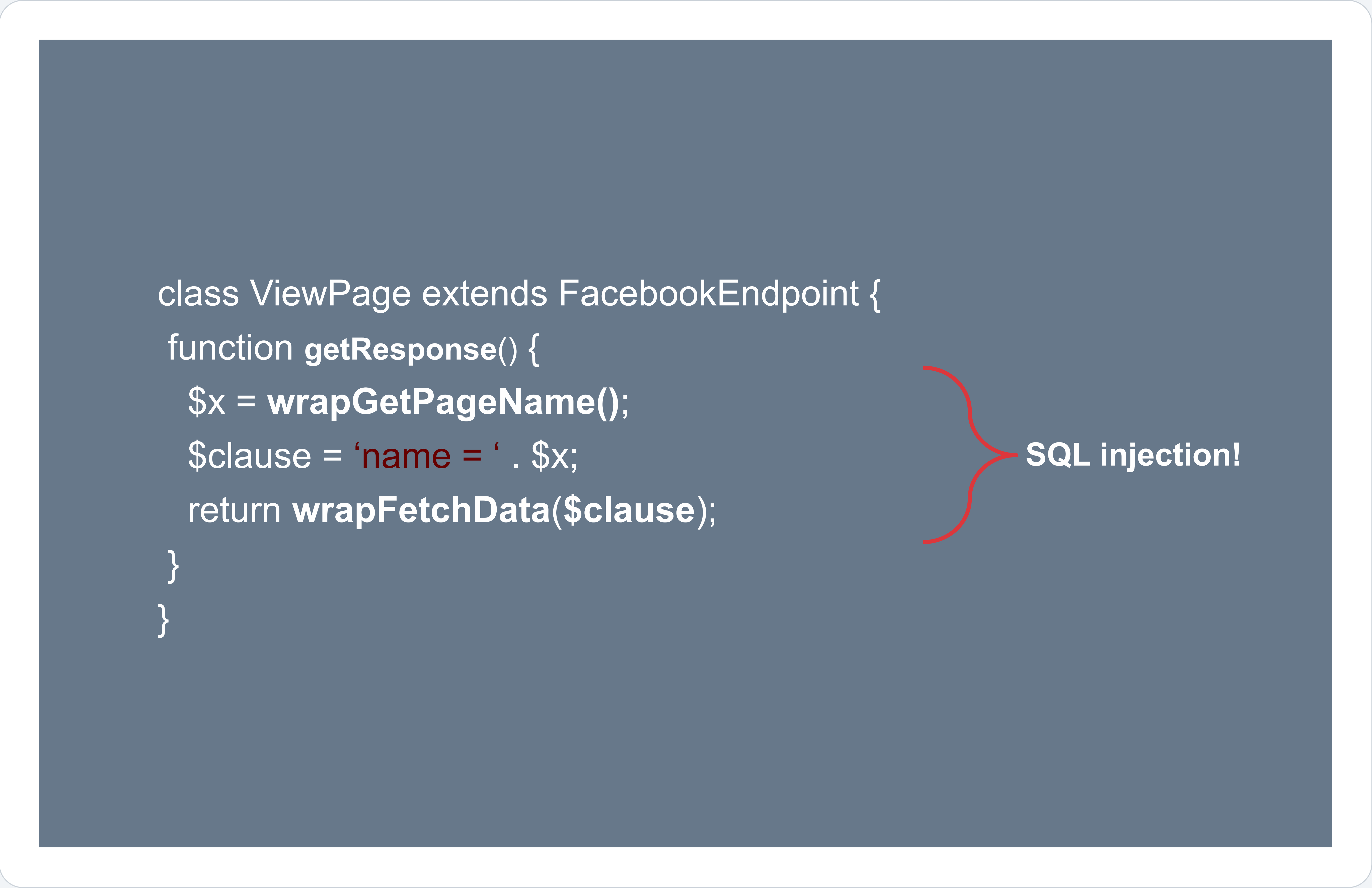
Do you speak my language?

