transaction);

    ....
}

public void lambdaExample(User user){
    ....
    //lambda expression which print out the uid and password
    OperationInter1 expose = (int uid, String password) -> System.out.println(
"UserID:"+uid+"password:"+password);
    ....
}
```

Code Fragment

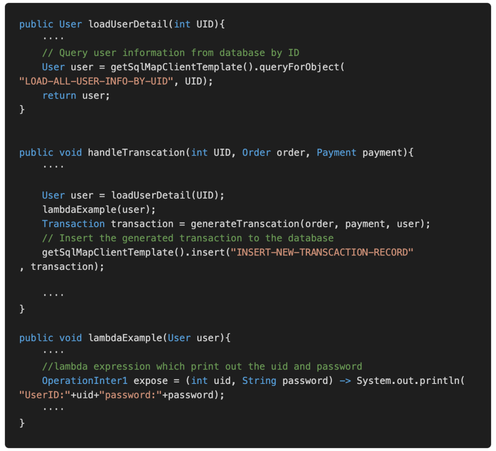


Call Graph



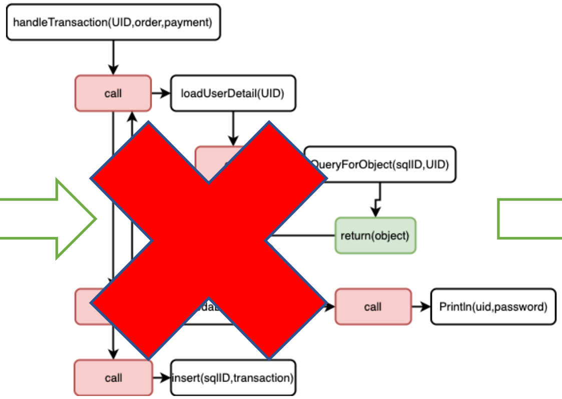
Data Flow Graph

Challenge1: Recall

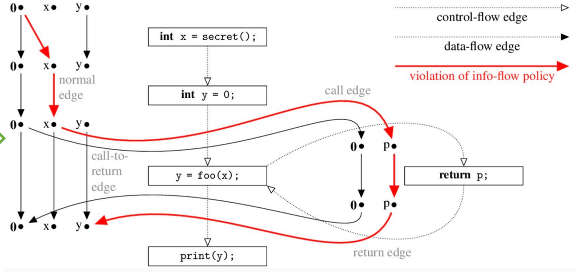


```
public User loadUserDetail(int UID){
    ....
    // Query user information from database by ID
    User user = getSqlMapClientTemplate().queryForObject(
"LOAD-ALL-USER-INFO-BY-UID", UID);
    return user;
}


public void handleTranscation(int UID, Order order, Payment payment){
    ....

    User user = loadUserDetail(UID);
    lambdaExample(user);
    Transaction transaction = generateTranscation(order, payment, user);
    // Insert the generated transaction to the database
    getSqlMapClientTemplate().insert("INSERT-NEW-TRANSCACTION-RECORD"
, transaction);

    ....
}

public void lambdaExample(User user){
    ....
    //lambda expression which print out the uid and password
    OperationInter1 expose = (int uid, String password) -> System.out.println(
"UserID:"+uid+"password:"+password);
    ....
}
```

Code Fragment


handleTransaction(UID,order,payment)
call
loadUserDetail(UID)
QueryForObject(sqlID,UID)
return(object)
call
Println(uid,password)
call
insert(sqlID,transaction)

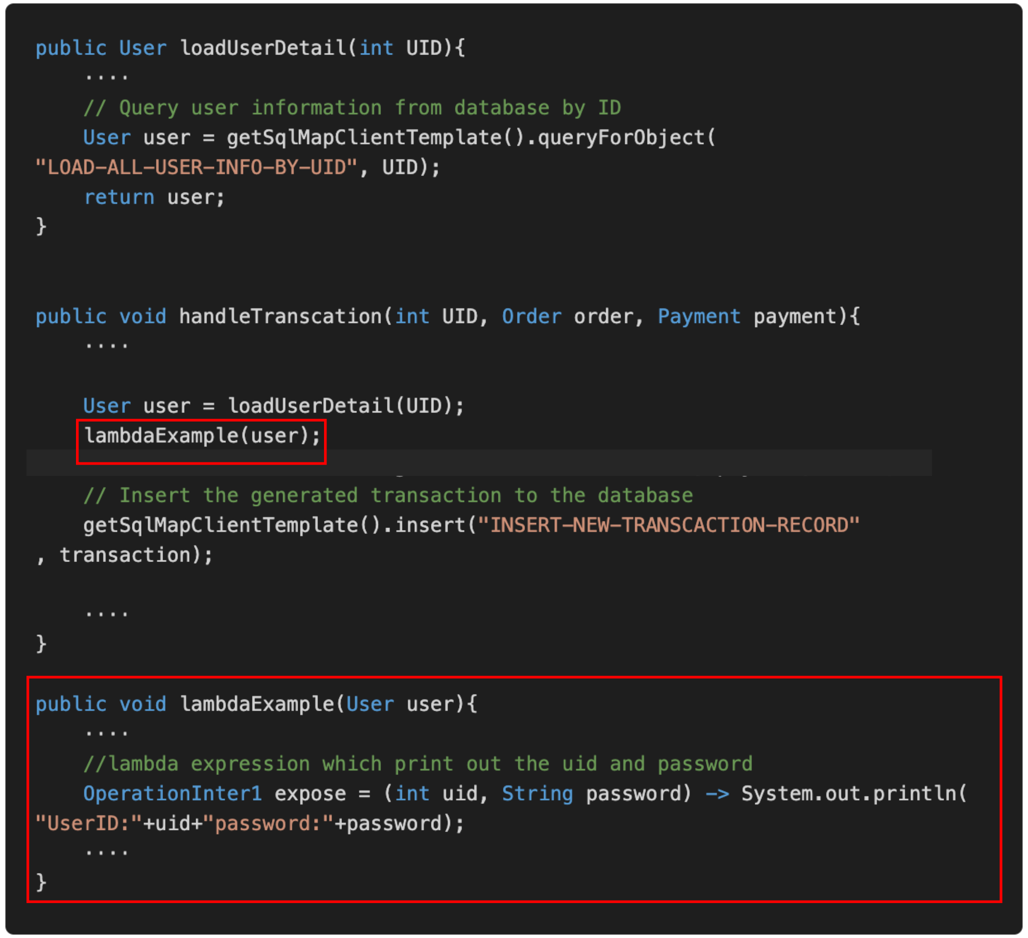
Call Graph



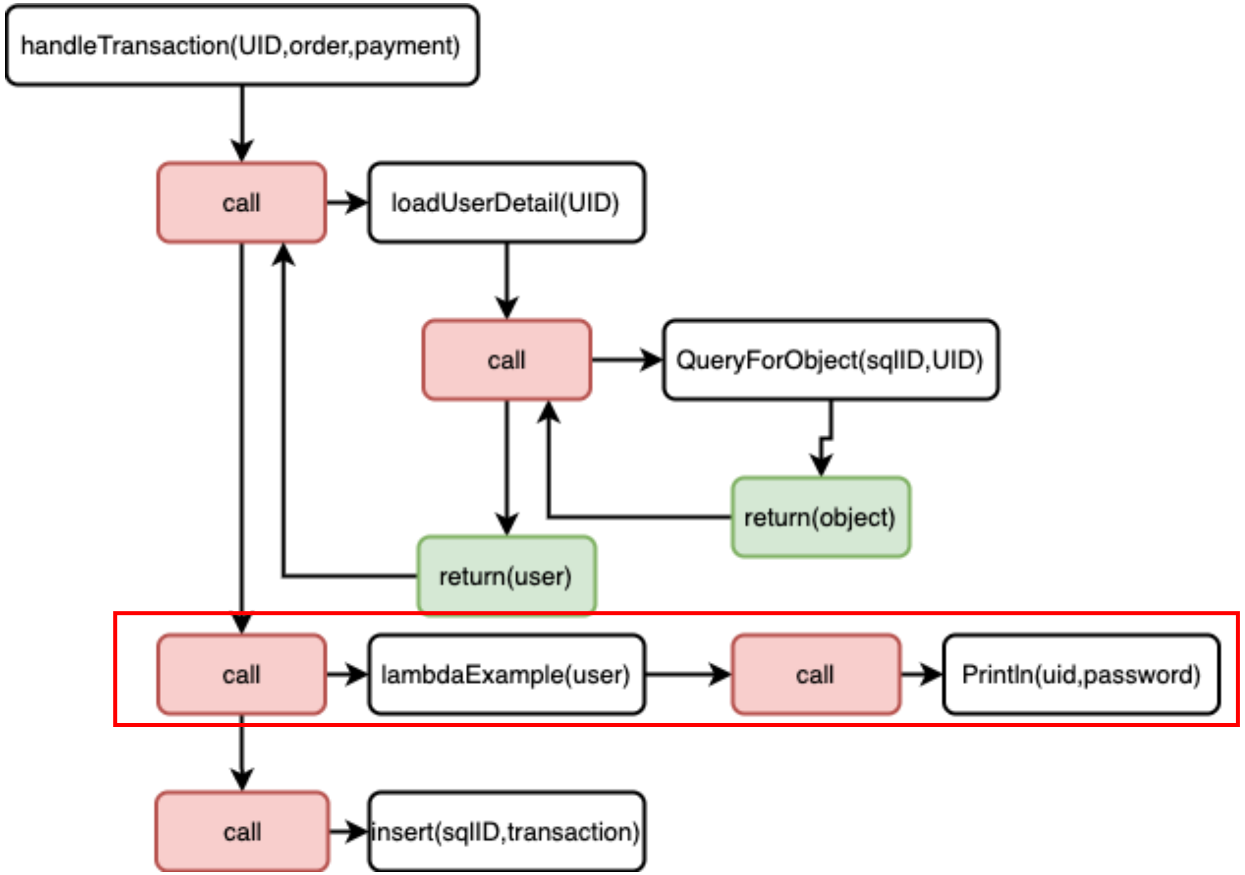
Control Flow Graph

Unsound call graph due to complex framework behaviors

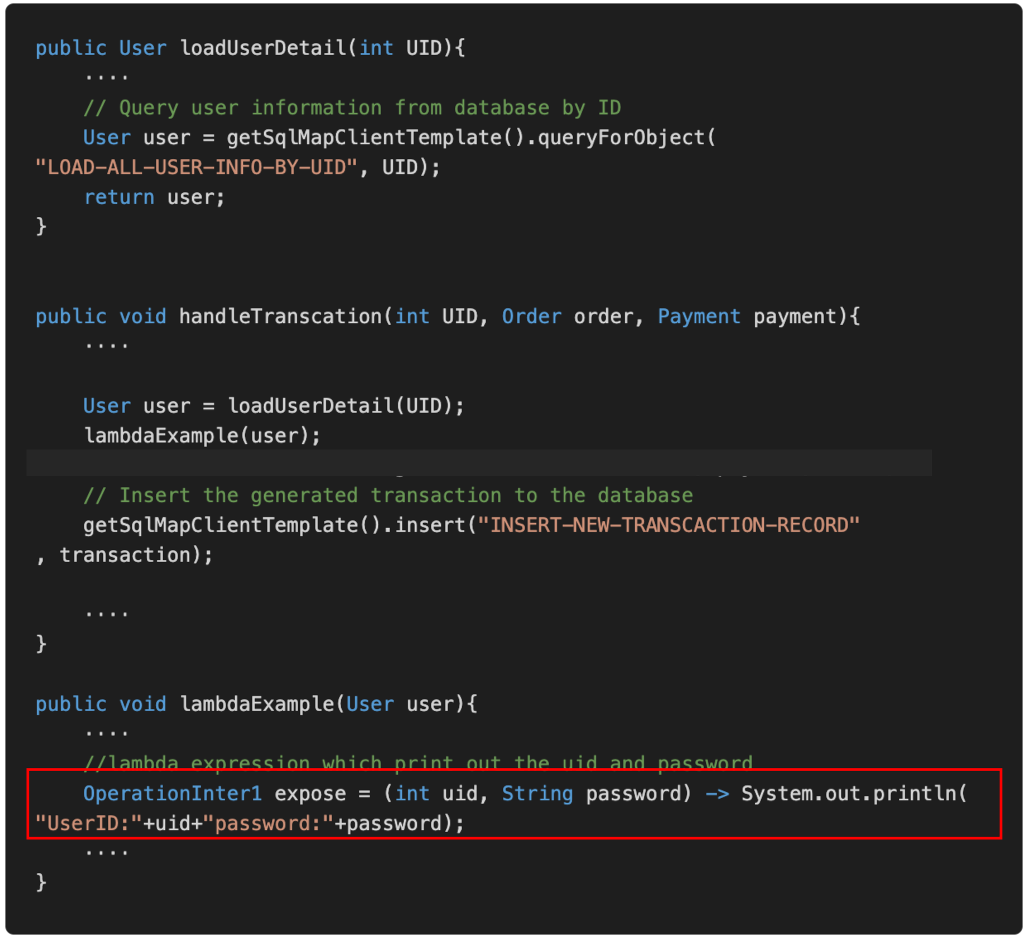
Challenge1: Recall



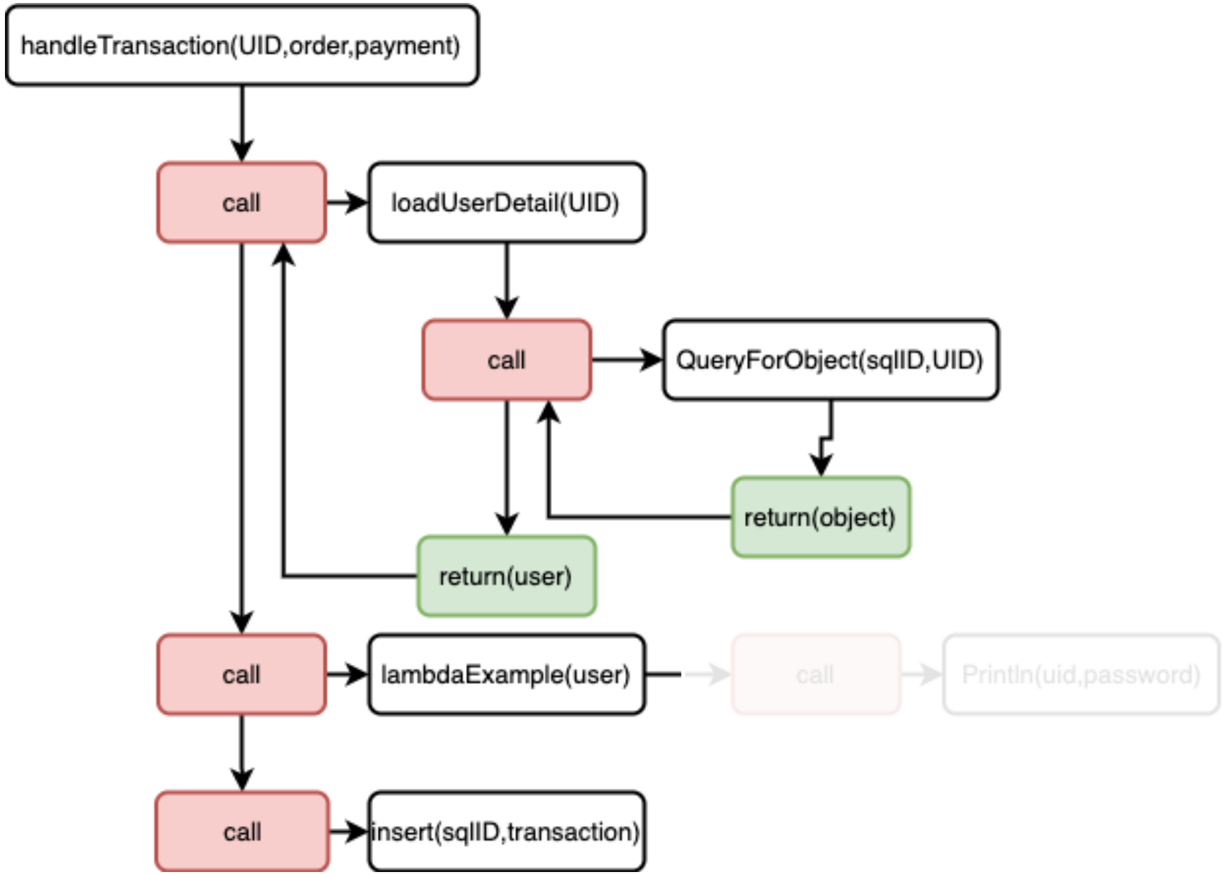
```
public User loadUserDetail(int UID){
    ....
    // Query user information from database by ID
    User user = getSqlMapClientTemplate().queryForObject(
"LOAD-ALL-USER-INFO-BY-UID", UID);
    return user;
}


public void handleTranscation(int UID, Order order, Payment payment){
    ....

    User user = loadUserDetail(UID);
    lambdaExample(user);

    // Insert the generated transaction to the database
    getSqlMapClientTemplate().insert("INSERT-NEW-TRANSCACTION-RECORD"
, transaction);

    ....
}

public void lambdaExample(User user){
    ....
    //lambda expression which print out the uid and password
    OperationInter1 expose = (int uid, String password) -> System.out.println(
"UserID:"+uid+"password:"+password);
    ....
}
```



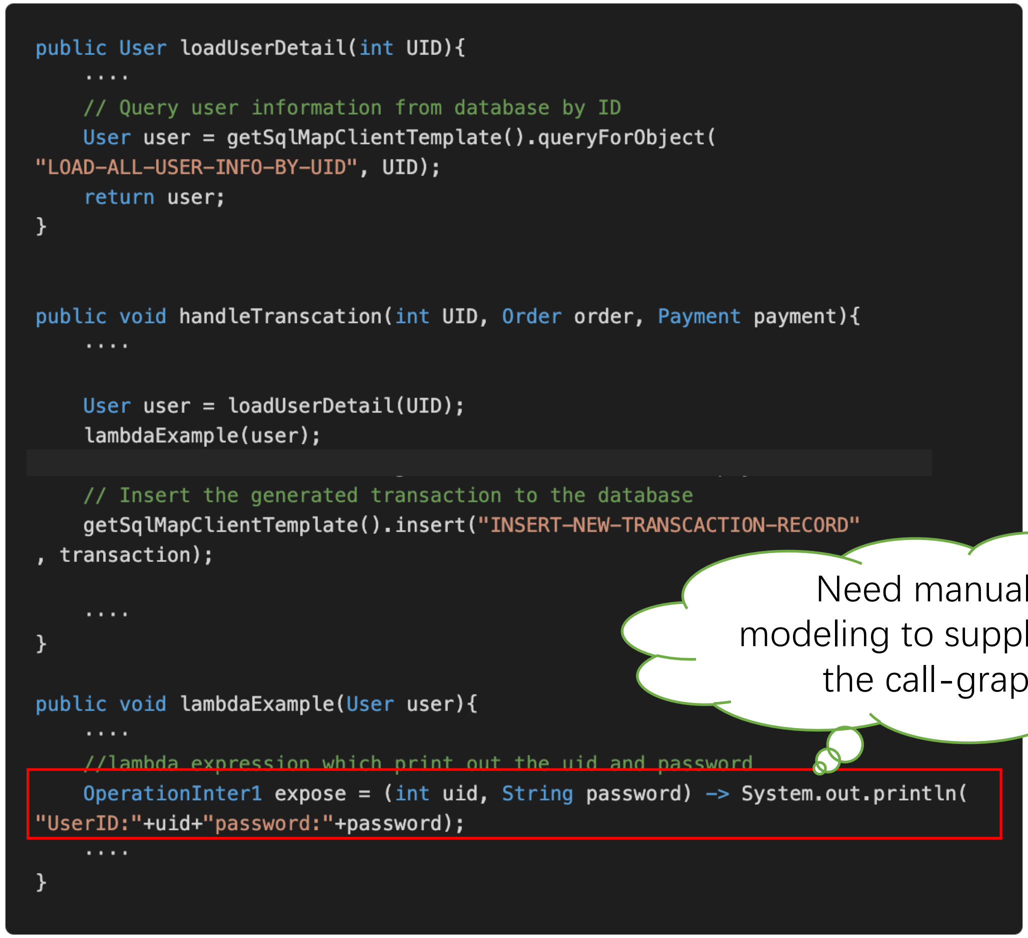
Challenge1: Recall



```
public User loadUserDetail(int UID){
    ....
    // Query user information from database by ID
    User user = getSqlMapClientTemplate().queryForObject(
"LOAD-ALL-USER-INFO-BY-UID", UID);
    return user;
}


public void handleTranscation(int UID, Order order, Payment payment){
    ....

    User user = loadUserDetail(UID);
    lambdaExample(user);

    // Insert the generated transaction to the database
    getSqlMapClientTemplate().insert("INSERT-NEW-TRANSCACTION-RECORD"
, transaction);

    ....
}

public void lambdaExample(User user){
    ....
    //lambda expression which print out the uid and password
    OperationInter1 expose = (int uid, String password) -> System.out.println(
"UserID:"+uid+"password:"+password);
    ....
}
```

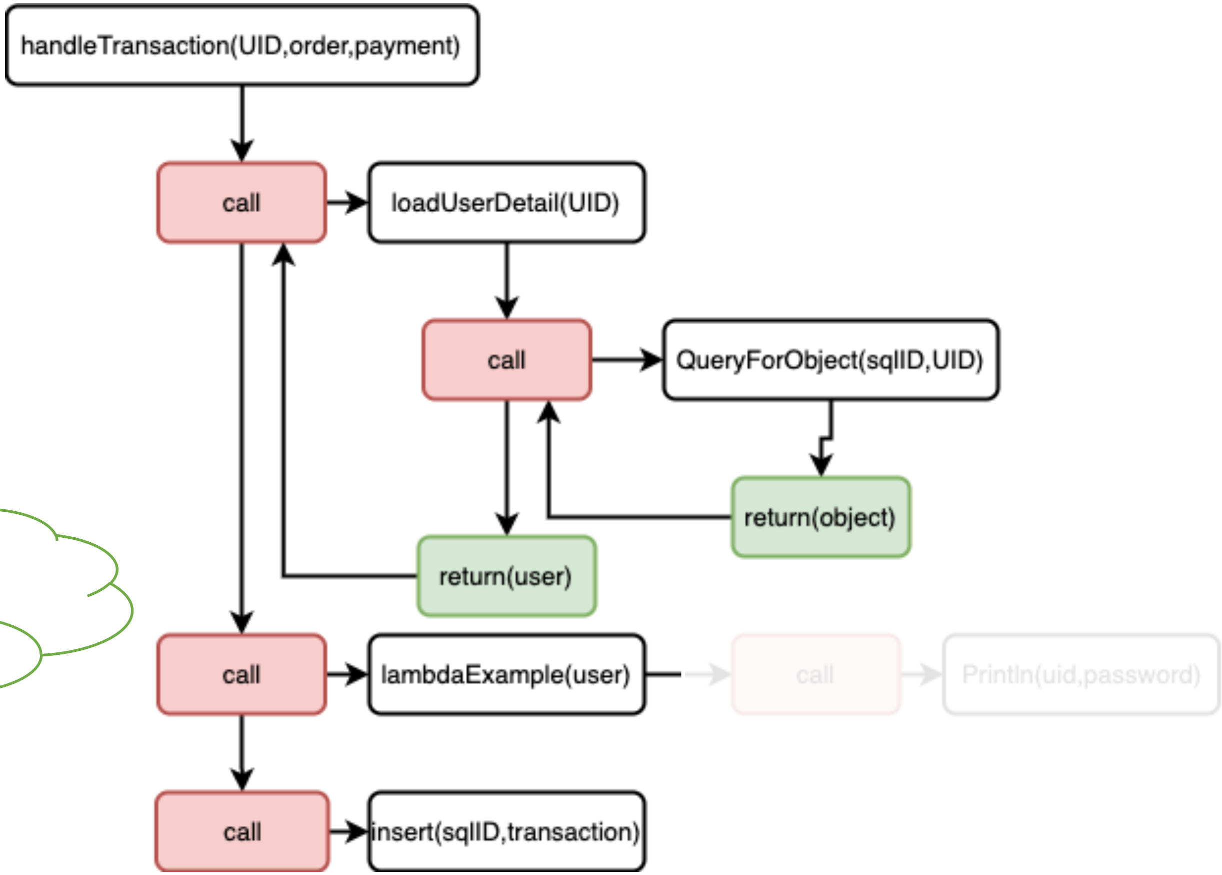


Challenge1: Recall

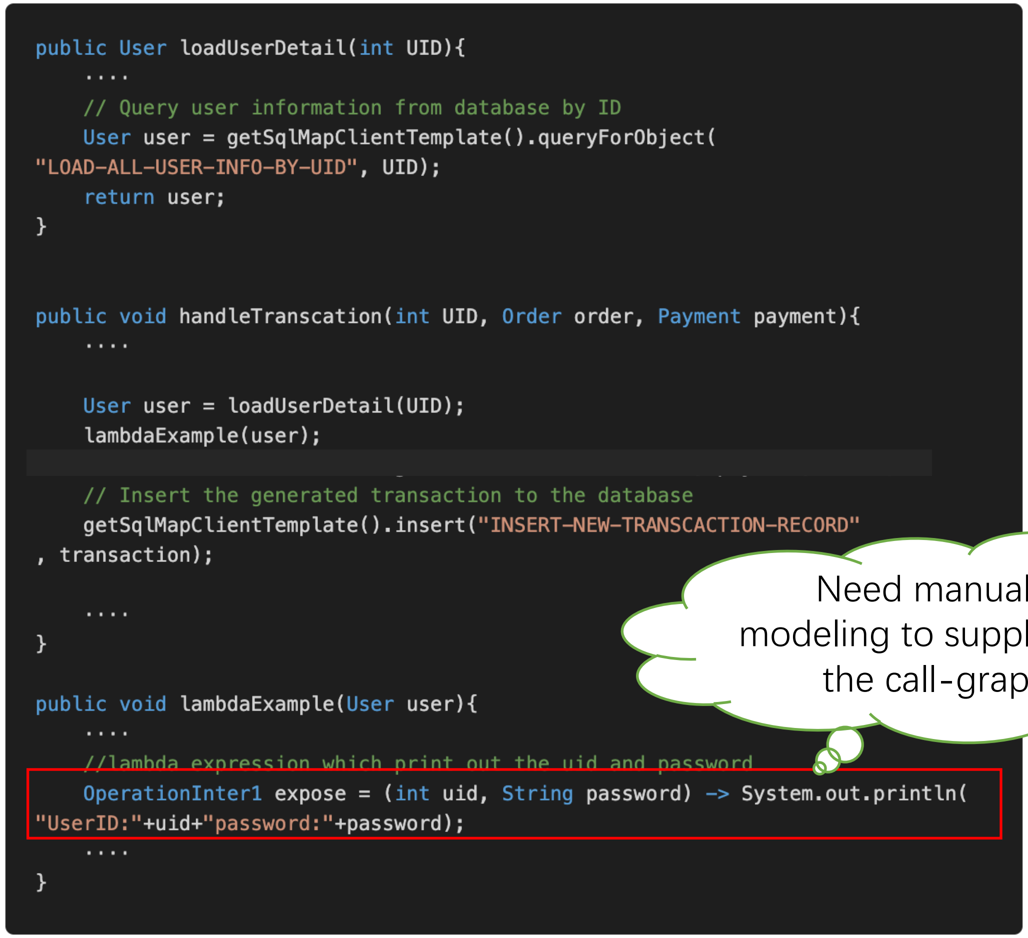


```
public User loadUserDetail(int UID){
    ....
    // Query user information from database by ID
    User user = getSqlMapClientTemplate().queryForObject(
"LOAD-ALL-USER-INFO-BY-UID", UID);
    return user;
}


public void handleTranscation(int UID, Order order, Payment payment){
    ....

    User user = loadUserDetail(UID);
    lambdaExample(user);


    // Insert the generated transaction to the database
    getSqlMapClientTemplate().insert("INSERT-NEW-TRANSCACTION-RECORD"
, transaction);

    ....
}

public void lambdaExample(User user){
    ....
    //lambda expression which print out the uid and password
    OperationInter1 expose = (int uid, String password) -> System.out.println(
"UserID:"+uid+"password:"+password);
    ....
}
```

Need manually modeling to supplement the call-graph

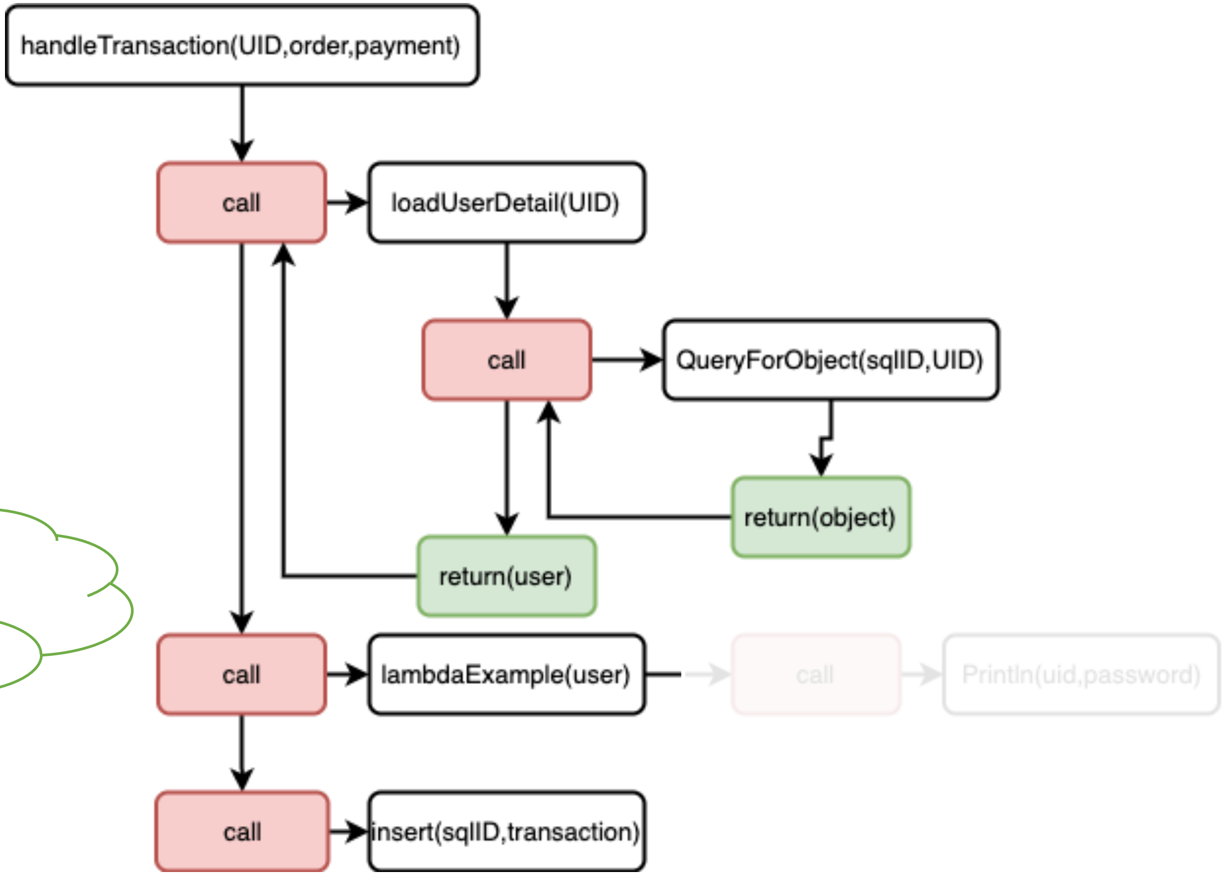


Challenge1: Recall



```
public User loadUserDetail(int UID){
    ....
    // Query user information from database by ID
    User user = getSqlMapClientTemplate().queryForObject(
"LOAD-ALL-USER-INFO-BY-UID", UID);
    return user;
}


public void handleTranscation(int UID, Order order, Payment payment){
    ....

    User user = loadUserDetail(UID);
    lambdaExample(user);

    // Insert the generated transaction to the database
    getSqlMapClientTemplate().insert("INSERT-NEW-TRANSCACTION-RECORD"
, transaction);