Make Static Analysis Engines Understand Each Other

Ibrahim Mohamed
Security Engineer

FACEBOOK

```
class ViewPage extends FacebookEndpoint {
 function getResponse() {
   $x = $_REQUEST[‘page_name’];
   $qry = “select * from pages where
           name = ““ . $x . “““;
   return mysql_query($qry);
 }
}
```



```
class ViewPage extends FacebookEndpoint {
 function getResponse() {
   $x = wrapGetPageName();
   $clause = 'name = ' . $x;
   return wrapFetchData($clause);
 }
}
```

SQL injection!

SQL injection!

```
class ViewPage extends FacebookEndpoint {
    function getResponse() {
      $x = wrapGetPageName();
      $clause = 'name = ' . $x;
      ...
      $client = new PageServiceClient(...);
      $client->fetchData($clause);
    }
}
```

```
class PageServiceHandler:
   def __init__(self):
      self.db = MySQLdb.connect(...)

   def fetchData(self, clause):
      cursor = db.cursor()
      query = "select * from pages" + clause
      return cursor.execute(query).fetchone()

if __name__ == '__main__':
   server = TSimpleServer(PageServiceHandler)
   print('Starting the server...')
   server.serve()
```

Ibrahim Mohamed

>= 2016 - now: Security engineer @ Facebook

< 2016: Security consultant

Agenda

Motivation

Single-repo analysis

Cross-repo analysis

Example flows

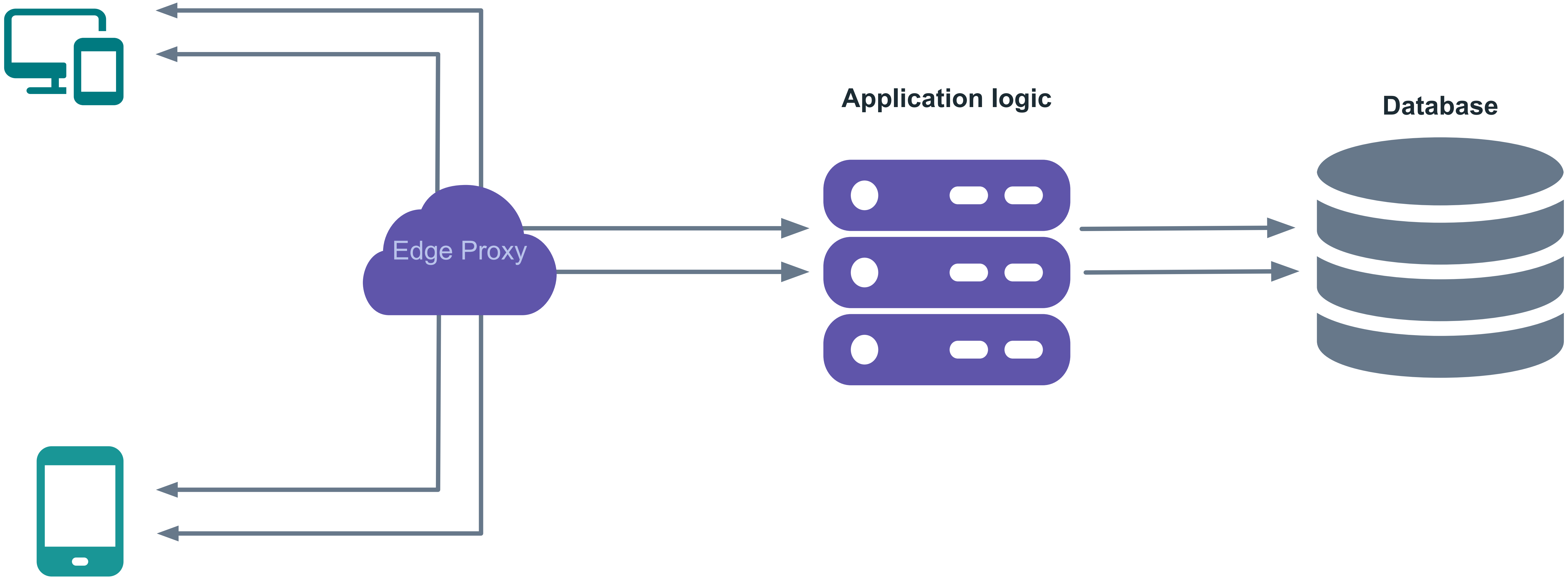
Looking forward

SQL injection!