    ....
}

public void lambdaExample(User user){
    ....
    //lambda expression which print out the uid and password
    OperationInter1 expose = (int uid, String password) -> System.out.println(
"UserID:"+uid+"password:"+password);
    ....
}
```

Need manually modeling to supplement the call-graph



- Incomplete call-graph caused the lack of trace

Challenges

Recall

Challenge 1: Low Recall Rate

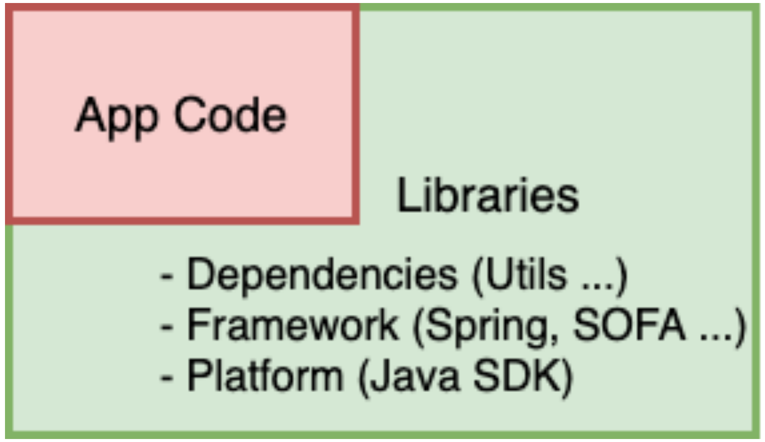
Unsound call-graph due to framework behaviours

Scalability

Challenge 2: Unscalable

Costly in both memory usage and time consumption

Challenge2: Scalability

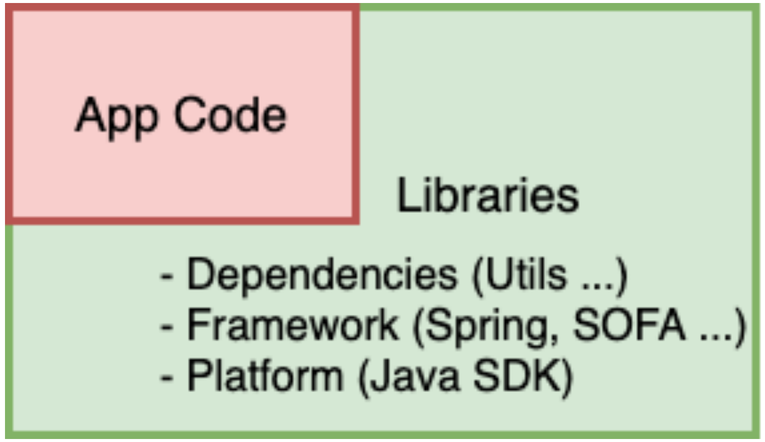


App Code

Libraries

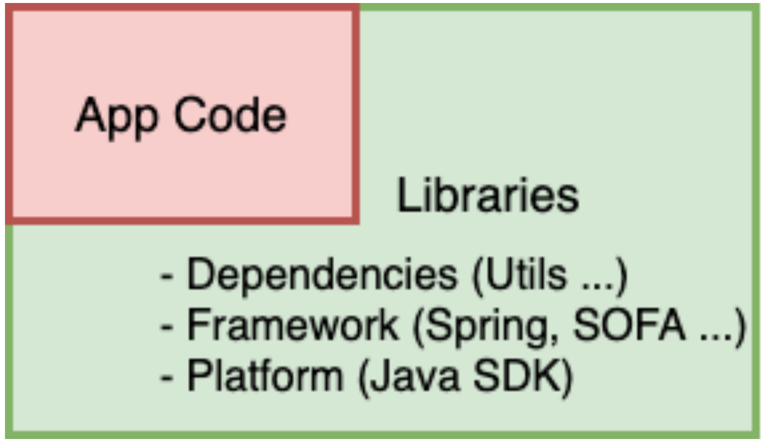
- Dependencies (Utils ...)
- Framework (Spring, SOFA ...)
- Platform (Java SDK)

Challenge2: Scalability



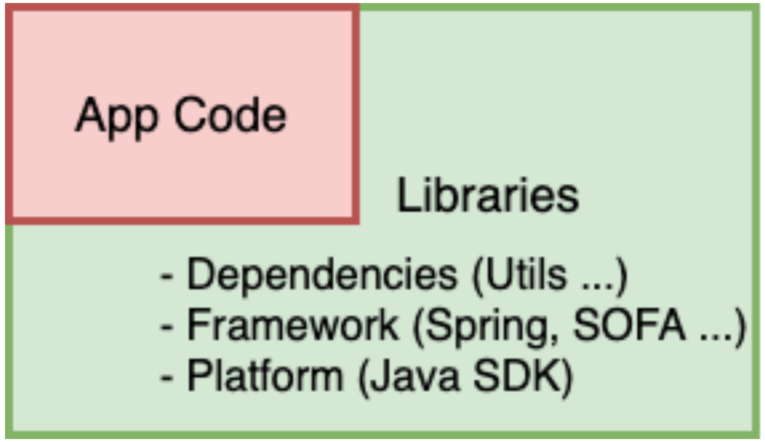
- Classes: dozens of thousands

Challenge2: Scalability



- Classes: dozens of thousands
- Jar Package: 100 MB+

Challenge2: Scalability



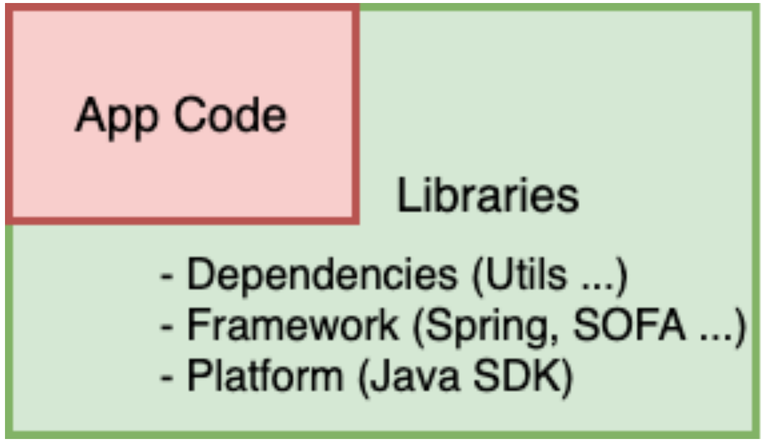
App Code

Libraries

- Dependencies (Utils ...)
- Framework (Spring, SOFA ...)
- Platform (Java SDK)

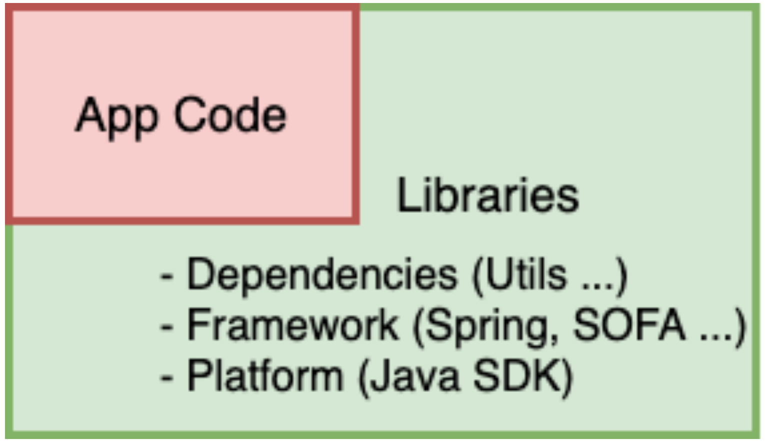
- Classes: dozens of thousands
- Jar Package: 100 MB+
- Services Methods: a few hundreds

Challenge2: Scalability



- Classes: dozens of thousands
- Jar Package: 100 MB+
- Services Methods: a few hundreds
- Interested Fields: dozens of thousands

Challenge2: Scalability



App Code

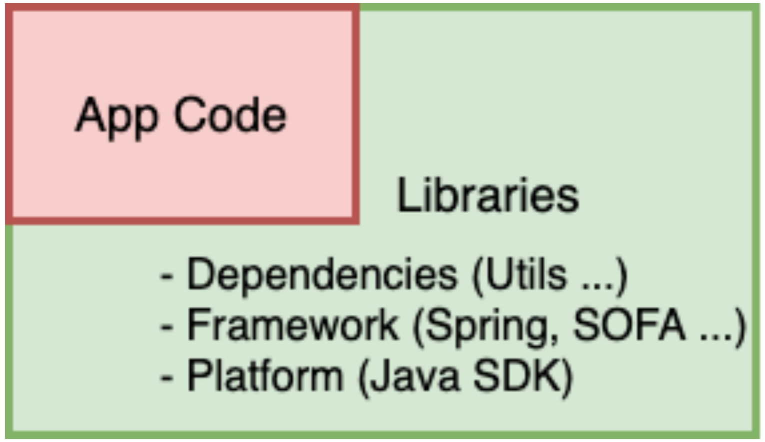
Libraries

- Dependencies (Utils ...)
- Framework (Spring, SOFA ...)
- Platform (Java SDK)

- Classes: dozens of thousands
- Jar Package: 100 MB+
- Services Methods: a few hundreds
- Interested Fields: dozens of thousands

- Massive amount of context information need to maintenance

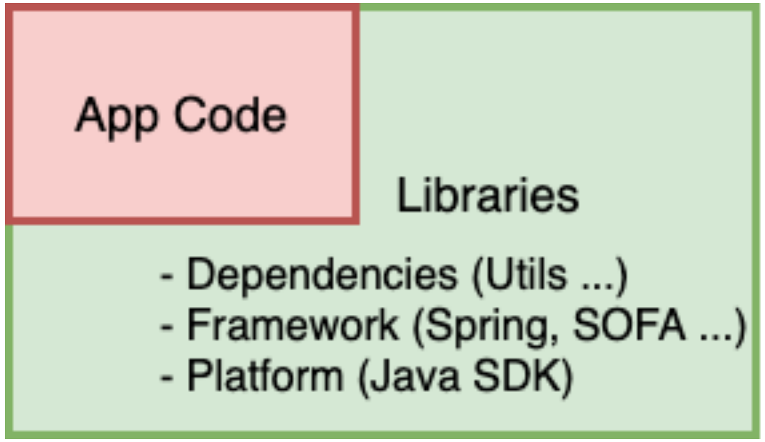
Challenge2: Scalability



- Classes: dozens of thousands
- Jar Package: 100 MB+
- Services Methods: a few hundreds
- Interested Fields: dozens of thousands

- Massive amount of context information need to maintenance
- Functions invoked under different contexts must be (re)analyzed

Challenge2: Scalability



App Code

Libraries

- Dependencies (Utils ...)
- Framework (Spring, SOFA ...)
- Platform (Java SDK)

- Classes: dozens of thousands
- Jar Package: 100 MB+
- Services Methods: a few hundreds
- Interested Fields: dozens of thousands

- Massive amount of context information need to maintenance
- Functions invoked under different contexts must be (re)analyzed

Existing tools run too long and use too much memory

Challenges

Recall

Challenge 1: Low Recall Rate

Unsound call-graph due to framework behaviours

Scalability

Challenge 2: Unscalable

Costly in both memory usage and time consumption

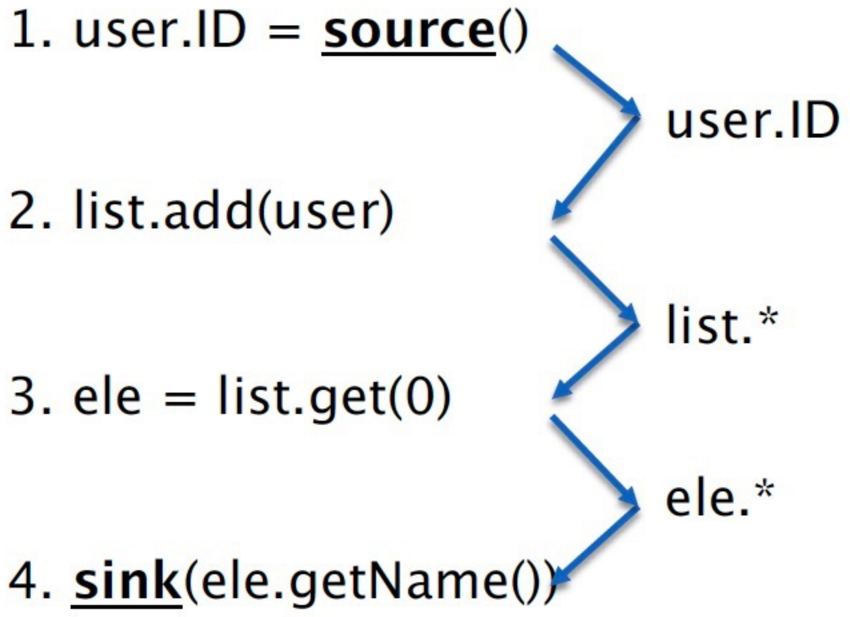
Precision

Challenge 3: Inaccuracy

Over-tainted due to inaccurate taint trace in container

Challenge3: Precision

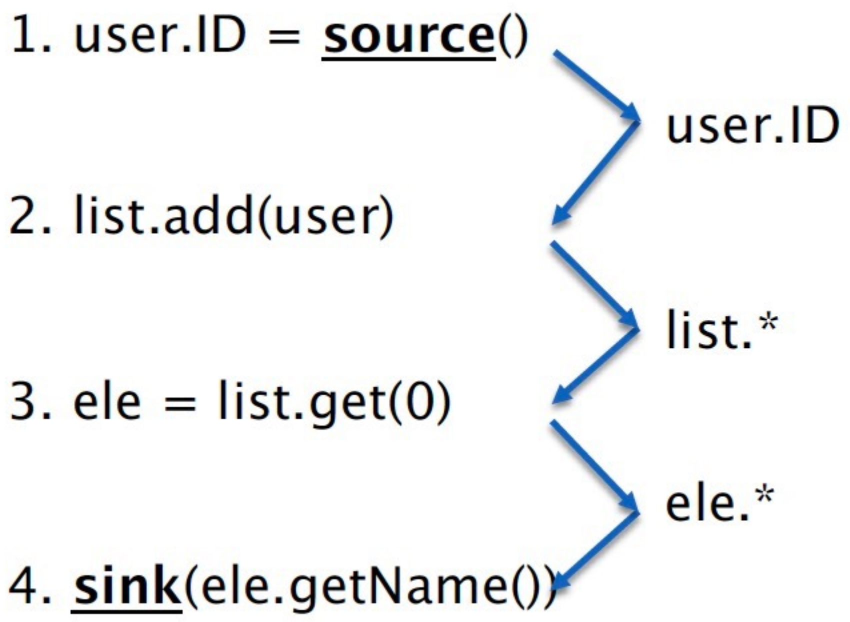
- A simple model introduces false positives



```
1. user.ID = source()
2. list.add(user)
3. ele = list.get(0)
4. sink(ele.getName())
```

Challenge3: Precision

- A simple model introduces false positives



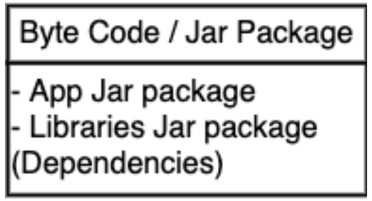
```
1. user.ID = source()
2. list.add(user)
3. ele = list.get(0)
4. sink(ele.getName())
```

ele.name is regarded as tainted ✗

UTS

Approach

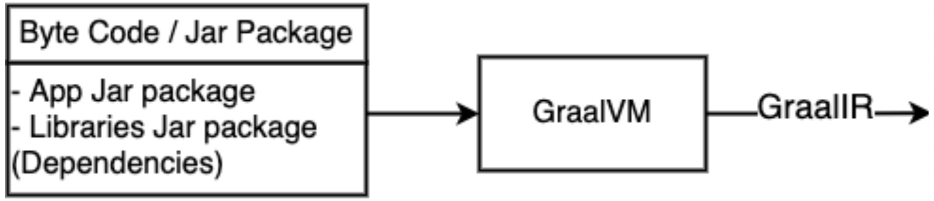
Overview of TaintFuzz



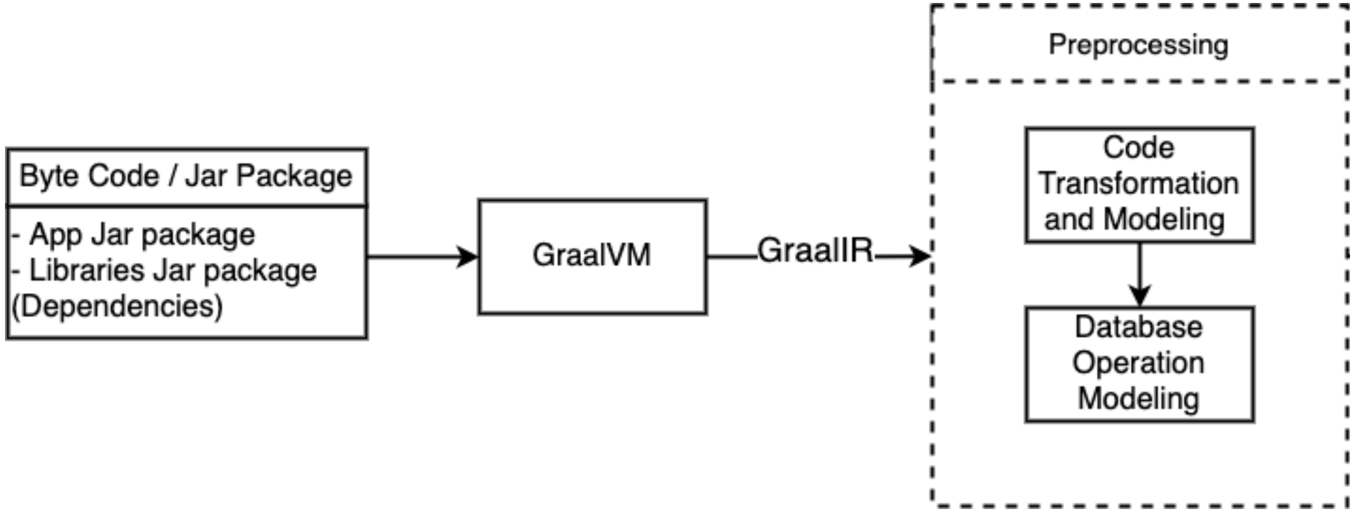
Byte Code / Jar Package

- App Jar package
- Libraries Jar package (Dependencies)

Overview of TaintFuzz



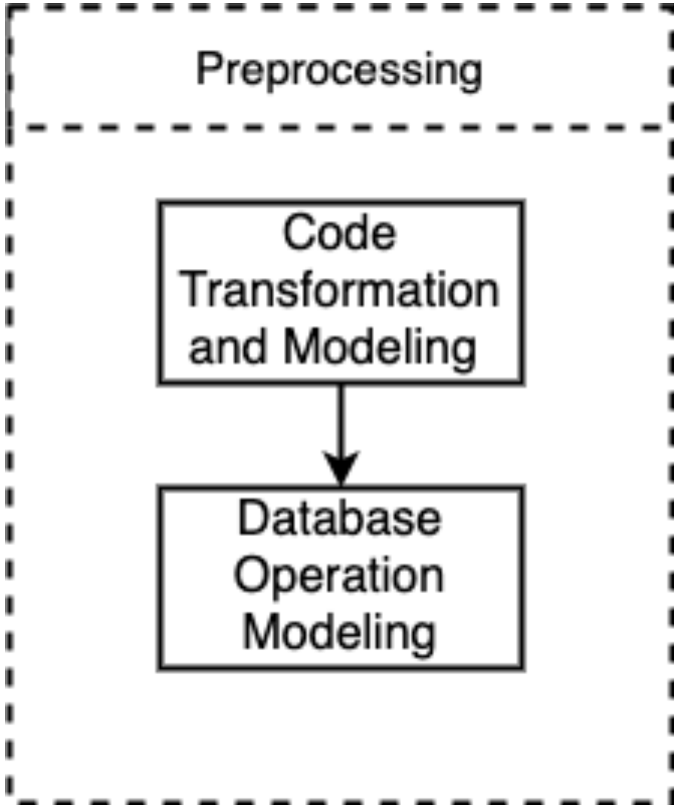
Overview of TaintFuzz



Preprocessing Module

Code Transformation and Modeling Module

Database Operation Modeling Module

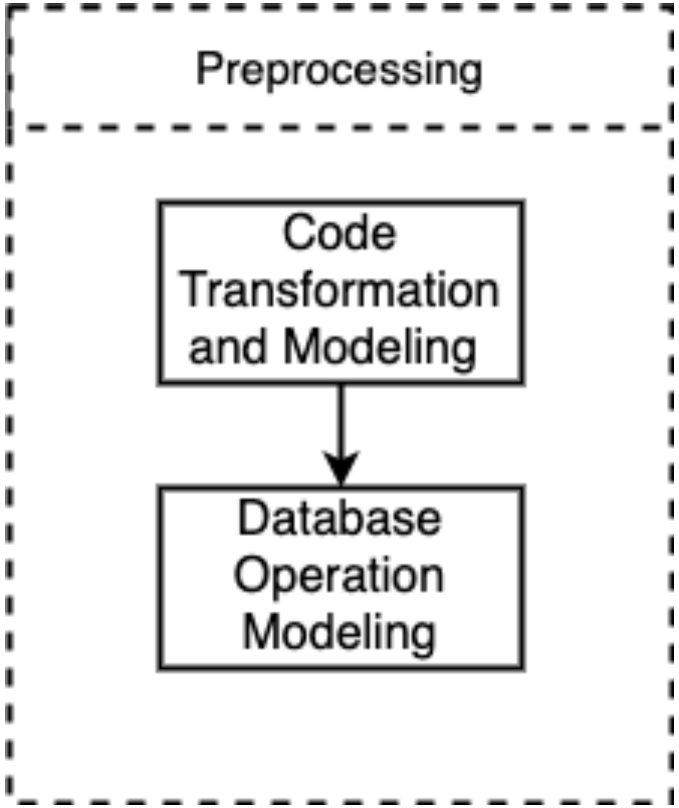


Preprocessing Module

Code Transformation and Modeling Module

- Framework code (e.g. RPC)

Database Operation Modeling Module

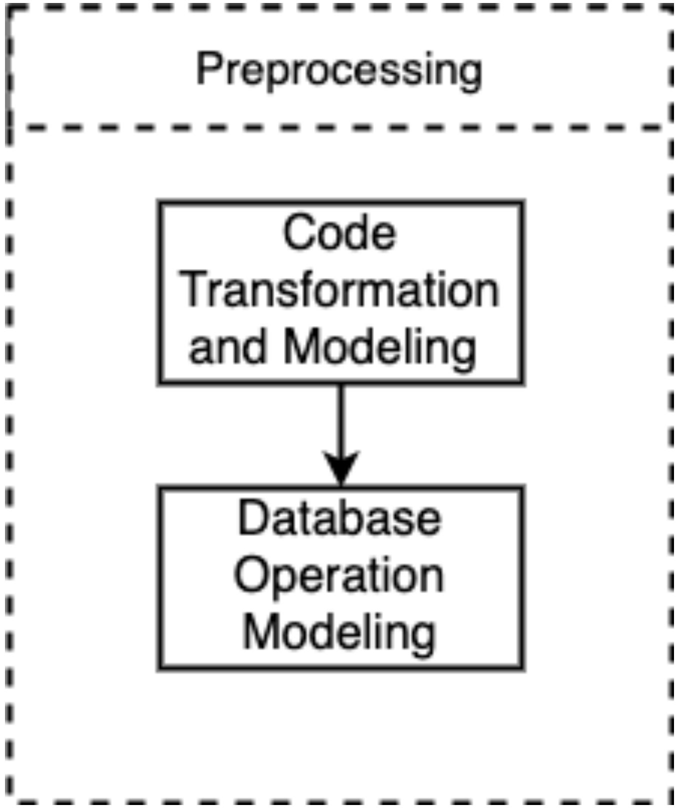


Preprocessing Module

Code Transformation and Modeling Module

- Framework code (e.g. RPC)
- Hidden code of the AOP invoke

Database Operation Modeling Module

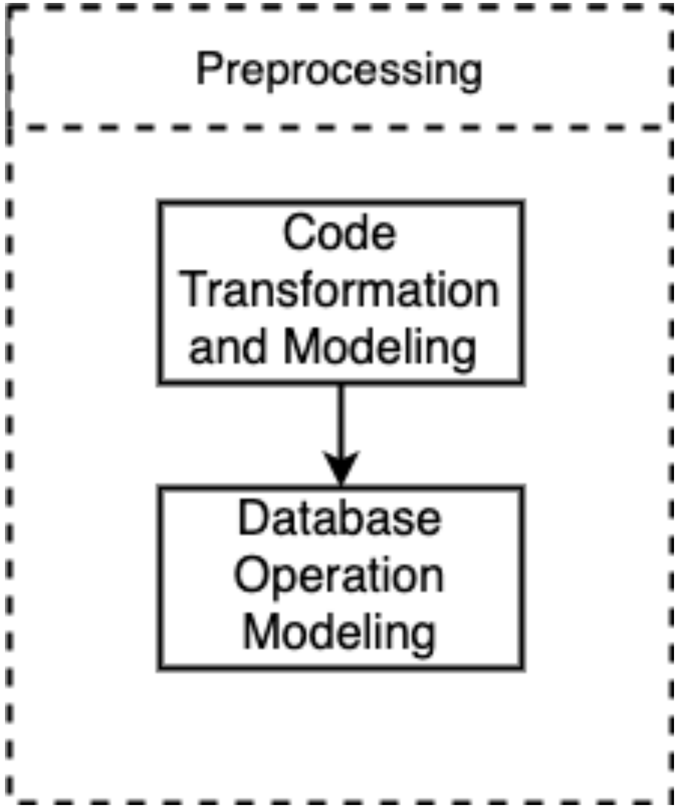


Preprocessing Module

Code Transformation and Modeling Module

- Framework code (e.g. RPC)
- Hidden code of the AOP invoke
- Operation of the container code (e.g. map.put(), list.add(), JSON.format(), etc.)

Database Operation Modeling Module



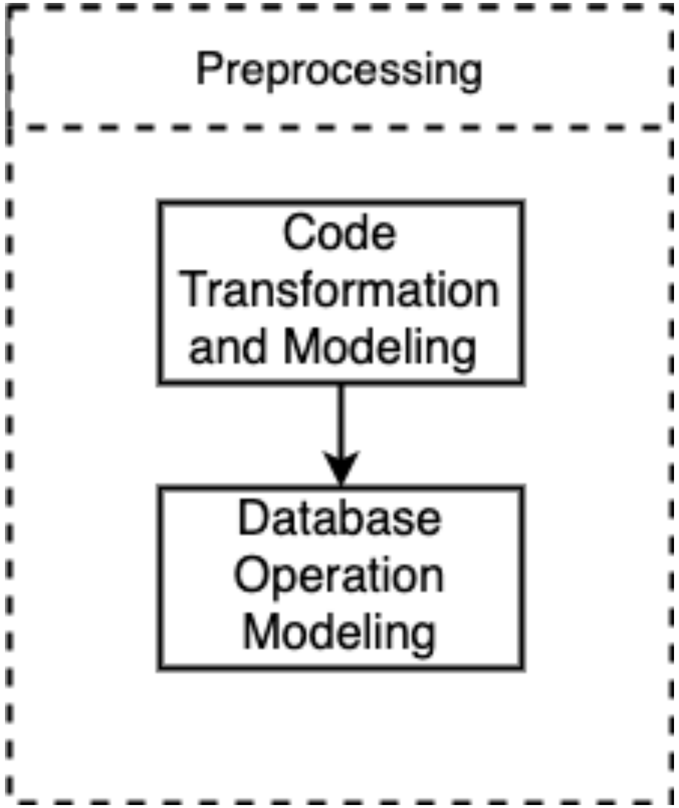
Preprocessing Module

Code Transformation and Modeling Module

- Framework code (e.g. RPC)
- Hidden code of the AOP invoke
- Operation of the container code (e.g. map.put(), list.add(), JSON.format(), etc.)

Database Operation Modeling Module

- Extract and analysis all SQL statement from the targeted code



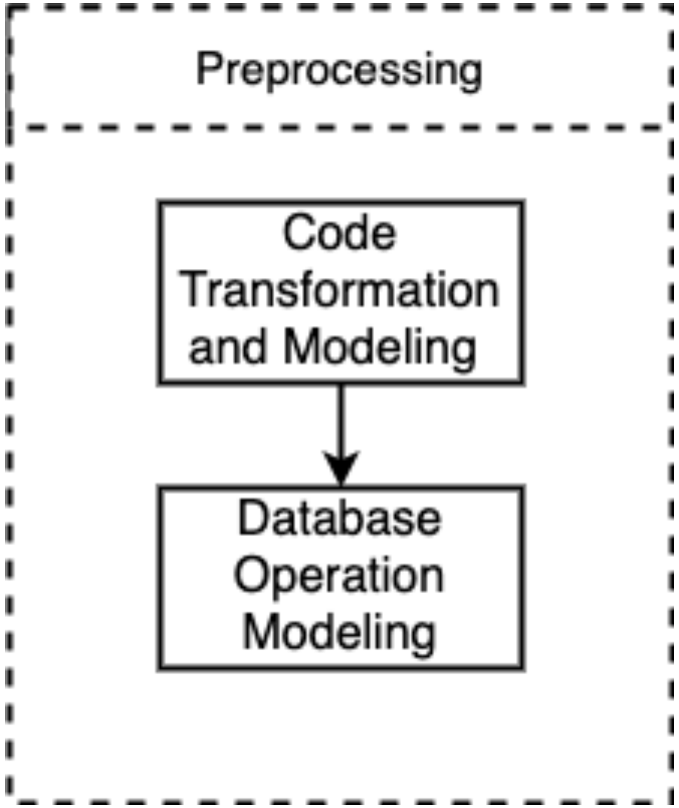
Preprocessing Module

Code Transformation and Modeling Module

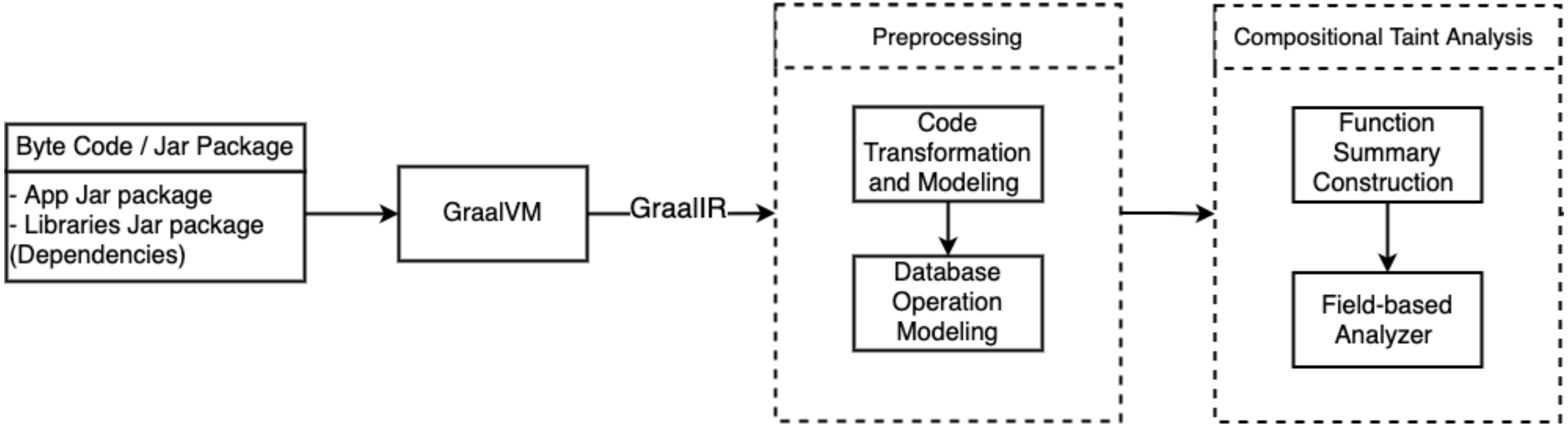
- Framework code (e.g. RPC)
- Hidden code of the AOP invoke
- Operation of the container code (e.g. map.put(), list.add(), JSON.format(), etc.)

Database Operation Modeling Module

- Extract and analysis all SQL statement from the targeted code
- Generate the relationship between variable and database column

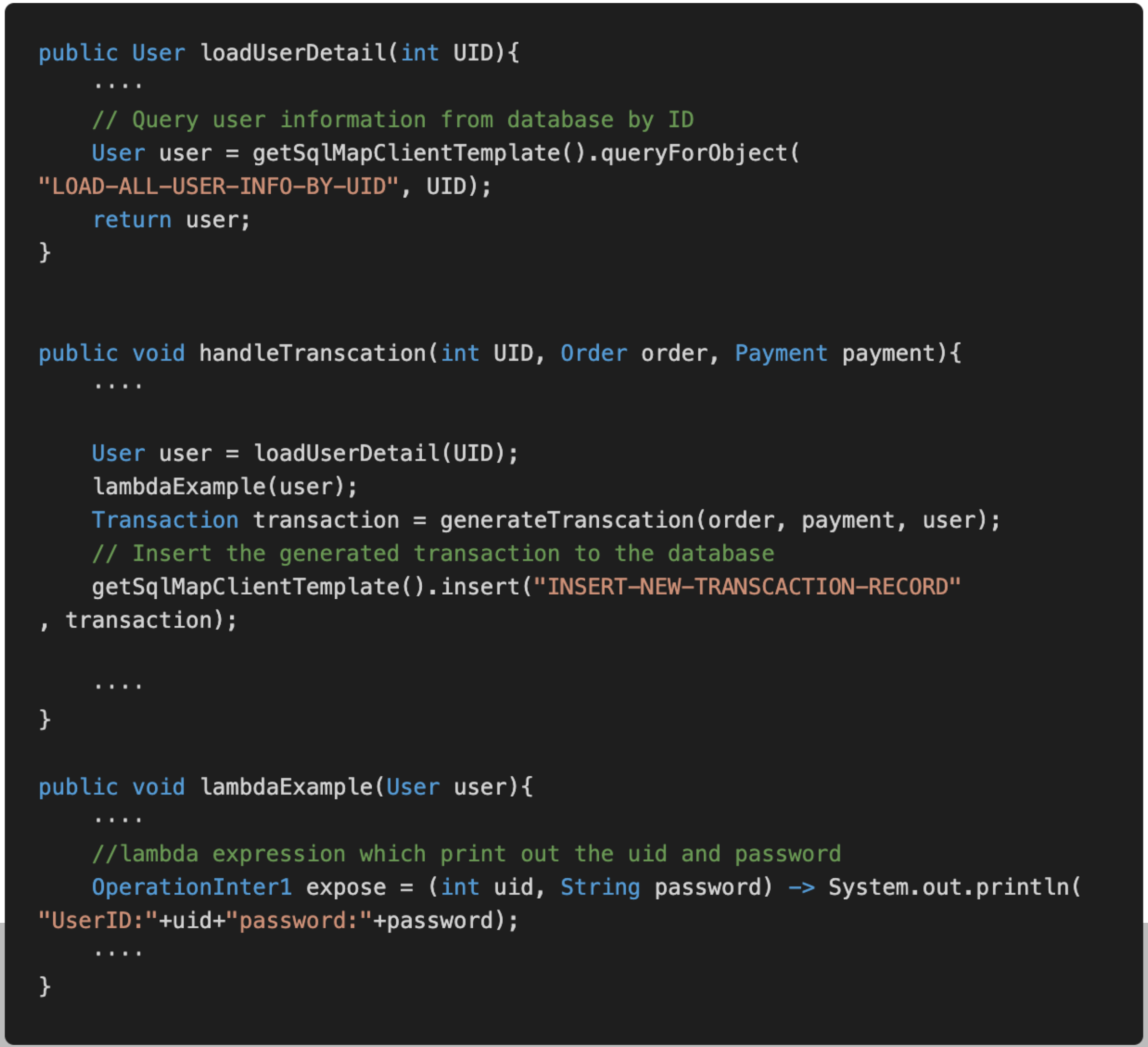


Overview of TaintFuzz

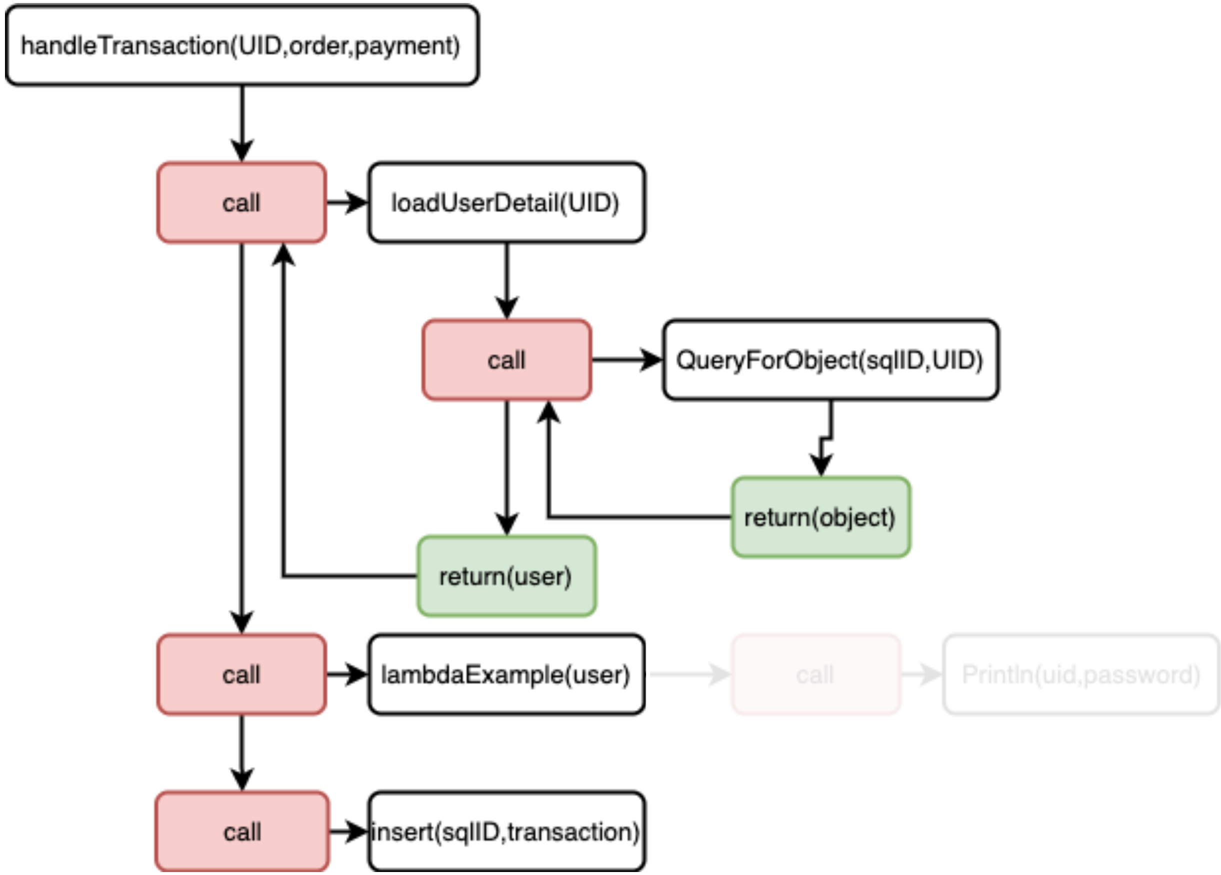


Recall: Filed-based Static Taint Analysis

Recall: Filed-based Static Taint Analysis

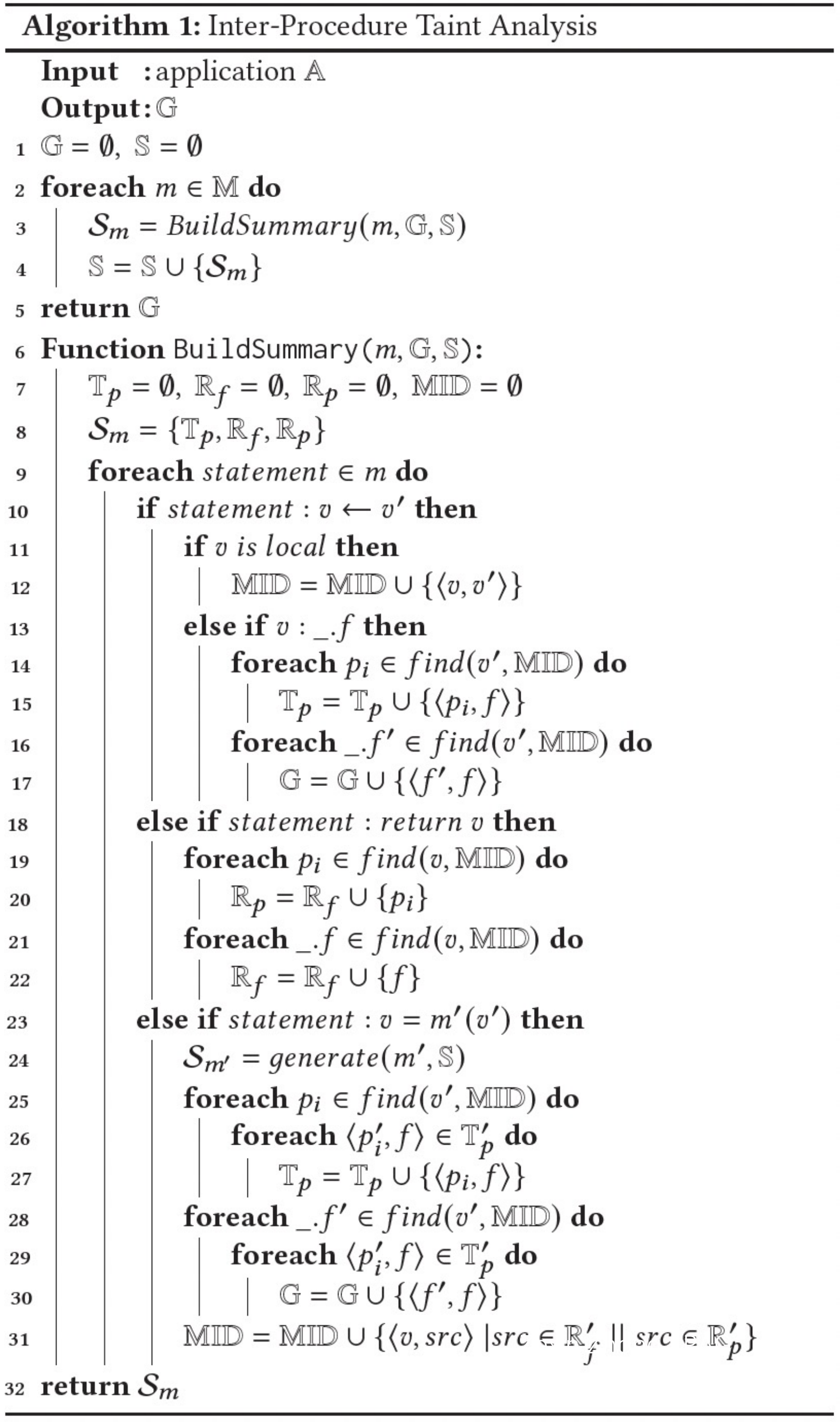


```
public User loadUserDetail(int UID){
    ....
    // Query user information from database by ID
    User user = getSqlMapClientTemplate().queryForObject(
"LOAD-ALL-USER-INFO-BY-UID", UID);
    return user;
}


public void handleTranscation(int UID, Order order, Payment payment){
    ....