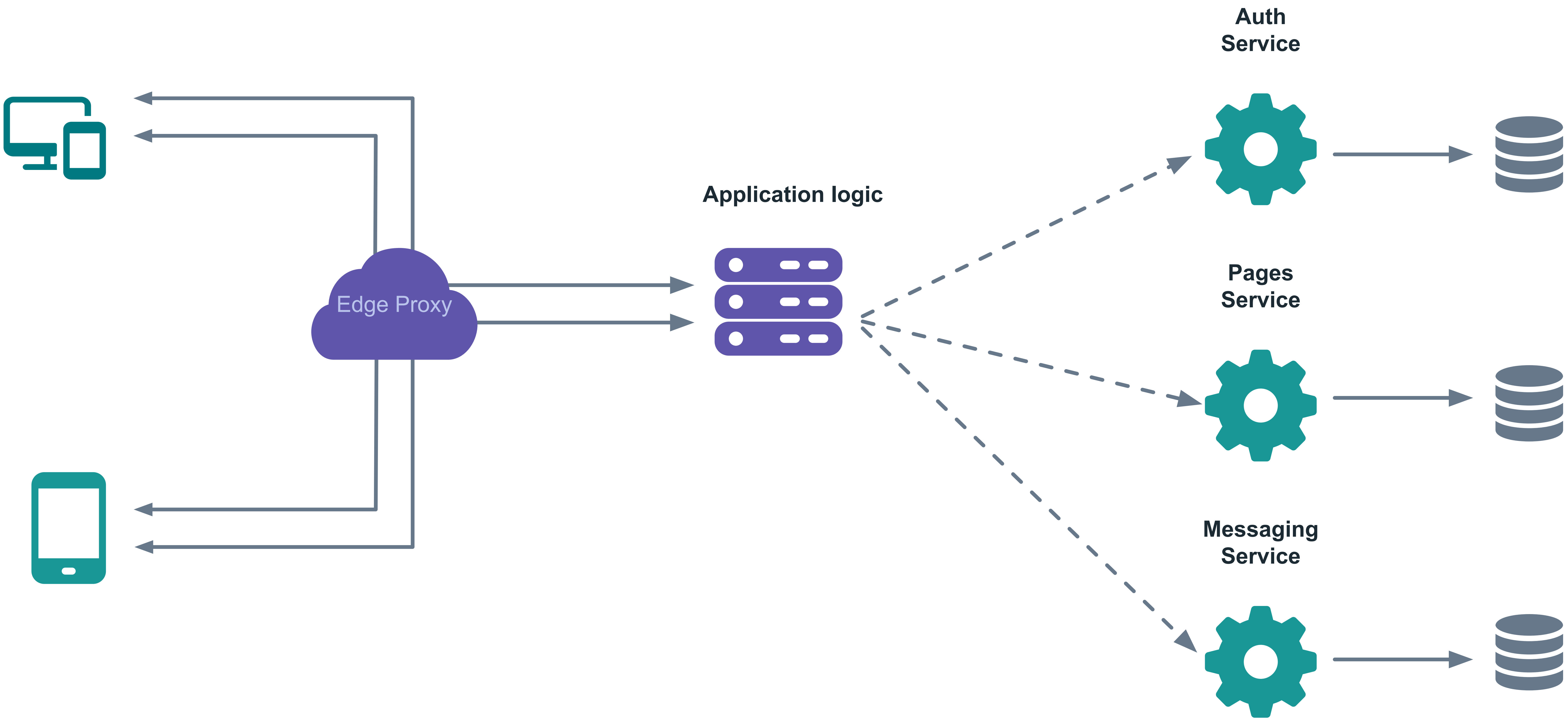
```
class ViewPage extends FacebookEndpoint {
    function getResponse() {
      $x = wrapGetPageName();
      $clause = ‘name = ‘ . $x;
      ...
      $client = new PageServiceClient(...);
      $client->fetchData($clause);
    }
}
```

```
class PageServiceHandler:
    def __init__(self):
        self.db = MySQLdb.connect(...)

    def fetchData(self, clause):
        cursor = db.cursor()
        query = “select * from pages” + clause
        return cursor.execute(query).fetchone()

if __name__ == '__main__':
    server = TSimpleServer(PageServiceHandler)
    print('Starting the server...')
    server.serve()
```