    User user = loadUserDetail(UID);
    lambdaExample(user);
    Transaction transaction = generateTranscation(order, payment, user);
    // Insert the generated transaction to the database
    getSqlMapClientTemplate().insert("INSERT-NEW-TRANSCACTION-RECORD"
, transaction);

    ....
}

public void lambdaExample(User user){
    ....
    //lambda expression which print out the uid and password
    OperationInter1 expose = (int uid, String password) -> System.out.println(
"UserID:"+uid+"password:"+password);
    ....
}
```



- All propagation of the same field are related

Scalability: Function Summary Construction

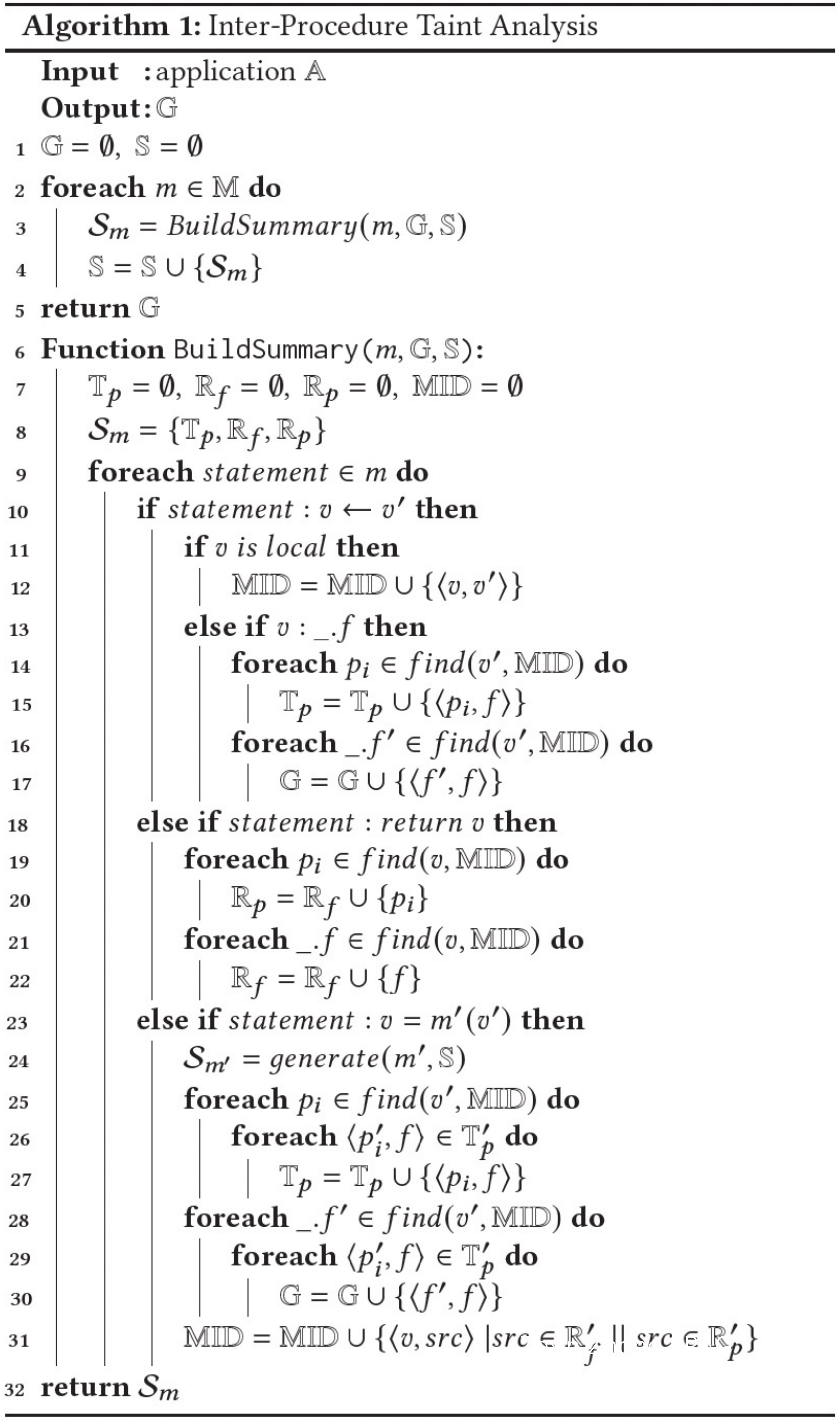


Algorithm 1: Inter-Procedure Taint Analysis

```
Input  : application $\mathbb{A}$
Output: $\mathbb{G}$
$\mathbb{G} = \emptyset,\ \mathbb{S} = \emptyset$
foreach $m \in \mathbb{M}$ do
    $\mathcal{S}_m = BuildSummary(m, \mathbb{G}, \mathbb{S})$
    $\mathbb{S} = \mathbb{S} \cup \{\mathcal{S}_m\}$
return $\mathbb{G}$
Function BuildSummary($m, \mathbb{G}, \mathbb{S}$):
    $\mathbb{T}_p = \emptyset,\ \mathbb{R}_f = \emptyset,\ \mathbb{R}_p = \emptyset,\ \mathbb{MID} = \emptyset$
    $\mathcal{S}_m = \{\mathbb{T}_p, \mathbb{R}_f, \mathbb{R}_p\}$
    foreach $statement \in m$ do
        if $statement : v \leftarrow v'$ then
            if $v$ $is\ local$ then
                $\mathbb{MID} = \mathbb{MID} \cup \{\langle v, v' \rangle\}$
            else if $v : \_.f$ then
                foreach $p_i \in find(v', \mathbb{MID})$ do
                    $\mathbb{T}_p = \mathbb{T}_p \cup \{\langle p_i, f \rangle\}$
                foreach $\_.f' \in find(v', \mathbb{MID})$ do
                    $\mathbb{G} = \mathbb{G} \cup \{\langle f', f \rangle\}$
        else if $statement : return\ v$ then
            foreach $p_i \in find(v, \mathbb{MID})$ do
                $\mathbb{R}_p = \mathbb{R}_f \cup \{p_i\}$
            foreach $\_.f \in find(v, \mathbb{MID})$ do
                $\mathbb{R}_f = \mathbb{R}_f \cup \{f\}$
        else if $statement : v = m'(v')$ then
            $\mathcal{S}_{m'} = generate(m', \mathbb{S})$
            foreach $p_i \in find(v', \mathbb{MID})$ do
                foreach $\langle p'_i, f \rangle \in \mathbb{T}'_p$ do
                    $\mathbb{T}_p = \mathbb{T}_p \cup \{\langle p_i, f \rangle\}$
            foreach $\_.f' \in find(v', \mathbb{MID})$ do
                foreach $\langle p'_i, f \rangle \in \mathbb{T}'_p$ do
                    $\mathbb{G} = \mathbb{G} \cup \{\langle f', f \rangle\}$
            $\mathbb{MID} = \mathbb{MID} \cup \{\langle v, src \rangle \mid src \in \mathbb{R}'_f \mid\mid src \in \mathbb{R}'_p\}$
    return $\mathcal{S}_m$
```

Scalability: Function Summary Construction

- Start the analysis from any function randomly

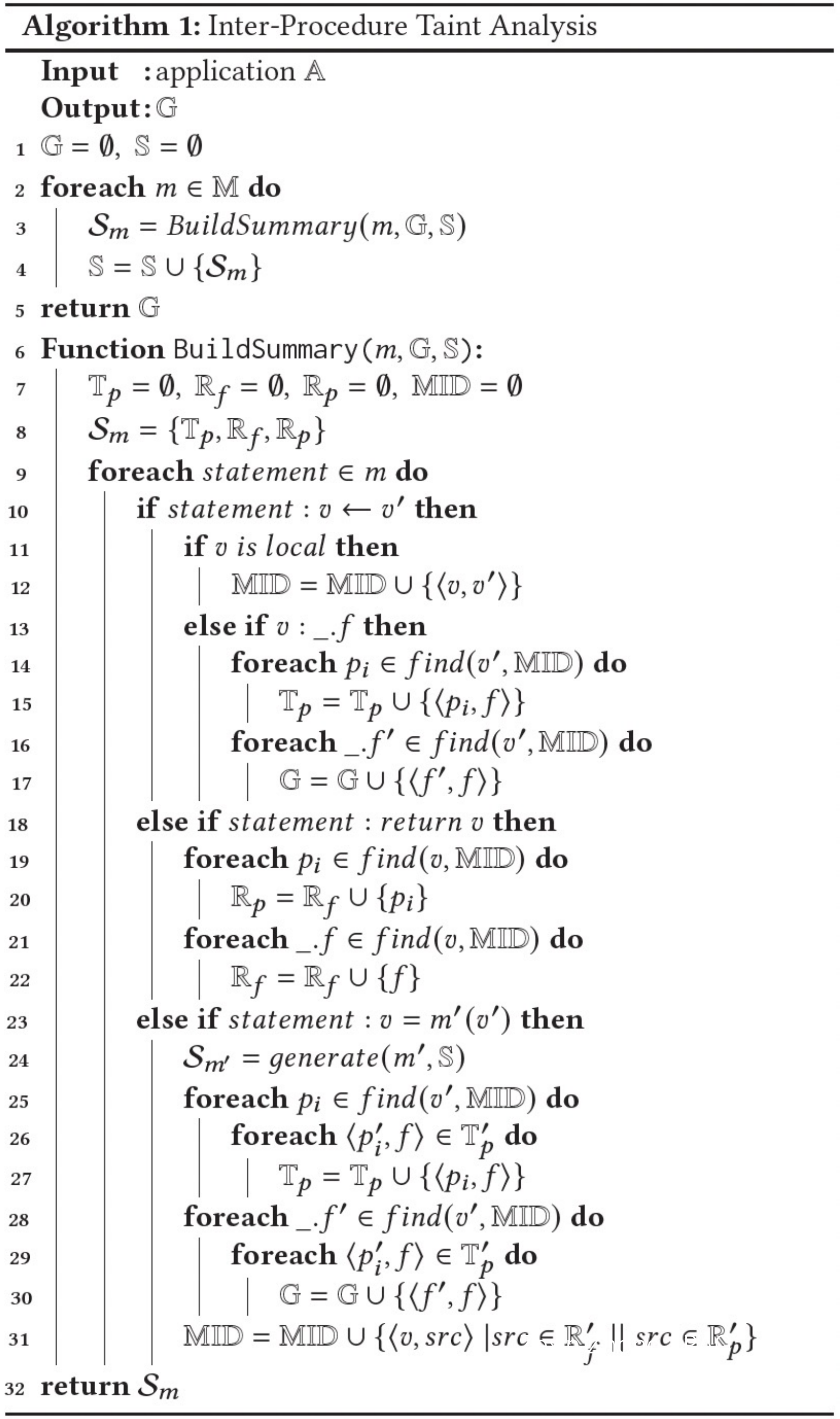


Algorithm 1: Inter-Procedure Taint Analysis

```
Input  : application $\mathbb{A}$
Output: $\mathbb{G}$
$\mathbb{G} = \emptyset,\ \mathbb{S} = \emptyset$
foreach $m \in \mathbb{M}$ do
    $\mathcal{S}_m = BuildSummary(m, \mathbb{G}, \mathbb{S})$
    $\mathbb{S} = \mathbb{S} \cup \{\mathcal{S}_m\}$
return $\mathbb{G}$
Function BuildSummary($m, \mathbb{G}, \mathbb{S}$):
    $\mathbb{T}_p = \emptyset,\ \mathbb{R}_f = \emptyset,\ \mathbb{R}_p = \emptyset,\ \mathbb{MID} = \emptyset$
    $\mathcal{S}_m = \{\mathbb{T}_p, \mathbb{R}_f, \mathbb{R}_p\}$
    foreach $statement \in m$ do
        if $statement : v \leftarrow v'$ then
            if $v$ is local then
                $\mathbb{MID} = \mathbb{MID} \cup \{\langle v, v' \rangle\}$
            else if $v : \_.f$ then
                foreach $p_i \in find(v', \mathbb{MID})$ do
                    $\mathbb{T}_p = \mathbb{T}_p \cup \{\langle p_i, f \rangle\}$
                foreach $\_.f' \in find(v', \mathbb{MID})$ do
                    $\mathbb{G} = \mathbb{G} \cup \{\langle f', f \rangle\}$
        else if $statement : return\ v$ then
            foreach $p_i \in find(v, \mathbb{MID})$ do
                $\mathbb{R}_p = \mathbb{R}_f \cup \{p_i\}$
            foreach $\_.f \in find(v, \mathbb{MID})$ do
                $\mathbb{R}_f = \mathbb{R}_f \cup \{f\}$
        else if $statement : v = m'(v')$ then
            $\mathcal{S}_{m'} = generate(m', \mathbb{S})$
            foreach $p_i \in find(v', \mathbb{MID})$ do
                foreach $\langle p_i', f \rangle \in \mathbb{T}_p'$ do
                    $\mathbb{T}_p = \mathbb{T}_p \cup \{\langle p_i, f \rangle\}$
            foreach $\_.f' \in find(v', \mathbb{MID})$ do
                foreach $\langle p_i', f \rangle \in \mathbb{T}_p'$ do
                    $\mathbb{G} = \mathbb{G} \cup \{\langle f', f \rangle\}$
            $\mathbb{MID} = \mathbb{MID} \cup \{\langle v, src \rangle \mid src \in \mathbb{R}_f' \mid\mid src \in \mathbb{R}_p'\}$
return $\mathcal{S}_m$
```

Scalability: Function Summary Construction

- Start the analysis from any function randomly
- Generate a data propagation summary for each function

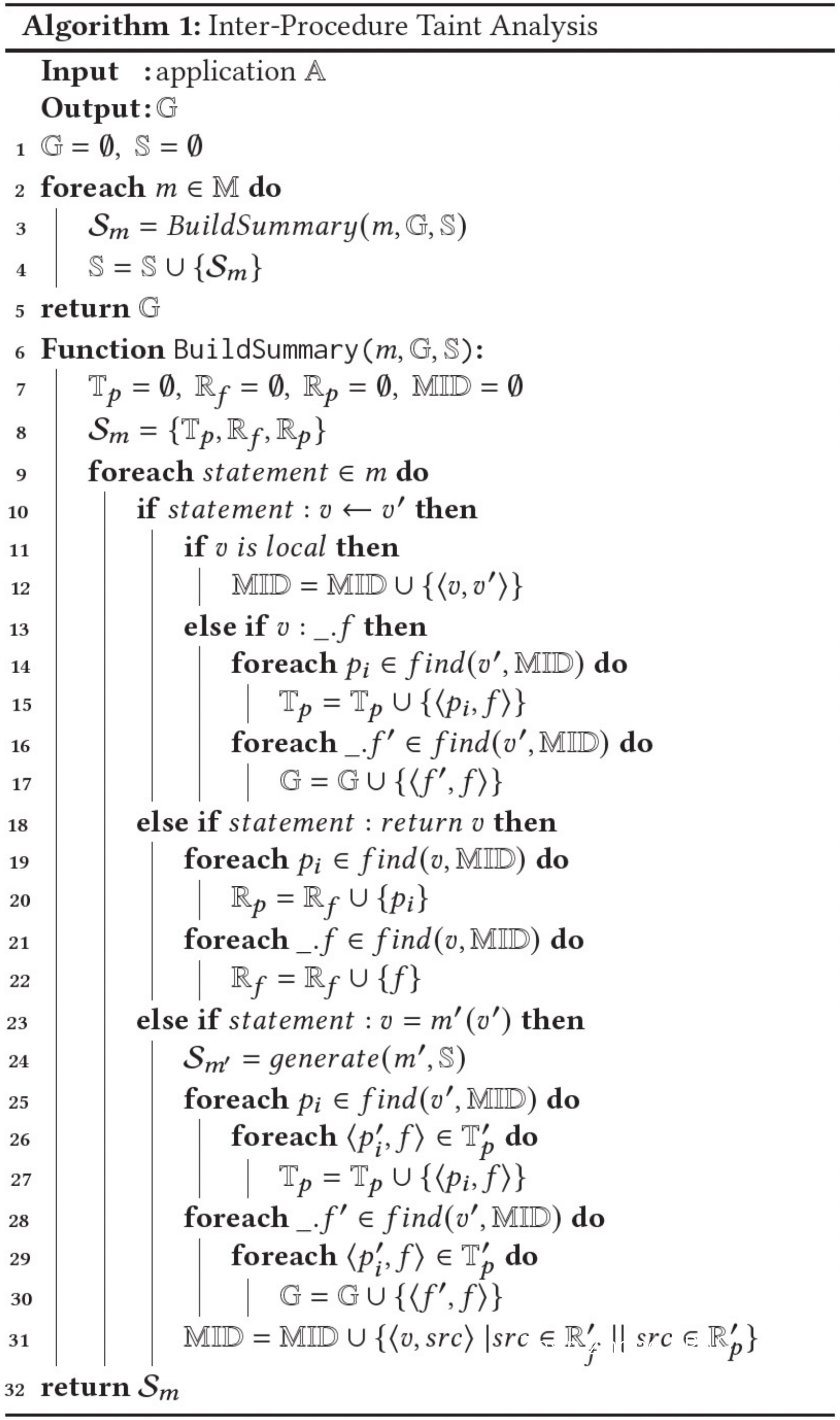


Algorithm 1: Inter-Procedure Taint Analysis

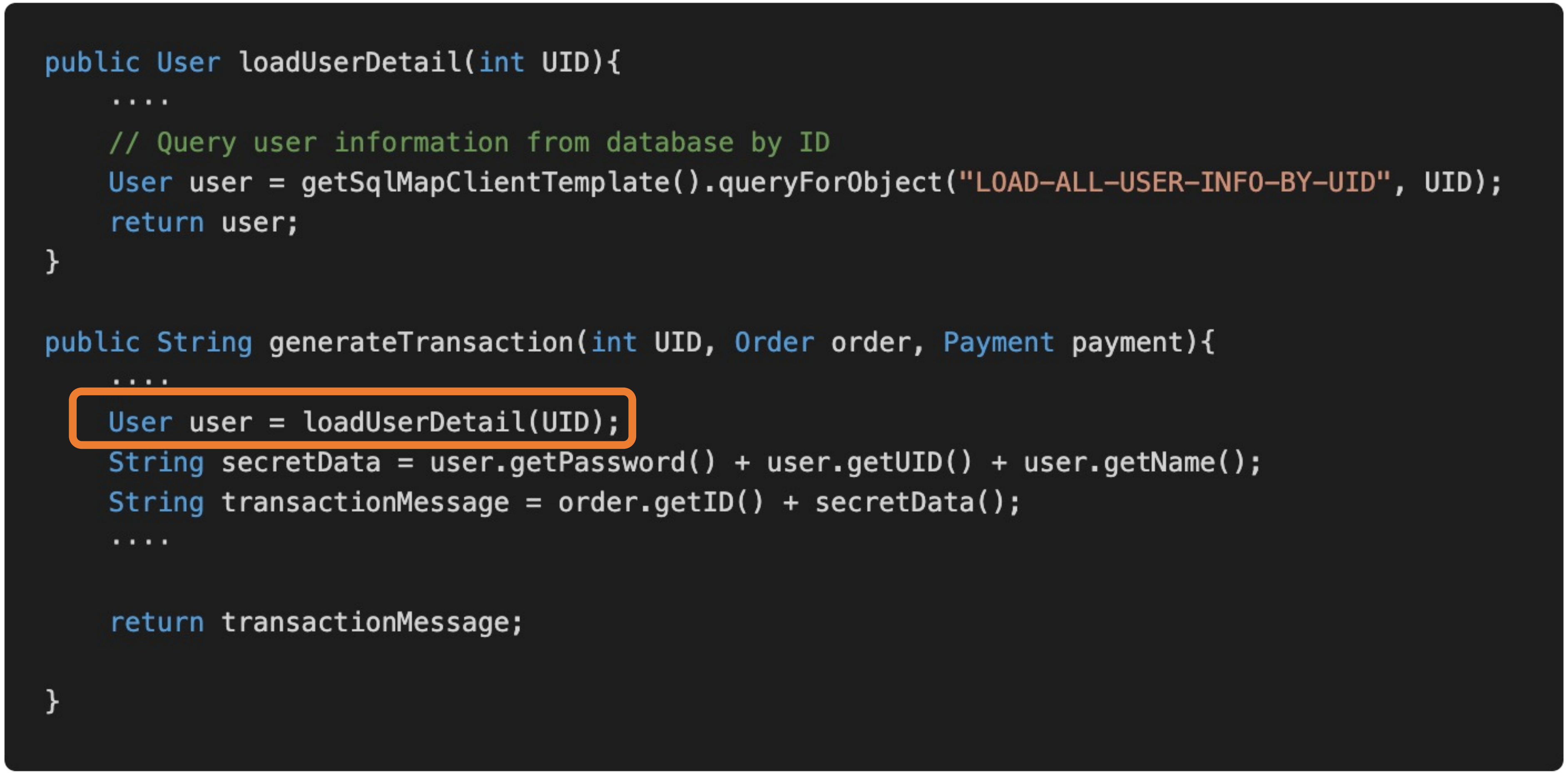
```
Input  : application $\mathbb{A}$
Output: $\mathbb{G}$
$\mathbb{G} = \emptyset,\ \mathbb{S} = \emptyset$
foreach $m \in \mathbb{M}$ do
    $\mathcal{S}_m = BuildSummary(m, \mathbb{G}, \mathbb{S})$
    $\mathbb{S} = \mathbb{S} \cup \{\mathcal{S}_m\}$
return $\mathbb{G}$
Function BuildSummary($m, \mathbb{G}, \mathbb{S}$):
    $\mathbb{T}_p = \emptyset,\ \mathbb{R}_f = \emptyset,\ \mathbb{R}_p = \emptyset,\ \mathbb{MID} = \emptyset$
    $\mathcal{S}_m = \{\mathbb{T}_p, \mathbb{R}_f, \mathbb{R}_p\}$
    foreach $statement \in m$ do
        if $statement : v \leftarrow v'$ then
            if $v$ is local then
                $\mathbb{MID} = \mathbb{MID} \cup \{\langle v, v' \rangle\}$
            else if $v : \_.f$ then
                foreach $p_i \in find(v', \mathbb{MID})$ do
                    $\mathbb{T}_p = \mathbb{T}_p \cup \{\langle p_i, f \rangle\}$
                foreach $\_.f' \in find(v', \mathbb{MID})$ do
                    $\mathbb{G} = \mathbb{G} \cup \{\langle f', f \rangle\}$
        else if $statement : return\ v$ then
            foreach $p_i \in find(v, \mathbb{MID})$ do
                $\mathbb{R}_p = \mathbb{R}_f \cup \{p_i\}$
            foreach $\_.f \in find(v, \mathbb{MID})$ do
                $\mathbb{R}_f = \mathbb{R}_f \cup \{f\}$
        else if $statement : v = m'(v')$ then
            $\mathcal{S}_{m'} = generate(m', \mathbb{S})$
            foreach $p_i \in find(v', \mathbb{MID})$ do
                foreach $\langle p'_i, f \rangle \in \mathbb{T}'_p$ do
                    $\mathbb{T}_p = \mathbb{T}_p \cup \{\langle p_i, f \rangle\}$
            foreach $\_.f' \in find(v', \mathbb{MID})$ do
                foreach $\langle p'_i, f \rangle \in \mathbb{T}'_p$ do
                    $\mathbb{G} = \mathbb{G} \cup \{\langle f', f \rangle\}$
            $\mathbb{MID} = \mathbb{MID} \cup \{\langle v, src \rangle \mid src \in \mathbb{R}'_f \mid\mid src \in \mathbb{R}'_p\}$
return $\mathcal{S}_m$
```

Scalability: Function Summary Construction

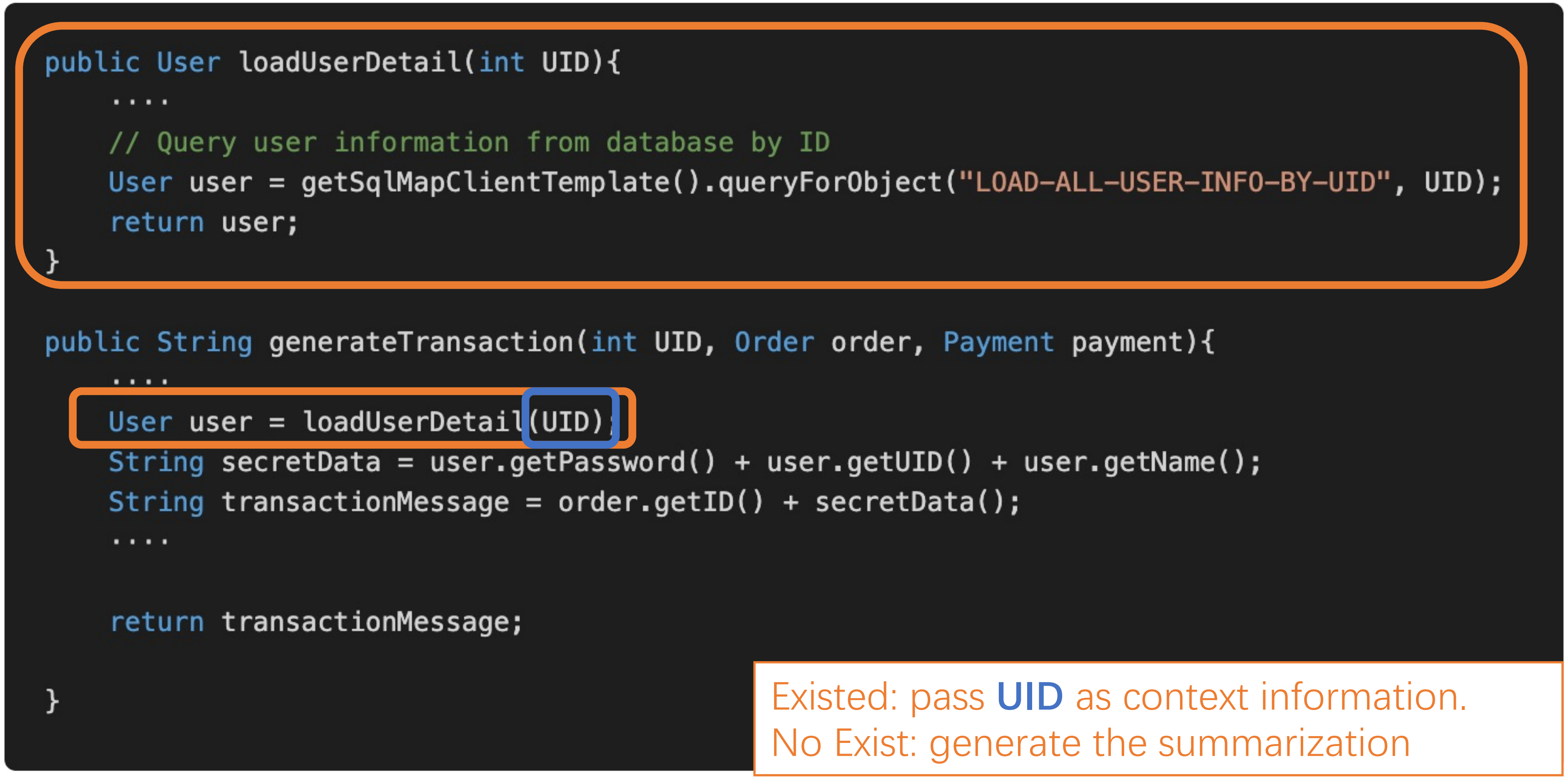
- Start the analysis from any function randomly
- Generate a data propagation summary for each function
  - Only maintenance the context information of the current function
  - On demand call analysis with the function summary



```
Algorithm 1: Inter-Procedure Taint Analysis
Input  : application $\mathbb{A}$
Output: $\mathbb{G}$
$\mathbb{G} = \emptyset,\ \mathbb{S} = \emptyset$
foreach $m \in \mathbb{M}$ do
    $\mathcal{S}_m = BuildSummary(m, \mathbb{G}, \mathbb{S})$
    $\mathbb{S} = \mathbb{S} \cup \{\mathcal{S}_m\}$
return $\mathbb{G}$
Function BuildSummary($m, \mathbb{G}, \mathbb{S}$):
    $\mathbb{T}_p = \emptyset,\ \mathbb{R}_f = \emptyset,\ \mathbb{R}_p = \emptyset,\ \mathbb{MID} = \emptyset$
    $\mathcal{S}_m = \{\mathbb{T}_p, \mathbb{R}_f, \mathbb{R}_p\}$
    foreach $statement \in m$ do
        if $statement : v \leftarrow v'$ then
            if $v$ is local then
                $\mathbb{MID} = \mathbb{MID} \cup \{\langle v, v' \rangle\}$
            else if $v : \_.f$ then
                foreach $p_i \in find(v', \mathbb{MID})$ do
                    $\mathbb{T}_p = \mathbb{T}_p \cup \{\langle p_i, f \rangle\}$
                foreach $\_.f' \in find(v', \mathbb{MID})$ do
                    $\mathbb{G} = \mathbb{G} \cup \{\langle f', f \rangle\}$
        else if $statement : return\ v$ then
            foreach $p_i \in find(v, \mathbb{MID})$ do
                $\mathbb{R}_p = \mathbb{R}_f \cup \{p_i\}$
            foreach $\_.f \in find(v, \mathbb{MID})$ do
                $\mathbb{R}_f = \mathbb{R}_f \cup \{f\}$
        else if $statement : v = m'(v')$ then
            $\mathcal{S}_{m'} = generate(m', \mathbb{S})$
            foreach $p_i \in find(v', \mathbb{MID})$ do
                foreach $\langle p'_i, f \rangle \in \mathbb{T}'_p$ do
                    $\mathbb{T}_p = \mathbb{T}_p \cup \{\langle p_i, f \rangle\}$
            foreach $\_.f' \in find(v', \mathbb{MID})$ do
                foreach $\langle p'_i, f \rangle \in \mathbb{T}'_p$ do
                    $\mathbb{G} = \mathbb{G} \cup \{\langle f', f \rangle\}$
            $\mathbb{MID} = \mathbb{MID} \cup \{\langle v, src \rangle \mid src \in \mathbb{R}'_f \mid\mid src \in \mathbb{R}'_p\}$
return $\mathcal{S}_m$
```

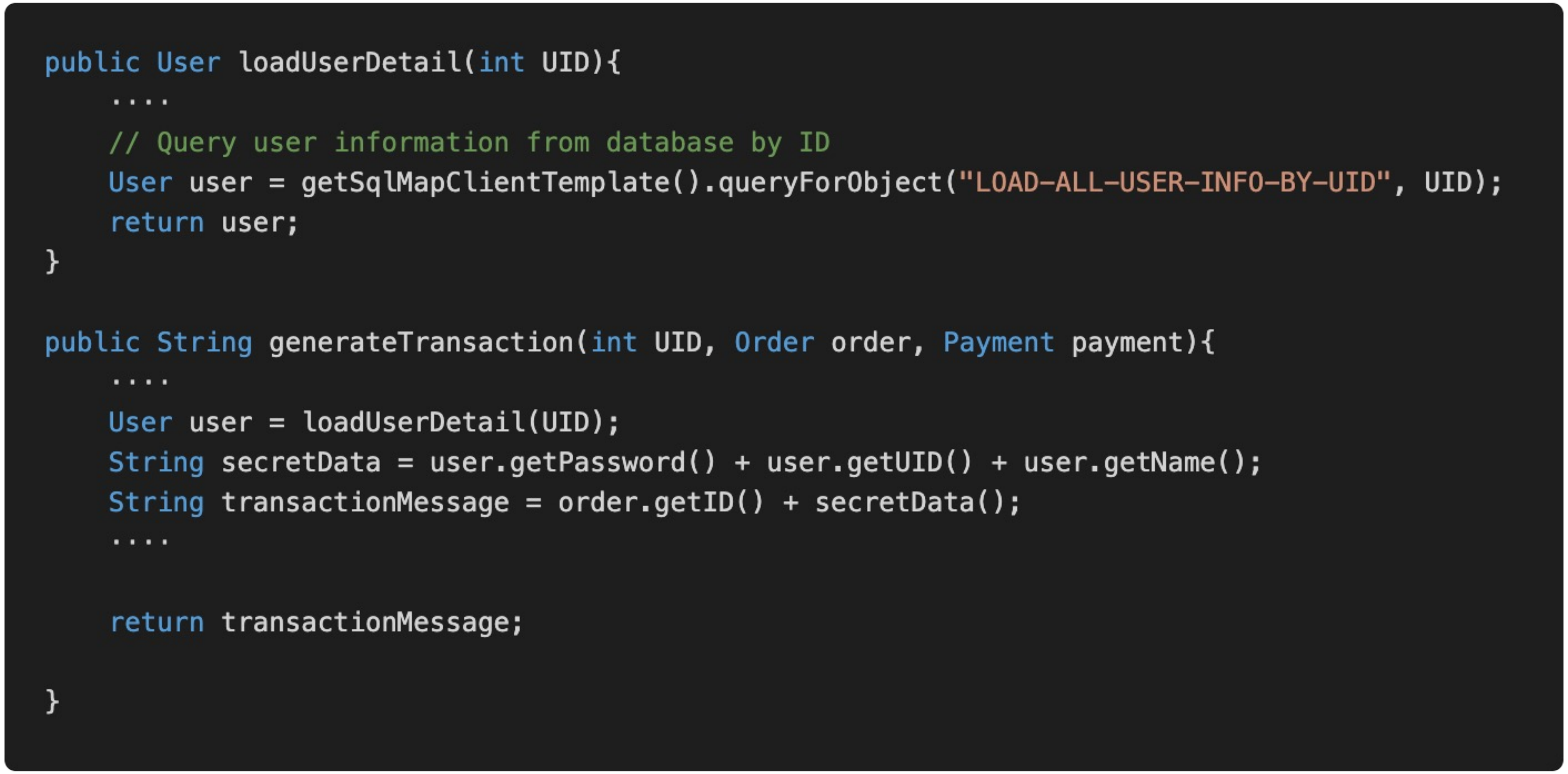


```java
public User loadUserDetail(int UID){
    ....
    // Query user information from database by ID
    User user = getSqlMapClientTemplate().queryForObject("LOAD-ALL-USER-INFO-BY-UID", UID);
    return user;
}

public String generateTransaction(int UID, Order order, Payment payment){
    ....
    User user = loadUserDetail(UID);
    String secretData = user.getPassword() + user.getUID() + user.getName();
    String transactionMessage = order.getID() + secretData();
    ....

    return transactionMessage;

}
```



```
public User loadUserDetail(int UID){
    ....
    // Query user information from database by ID
    User user = getSqlMapClientTemplate().queryForObject("LOAD-ALL-USER-INFO-BY-UID", UID);
    return user;
}

public String generateTransaction(int UID, Order order, Payment payment){
    ....
    User user = loadUserDetail(UID);
    String secretData = user.getPassword() + user.getUID() + user.getName();
    String transactionMessage = order.getID() + secretData();
    ....

    return transactionMessage;

}
```

Existed: pass **UID** as context information.
No Exist: generate the summarization

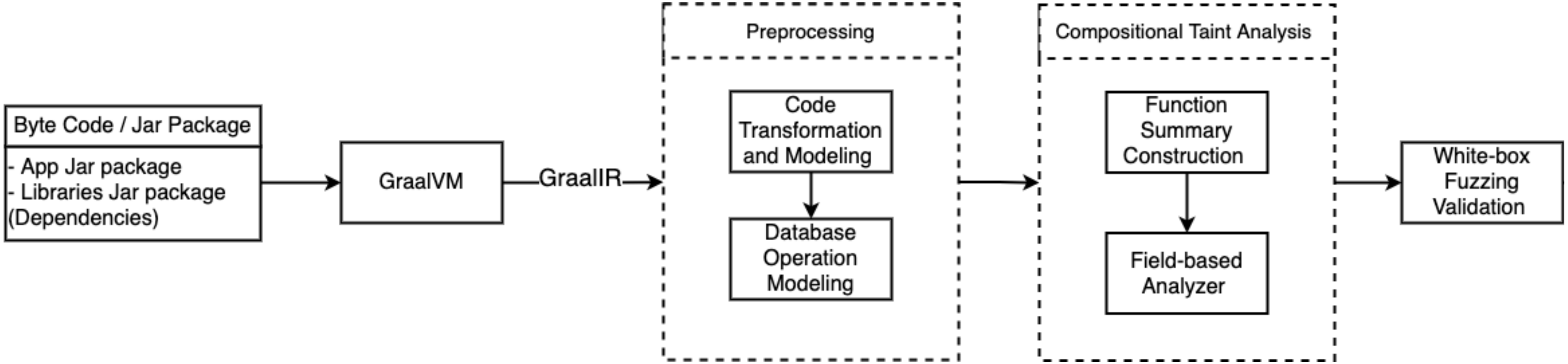


```java
public User loadUserDetail(int UID){
    ....
    // Query user information from database by ID
    User user = getSqlMapClientTemplate().queryForObject("LOAD-ALL-USER-INFO-BY-UID", UID);
    return user;
}

public String generateTransaction(int UID, Order order, Payment payment){
    ....
    User user = loadUserDetail(UID);
    String secretData = user.getPassword() + user.getUID() + user.getName();
    String transactionMessage = order.getID() + secretData();
    ....

    return transactionMessage;