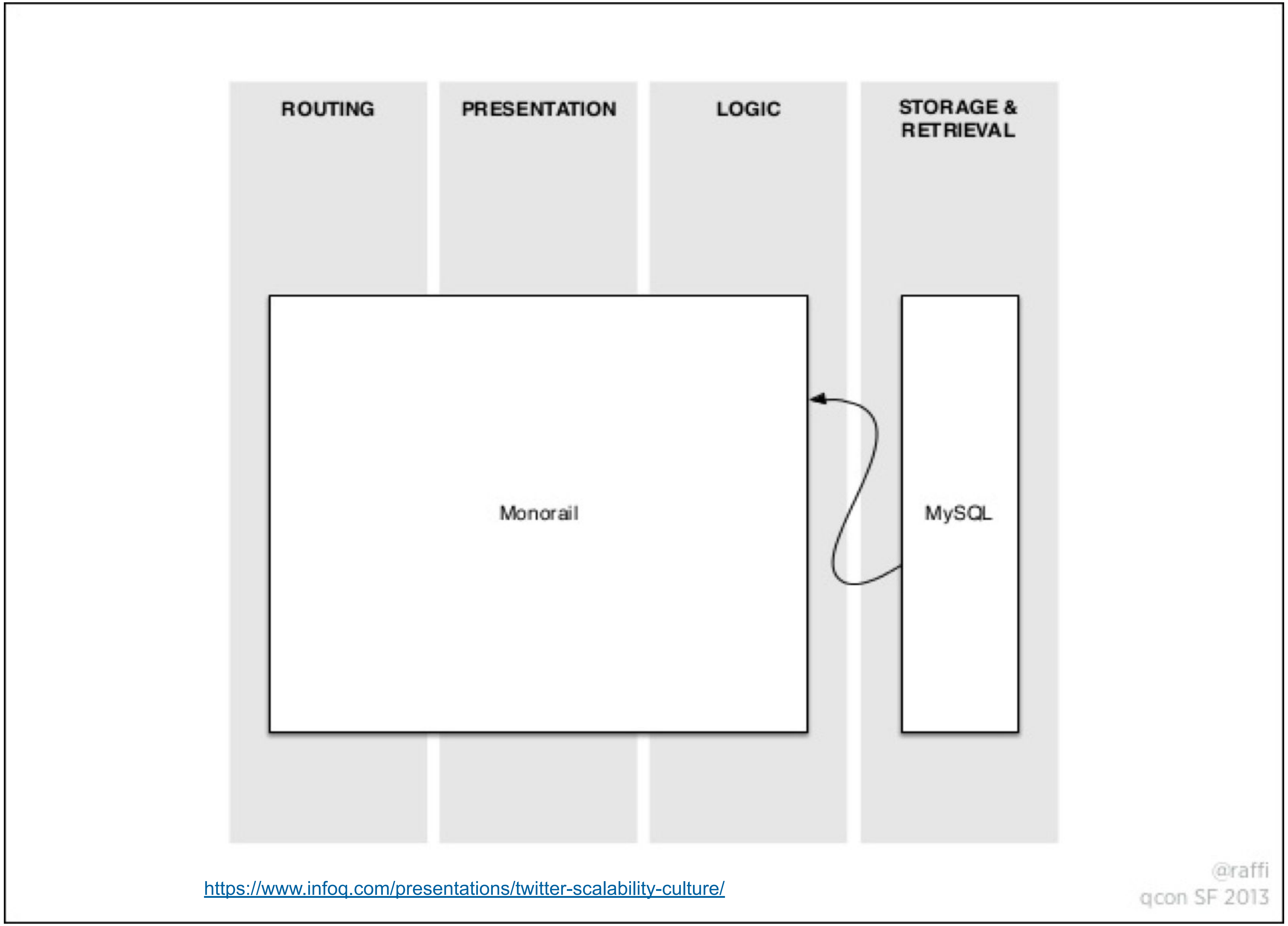
Motivation: Service-oriented architecture



Motivation: Service-oriented architecture



Motivation: Service-oriented architecture

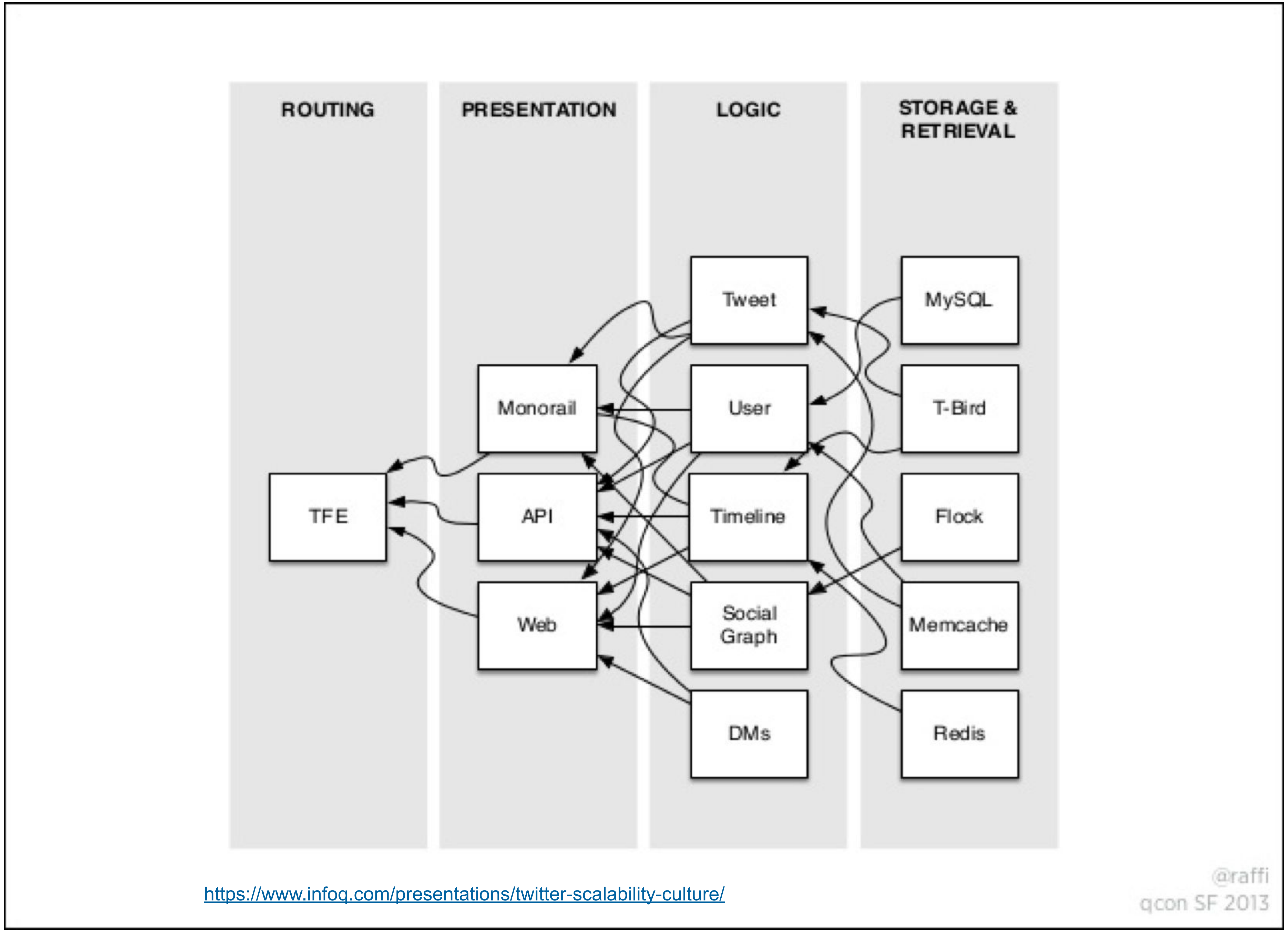


@raffi
qcon SF 2013

https://www.infoq.com/presentations/twitter-scalability-culture/