}
```

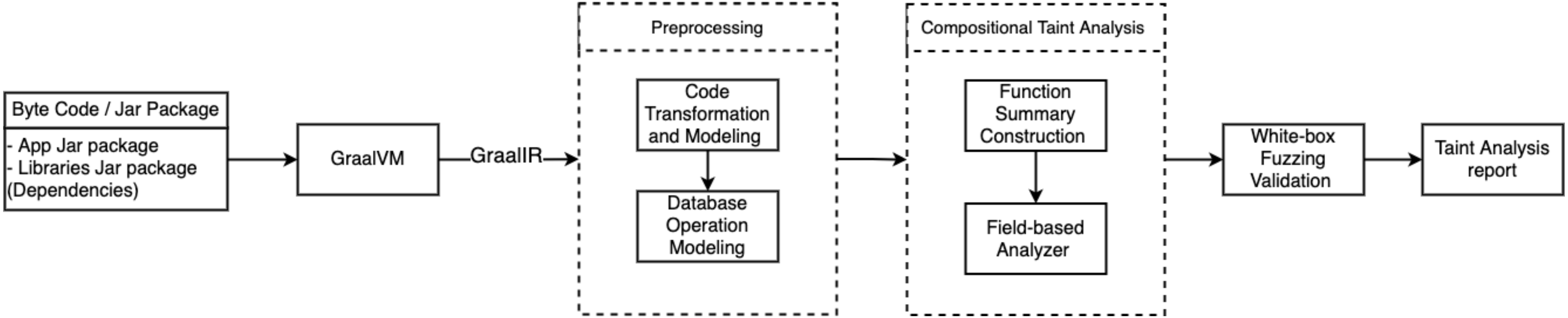
{ [SQL:LOAD-ALL-USER -> returnObject] -> user

-> secretData -> transactionMessage -> returnObj}

Overview of TaintFuzz


Byte Code / Jar Package
- App Jar package
- Libraries Jar package
(Dependencies)
GraalVM
GraalIR
Preprocessing
Code Transformation and Modeling
Database Operation Modeling
Compositional Taint Analysis
Function Summary Construction
Field-based Analyzer
White-box Fuzzing Validation

Overview of TaintFuzz



UTS

Evaluation

Datasets

- Open-source Micro-benchmark

- Production-benchmark

Datasets

- Open-source Micro-benchmark
  - ◆ Provides by Wang et al. (2020) on GitHub
  - ◆ Contributed by the industry experts
  - ◆ Including 27 cases that developers are concerned with in the industry scenario
- Production-benchmark

Datasets

- Open-source Micro-benchmark
  - ◆ Provides by Wang et al. (2020) on GitHub
  - ◆ Contributed by the industry experts
  - ◆ Including 27 cases that developers are concerned with in the industry scenario
- Production-benchmark
  - ◆ Contains more than 500 core production microservices applications of the industry partner
  - ◆ Written in SOFA architecture

Research Questions

- *RQ1:* What is the performance of our approach when it is applied on real world industrial cases ?

Research Questions

- *RQ1:* What is the performance of our approach when it is applied on real world industrial cases ?

- *RQ2:* How well does our approach in terms of precision and recall for sensitive data tracing on industrial microservices applications?

Research Questions

- *RQ1:* What is the performance of our approach when it is applied on real world industrial cases ?
- *RQ2:* How well does our approach in terms of precision and recall for sensitive data tracing on industrial microservices applications?
- *RQ3:* How TaintFuzz is compared to existing tools, such as ANTaint?

Evaluation: Performance

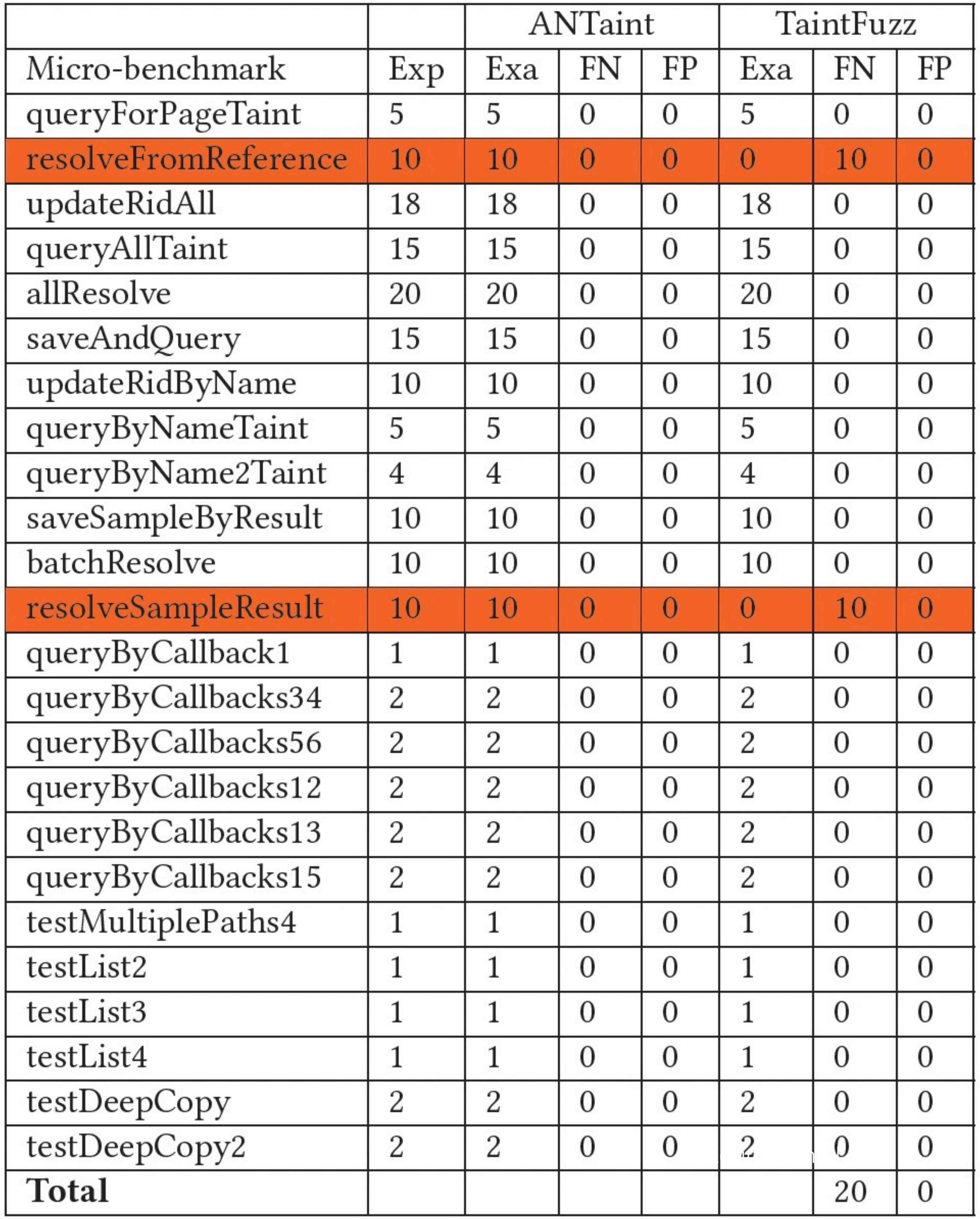
RQ1: What is the performance of our approach when it is applied on real world industrial cases ?

- Evaluate on the Open-source Micro-benchmark

Evaluation: Performance

RQ1: What is the performance of our approach when it is applied on real world industrial cases ?

- Evaluate on the Open-source Micro-benchmark
  - Pass 93% (25 out of 27 cases) of the test scenarios
  - With the precision of 100%



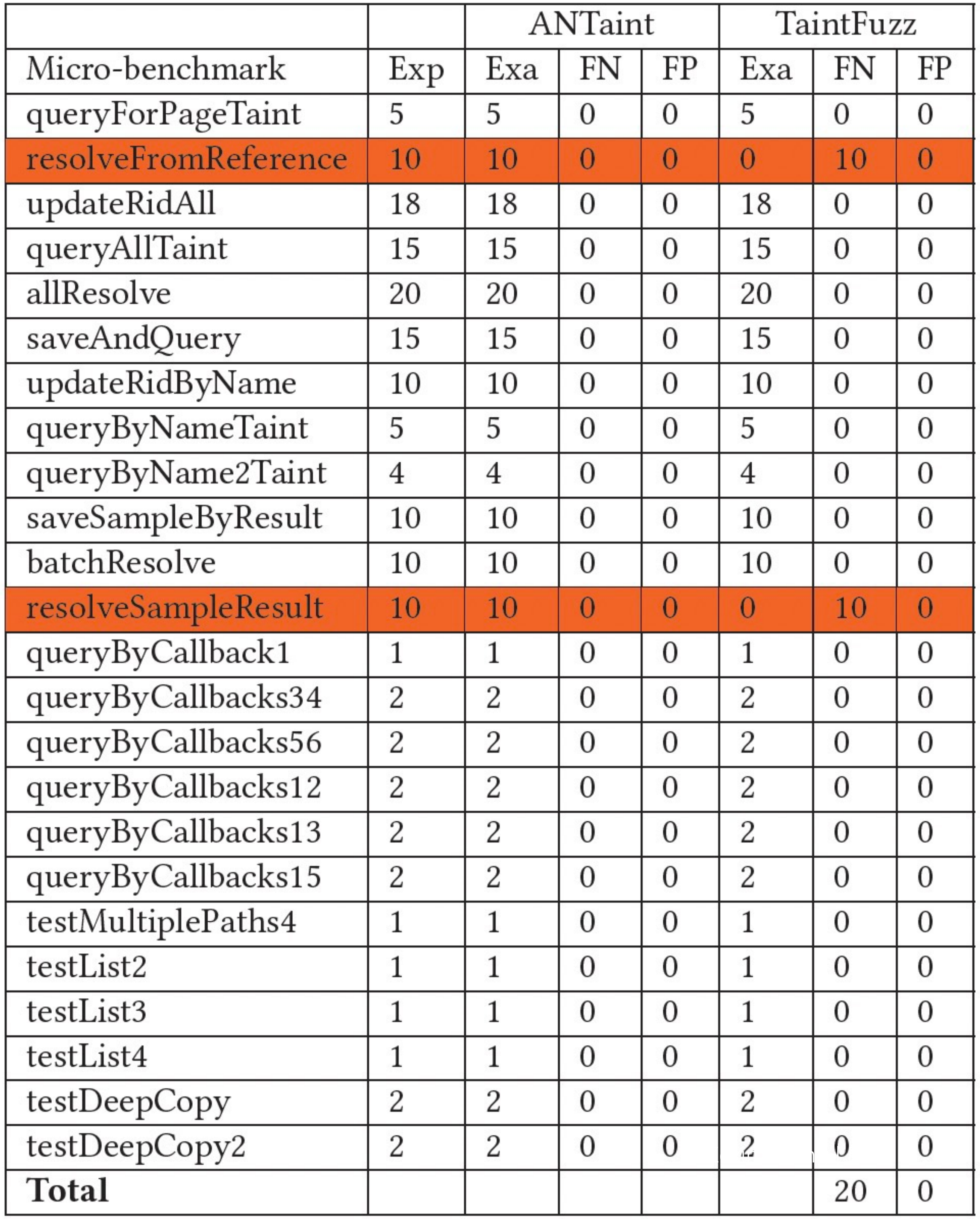
		ANTaint			TaintFuzz		
Micro-benchmark	Exp	Exa	FN	FP	Exa	FN	FP
queryForPageTaint	5	5	0	0	5	0	0
resolveFromReference	10	10	0	0	0	10	0
updateRidAll	18	18	0	0	18	0	0
queryAllTaint	15	15	0	0	15	0	0
allResolve	20	20	0	0	20	0	0
saveAndQuery	15	15	0	0	15	0	0
updateRidByName	10	10	0	0	10	0	0
queryByNameTaint	5	5	0	0	5	0	0
queryByName2Taint	4	4	0	0	4	0	0
saveSampleByResult	10	10	0	0	10	0	0
batchResolve	10	10	0	0	10	0	0
resolveSampleResult	10	10	0	0	0	10	0
queryByCallback1	1	1	0	0	1	0	0
queryByCallbacks34	2	2	0	0	2	0	0
queryByCallbacks56	2	2	0	0	2	0	0
queryByCallbacks12	2	2	0	0	2	0	0
queryByCallbacks13	2	2	0	0	2	0	0
queryByCallbacks15	2	2	0	0	2	0	0
testMultiplePaths4	1	1	0	0	1	0	0
testList2	1	1	0	0	1	0	0
testList3	1	1	0	0	1	0	0
testList4	1	1	0	0	1	0	0
testDeepCopy	2	2	0	0	2	0	0
testDeepCopy2	2	2	0	0	2	0	0
Total						20	0

Evaluation: Performance

RQ1: What is the performance of our approach when it is applied on real world industrial cases ?

- Evaluate on the Open-source Micro-benchmark
  - Pass 93% (25 out of 27 cases) of the test scenarios
  - With the precision of 100%

Two cases are failed to find any taint propagation paths because of using a super-class object as a source for the analysis



		ANTaint			TaintFuzz		
Micro-benchmark	Exp	Exa	FN	FP	Exa	FN	FP
queryForPageTaint	5	5	0	0	5	0	0
resolveFromReference	10	10	0	0	0	10	0
updateRidAll	18	18	0	0	18	0	0
queryAllTaint	15	15	0	0	15	0	0
allResolve	20	20	0	0	20	0	0
saveAndQuery	15	15	0	0	15	0	0
updateRidByName	10	10	0	0	10	0	0
queryByNameTaint	5	5	0	0	5	0	0
queryByName2Taint	4	4	0	0	4	0	0
saveSampleByResult	10	10	0	0	10	0	0
batchResolve	10	10	0	0	10	0	0
resolveSampleResult	10	10	0	0	0	10	0
queryByCallback1	1	1	0	0	1	0	0
queryByCallbacks34	2	2	0	0	2	0	0
queryByCallbacks56	2	2	0	0	2	0	0
queryByCallbacks12	2	2	0	0	2	0	0
queryByCallbacks13	2	2	0	0	2	0	0
queryByCallbacks15	2	2	0	0	2	0	0
testMultiplePaths4	1	1	0	0	1	0	0
testList2	1	1	0	0	1	0	0
testList3	1	1	0	0	1	0	0
testList4	1	1	0	0	1	0	0
testDeepCopy	2	2	0	0	2	0	0
testDeepCopy2	2	2	0	0	2	0	0
Total						20	0

Evaluation: Precision & Recall

RQ2: How well does our approach in terms of precision and recall for sensitive data tracing on industrial microservices applications?

- Production benchmark: more than 500 microservices applications

Metric Criteria

- Ture/Correct Path **(TP)** : Correct taint paths that are identified

Metric Criteria

- Ture/Correct Path **(TP)** : Correct taint paths that are identified

- False-Negative Path **(FN)** : Correct taint paths that are not identified

Metric Criteria

- Ture/Correct Path **(TP)** : Correct taint paths that are identified
- False-Negative Path **(FN)** : Correct taint paths that are not identified
- False-Positive Path **(FP)** : Incorrect taint paths that are identified

Metric Criteria

- Ture/Correct Path **(TP)** : Correct taint paths that are identified
- False-Negative Path **(FN)** : Correct taint paths that are not identified
- False-Positive Path **(FP)** : Incorrect taint paths that are identified
- Recall of our approach: $Recall\ =\ \frac{TP}{TP+FN}$

Metric Criteria

- Ture/Correct Path **(TP)** : Correct taint paths that are identified
- False-Negative Path **(FN)** : Correct taint paths that are not identified
- False-Positive Path **(FP)** : Incorrect taint paths that are identified
- Recall of our approach: $Recall\ = \ \frac{TP}{TP+FN}$
- Precision **(P)** of our approach: $P\ = \frac{TP}{TP\ +\ FP}$

Metric Criteria

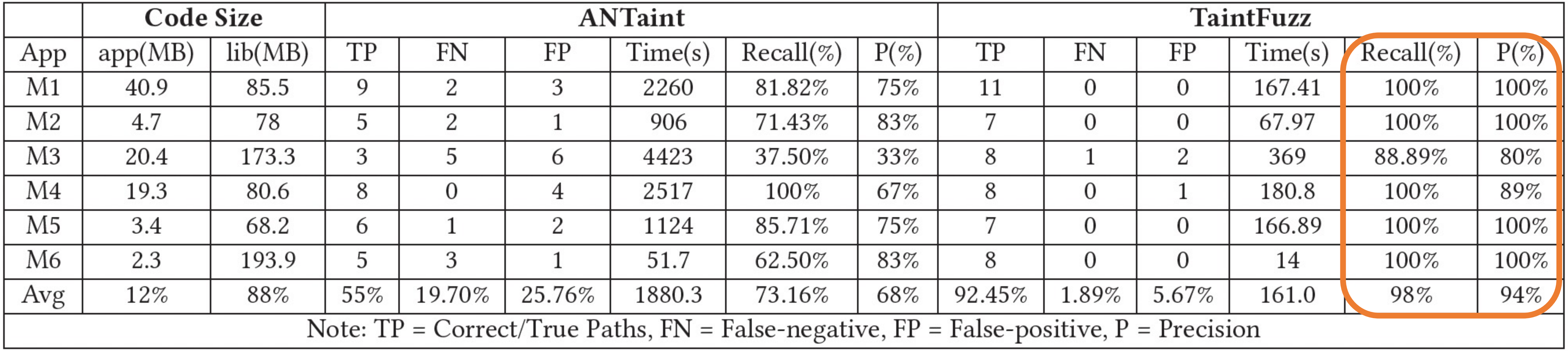
- Ture/Correct Path **(TP)** : Correct taint paths that are identified
- False-Negative Path **(FN)** : Correct taint paths that are not identified
- False-Positive Path **(FP)** : Incorrect taint paths that are identified
- Recall **(R)** of our approach: $R = \frac{TP}{TP+FN}$
- Precision **(P)** of our approach: $P = \frac{TP}{TP + FP}$
- Overall time consumption

Evaluation: Precision & Recall

RQ2: How well does our approach in terms of precision and recall for sensitive data tracing on industrial microservices applications?

- Production benchmark: more than 500 microservices applications

Average Precision of 94% and Recall of 98%



	Code Size		ANTaint						TaintFuzz					
App	app(MB)	lib(MB)	TP	FN	FP	Time(s)	Recall(%)	P(%)	TP	FN	FP	Time(s)	Recall(%)	P(%)
M1	40.9	85.5	9	2	3	2260	81.82%	75%	11	0	0	167.41	100%	100%
M2	4.7	78	5	2	1	906	71.43%	83%	7	0	0	67.97	100%	100%
M3	20.4	173.3	3	5	6	4423	37.50%	33%	8	1	2	369	88.89%	80%
M4	19.3	80.6	8	0	4	2517	100%	67%	8	0	1	180.8	100%	89%
M5	3.4	68.2	6	1	2	1124	85.71%	75%	7	0	0	166.89	100%	100%
M6	2.3	193.9	5	3	1	51.7	62.50%	83%	8	0	0	14	100%	100%
Avg	12%	88%	55%	19.70%	25.76%	1880.3	73.16%	68%	92.45%	1.89%	5.67%	161.0	98%	94%

Note: TP = Correct/True Paths, FN = False-negative, FP = False-positive, P = Precision

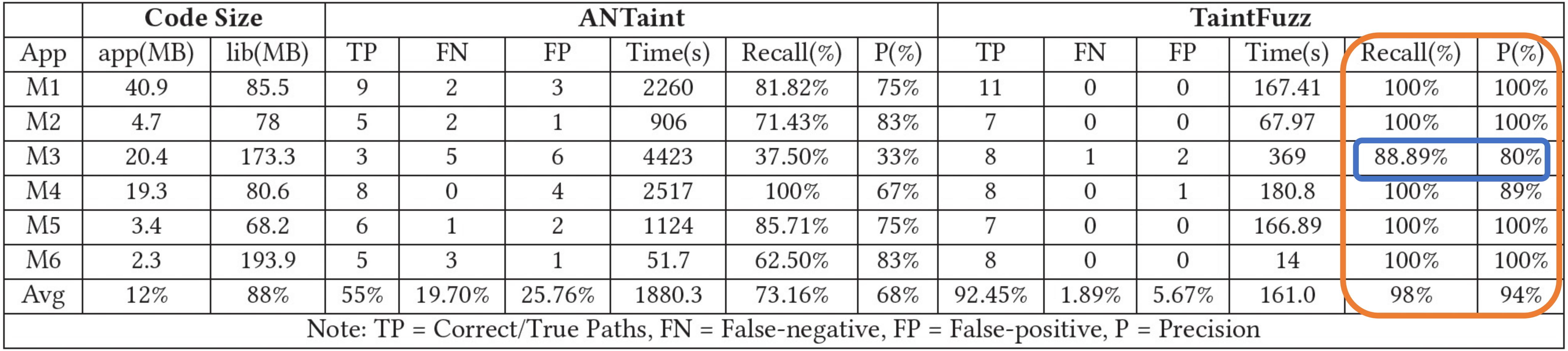
Evaluation: Precision & Recall

RQ2: How well does our approach in terms of precision and recall for sensitive data tracing on industrial microservices applications?

- Production benchmark: more than 500 microservices applications

Average Precision of 94% and **Recall of 98%**

Worst Precision of 80% and **Recall of 88.98%**



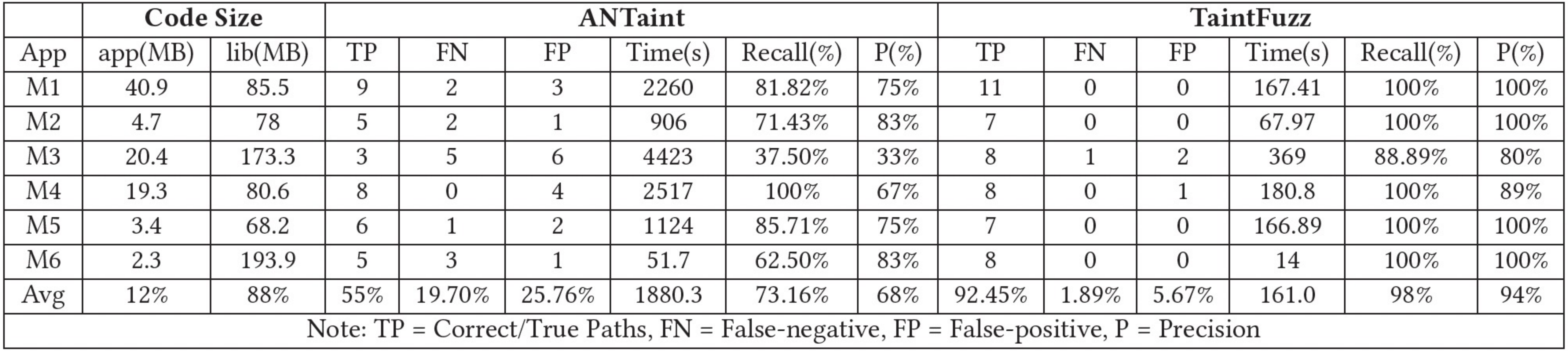
	Code Size		ANTaint						TaintFuzz					
App	app(MB)	lib(MB)	TP	FN	FP	Time(s)	Recall(%)	P(%)	TP	FN	FP	Time(s)	Recall(%)	P(%)
M1	40.9	85.5	9	2	3	2260	81.82%	75%	11	0	0	167.41	100%	100%
M2	4.7	78	5	2	1	906	71.43%	83%	7	0	0	67.97	100%	100%
M3	20.4	173.3	3	5	6	4423	37.50%	33%	8	1	2	369	88.89%	80%
M4	19.3	80.6	8	0	4	2517	100%	67%	8	0	1	180.8	100%	89%
M5	3.4	68.2	6	1	2	1124	85.71%	75%	7	0	0	166.89	100%	100%
M6	2.3	193.9	5	3	1	51.7	62.50%	83%	8	0	0	14	100%	100%
Avg	12%	88%	55%	19.70%	25.76%	1880.3	73.16%	68%	92.45%	1.89%	5.67%	161.0	98%	94%

Note: TP = Correct/True Paths, FN = False-negative, FP = False-positive, P = Precision

Evaluation: Precision & Recall

RQ2: How well does our approach in terms of precision and recall for sensitive data tracing on industrial microservices applications?

- Production benchmark: more than 500 microservices applications



	Code Size		ANTaint						TaintFuzz					
App	app(MB)	lib(MB)	TP	FN	FP	Time(s)	Recall(%)	P(%)	TP	FN	FP	Time(s)	Recall(%)	P(%)
M1	40.9	85.5	9	2	3	2260	81.82%	75%	11	0	0	167.41	100%	100%
M2	4.7	78	5	2	1	906	71.43%	83%	7	0	0	67.97	100%	100%
M3	20.4	173.3	3	5	6	4423	37.50%	33%	8	1	2	369	88.89%	80%
M4	19.3	80.6	8	0	4	2517	100%	67%	8	0	1	180.8	100%	89%
M5	3.4	68.2	6	1	2	1124	85.71%	75%	7	0	0	166.89	100%	100%
M6	2.3	193.9	5	3	1	51.7	62.50%	83%	8	0	0	14	100%	100%
Avg	12%	88%	55%	19.70%	25.76%	1880.3	73.16%	68%	92.45%	1.89%	5.67%	161.0	98%	94%

Note: TP = Correct/True Paths, FN = False-negative, FP = False-positive, P = Precision

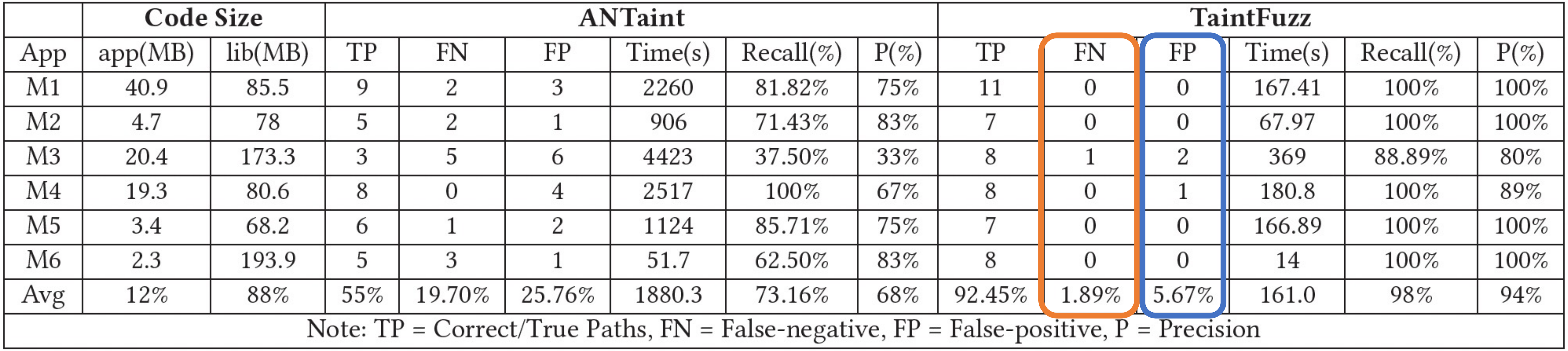
Evaluation: Precision & Recall

RQ2: How well does our approach in terms of precision and recall for sensitive data tracing on industrial microservices applications?

- Production benchmark: more than 500 microservices applications

1 FN Results: caused by the **usage of a vague object** as the taint source

3 FP Results: caused by the propagation between **container with a non-constant key**

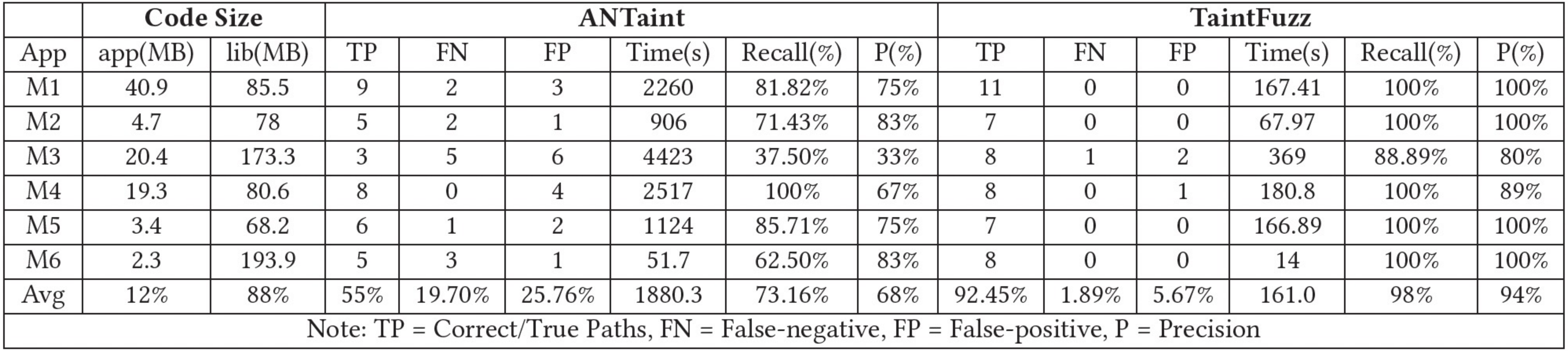


	Code Size		**ANTaint**						**TaintFuzz**					
App	app(MB)	lib(MB)	TP	FN	FP	Time(s)	Recall(%)	P(%)	TP	FN	FP	Time(s)	Recall(%)	P(%)
M1	40.9	85.5	9	2	3	2260	81.82%	75%	11	0	0	167.41	100%	100%
M2	4.7	78	5	2	1	906	71.43%	83%	7	0	0	67.97	100%	100%
M3	20.4	173.3	3	5	6	4423	37.50%	33%	8	1	2	369	88.89%	80%
M4	19.3	80.6	8	0	4	2517	100%	67%	8	0	1	180.8	100%	89%
M5	3.4	68.2	6	1	2	1124	85.71%	75%	7	0	0	166.89	100%	100%
M6	2.3	193.9	5	3	1	51.7	62.50%	83%	8	0	0	14	100%	100%
Avg	12%	88%	55%	19.70%	25.76%	1880.3	73.16%	68%	92.45%	1.89%	5.67%	161.0	98%	94%

Note: TP = Correct/True Paths, FN = False-negative, FP = False-positive, P = Precision

Evaluation: Comparison with ANTaint

RQ3: How TaintFuzz is compared to existing tools, such as ANTaint?



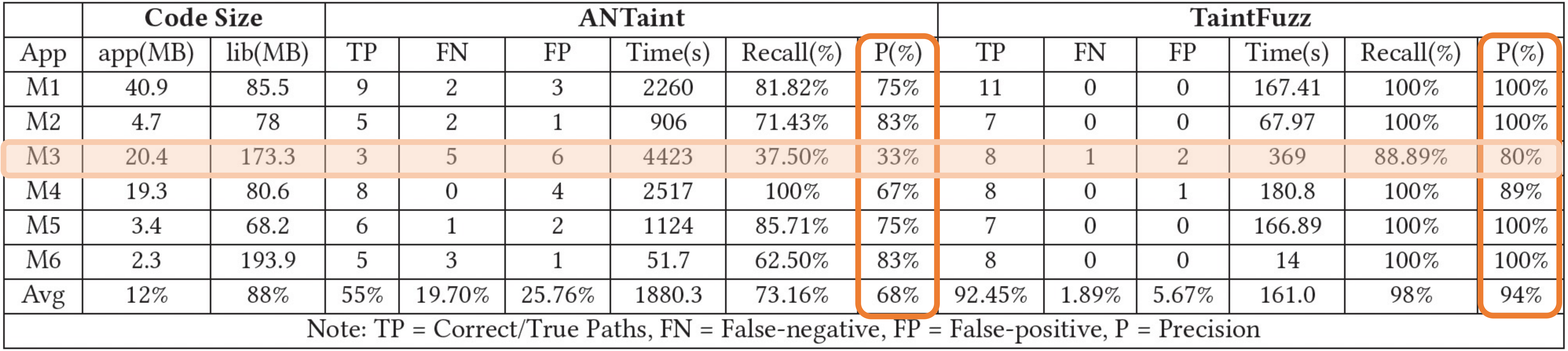
	Code Size		ANTaint						TaintFuzz					
App	app(MB)	lib(MB)	TP	FN	FP	Time(s)	Recall(%)	P(%)	TP	FN	FP	Time(s)	Recall(%)	P(%)
M1	40.9	85.5	9	2	3	2260	81.82%	75%	11	0	0	167.41	100%	100%
M2	4.7	78	5	2	1	906	71.43%	83%	7	0	0	67.97	100%	100%
M3	20.4	173.3	3	5	6	4423	37.50%	33%	8	1	2	369	88.89%	80%
M4	19.3	80.6	8	0	4	2517	100%	67%	8	0	1	180.8	100%	89%
M5	3.4	68.2	6	1	2	1124	85.71%	75%	7	0	0	166.89	100%	100%
M6	2.3	193.9	5	3	1	51.7	62.50%	83%	8	0	0	14	100%	100%
Avg	12%	88%	55%	19.70%	25.76%	1880.3	73.16%	68%	92.45%	1.89%	5.67%	161.0	98%	94%

Note: TP = Correct/True Paths, FN = False-negative, FP = False-positive, P = Precision

Evaluation: Comparison with ANTaint

RQ3: How TaintFuzz is compared to existing tools, such as ANTaint?

Precision: **68%**(ANTaint) *V.S.* **94%**(TaintFuzz) ➔ **25% increase**



	Code Size		**ANTaint**						**TaintFuzz**					
App	app(MB)	lib(MB)	TP	FN	FP	Time(s)	Recall(%)	P(%)	TP	FN	FP	Time(s)	Recall(%)	P(%)
M1	40.9	85.5	9	2	3	2260	81.82%	75%	11	0	0	167.41	100%	100%
M2	4.7	78	5	2	1	906	71.43%	83%	7	0	0	67.97	100%	100%
M3	20.4	173.3	3	5	6	4423	37.50%	33%	8	1	2	369	88.89%	80%
M4	19.3	80.6	8	0	4	2517	100%	67%	8	0	1	180.8	100%	89%
M5	3.4	68.2	6	1	2	1124	85.71%	75%	7	0	0	166.89	100%	100%
M6	2.3	193.9	5	3	1	51.7	62.50%	83%	8	0	0	14	100%	100%
Avg	12%	88%	55%	19.70%	25.76%	1880.3	73.16%	68%	92.45%	1.89%	5.67%	161.0	98%	94%

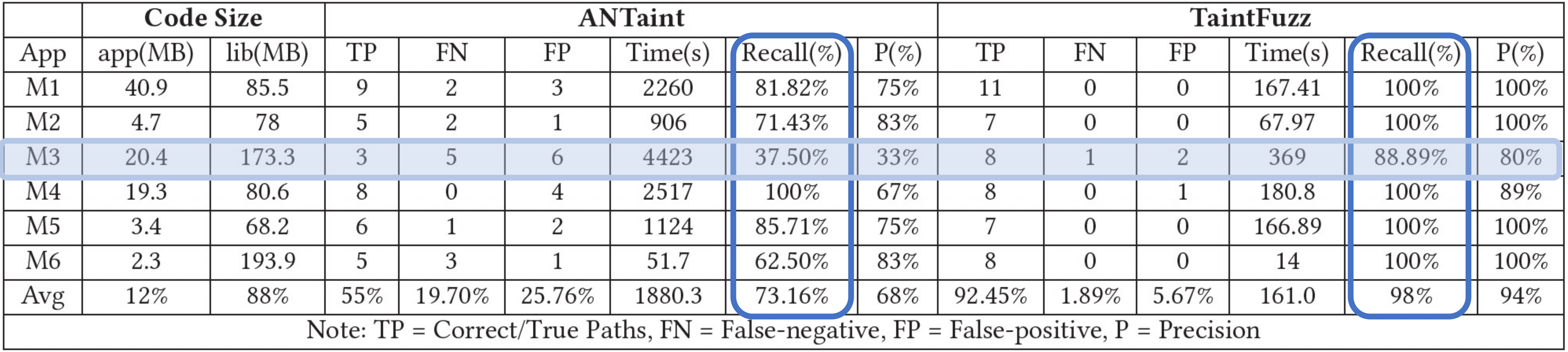
Note: TP = Correct/True Paths, FN = False-negative, FP = False-positive, P = Precision

Evaluation: Comparison with ANTaint

RQ3: How TaintFuzz is compared to existing tools, such as ANTaint?

Precision: **68%**(ANTaint) *V.S.* **94%**(TaintFuzz) ➔ **25% increase**

Recall: **73.16%**(ANTaint) *V.S.* **98%**(TaintFuzz) ➔ **24.84% increase**



	Code Size		ANTaint						TaintFuzz					
App	app(MB)	lib(MB)	TP	FN	FP	Time(s)	Recall(%)	P(%)	TP	FN	FP	Time(s)	Recall(%)	P(%)
M1	40.9	85.5	9	2	3	2260	81.82%	75%	11	0	0	167.41	100%	100%
M2	4.7	78	5	2	1	906	71.43%	83%	7	0	0	67.97	100%	100%
M3	20.4	173.3	3	5	6	4423	37.50%	33%	8	1	2	369	88.89%	80%
M4	19.3	80.6	8	0	4	2517	100%	67%	8	0	1	180.8	100%	89%
M5	3.4	68.2	6	1	2	1124	85.71%	75%	7	0	0	166.89	100%	100%
M6	2.3	193.9	5	3	1	51.7	62.50%	83%	8	0	0	14	100%	100%
Avg	12%	88%	55%	19.70%	25.76%	1880.3	73.16%	68%	92.45%	1.89%	5.67%	161.0	98%	94%

Note: TP = Correct/True Paths, FN = False-negative, FP = False-positive, P = Precision

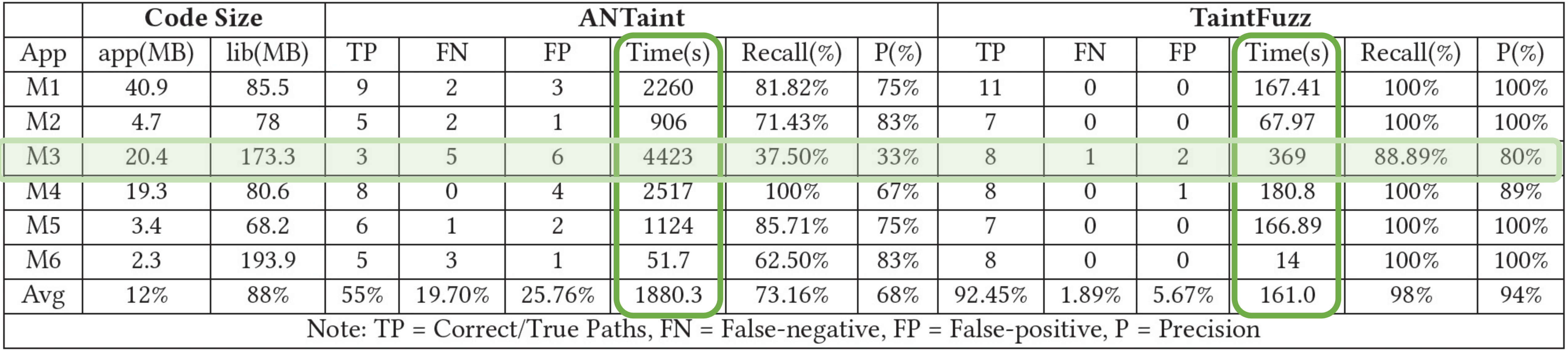
Evaluation: Comparison with ANTaint

RQ3: How TaintFuzz is compared to existing tools, such as ANTaint?

Precision: **68%**(ANTaint) *V.S.* **94%**(TaintFuzz) ➔ **25% increase**

Recall: **73.16%**(ANTaint) *V.S.* **98%**(TaintFuzz) ➔ **24.84% increase**

Time: **31mins**(ANTaint) *V.S.* **2.67mins**(TaintFuzz) ➔ **91% off & 10 times faster**



	Code Size		**ANTaint**						**TaintFuzz**					
App	app(MB)	lib(MB)	TP	FN	FP	Time(s)	Recall(%)	P(%)	TP	FN	FP	Time(s)	Recall(%)	P(%)
M1	40.9	85.5	9	2	3	2260	81.82%	75%	11	0	0	167.41	100%	100%
M2	4.7	78	5	2	1	906	71.43%	83%	7	0	0	67.97	100%	100%
M3	20.4	173.3	3	5	6	4423	37.50%	33%	8	1	2	369	88.89%	80%
M4	19.3	80.6	8	0	4	2517	100%	67%	8	0	1	180.8	100%	89%
M5	3.4	68.2	6	1	2	1124	85.71%	75%	7	0	0	166.89	100%	100%
M6	2.3	193.9	5	3	1	51.7	62.50%	83%	8	0	0	14	100%	100%
Avg	12%	88%	55%	19.70%	25.76%	1880.3	73.16%	68%	92.45%	1.89%	5.67%	161.0	98%	94%

Note: TP = Correct/True Paths, FN = False-negative, FP = False-positive, P = Precision

UTS

Conclusion

In Conclusion

- Features
  - ✓ A compositional taint analysis approach

In Conclusion

- Features
  - ✓ A compositional taint analysis approach
  - ✓ Used the function summary strategy for more scalable analysis

In Conclusion

- Features
  - ✓ A compositional taint analysis approach
  - ✓ Used the function summary strategy for scalable analysis
  - ✓ Used the field-based algorithm to provide more sound analysis

In Conclusion

- Features
  - ✓ A compositional taint analysis approach
  - ✓ Used the function summary strategy for scalable analysis
  - ✓ Used the field-based algorithm to provide more sound analysis
  - ✓ Used the white-box fuzzing to verify the massive amount of taint analysis results

In Conclusion

- Features
  - ✓ A compositional taint analysis approach
  - ✓ Used the function summary strategy for scalable analysis
  - ✓ Used the field-based algorithm to provide more sound analysis
  - ✓ Used the white-box fuzzing to verify the massive amount of taint analysis results
  - ✓ 25% higher than ANTaint in terms of precision and recall

In Conclusion

- Features
  - ✓ A compositional taint analysis approach
  - ✓ Used the function summary strategy for scalable analysis
  - ✓ Used the field-based algorithm to provide more sound analysis
  - ✓ Used the white-box fuzzing to verify the massive amount of taint analysis results
  - ✓ 25% higher than ANTaint in terms of precision and recall
  - ✓ 10 times faster than ANTaint

In Conclusion

- Features
  - ✓ A compositional taint analysis approach
  - ✓ Used the function summary strategy for scalable analysis
  - ✓ Used the field-based algorithm to provide more sound analysis
  - ✓ Used the white-box fuzzing to verify the massive amount of taint analysis results
  - ✓ 25% higher than ANTaint in terms of precision and recall
  - ✓ 10 times faster than ANTaint

In Conclusion

- Features
  - ✓ A compositional taint analysis approach
  - ✓ Used the function summary strategy for scalable analysis
  - ✓ Used the field-based algorithm to provide more sound analysis
  - ✓ Used the white-box fuzzing to verify the massive amount of taint analysis results
  - ✓ 25% higher than ANTaint in terms of precision and recall
  - ✓ 10 times faster than ANTaint
- Limitations

In Conclusion

- Features
  - ✓ A compositional taint analysis approach
  - ✓ Used the function summary strategy for scalable analysis
  - ✓ Used the field-based algorithm to provide more sound analysis
  - ✓ Used the white-box fuzzing to verify the massive amount of taint analysis results
  - ✓ 25% higher than ANTaint in terms of precision and recall
  - ✓ 10 times faster than ANTaint

- Limitations
  - ✓ Unable to tracking taint propagation with non-constant key data in container

In Conclusion

- Features
  - ✓ A compositional taint analysis approach
  - ✓ Used the function summary strategy for scalable analysis
  - ✓ Used the field-based algorithm to provide more sound analysis
  - ✓ Used the white-box fuzzing to verify the massive amount of taint analysis results
  - ✓ 25% higher than ANTaint in terms of precision and recall
  - ✓ 10 times faster than ANTaint
- Limitations
  - ✓ Unable to tracking taint propagation with non-constant key data in container
  - ✓ Unable to tracking taint propagation of a source of an superclass object

UTS

Questions ?