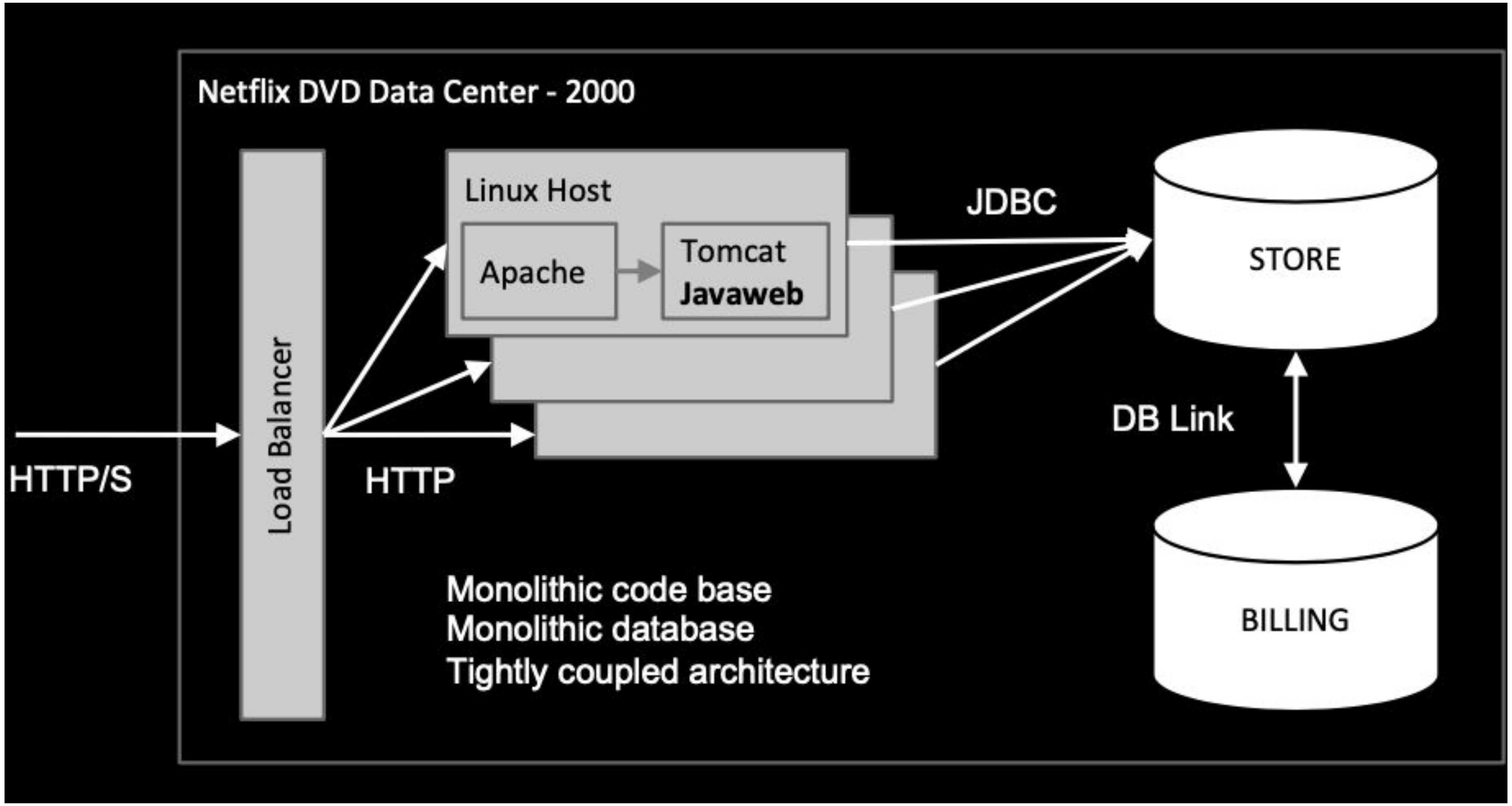
TWITTER, TWEET, RETWEET and the Twitter Bird logo are trademarks of Twitter Inc. or its affiliates.

Motivation: Service-oriented architecture

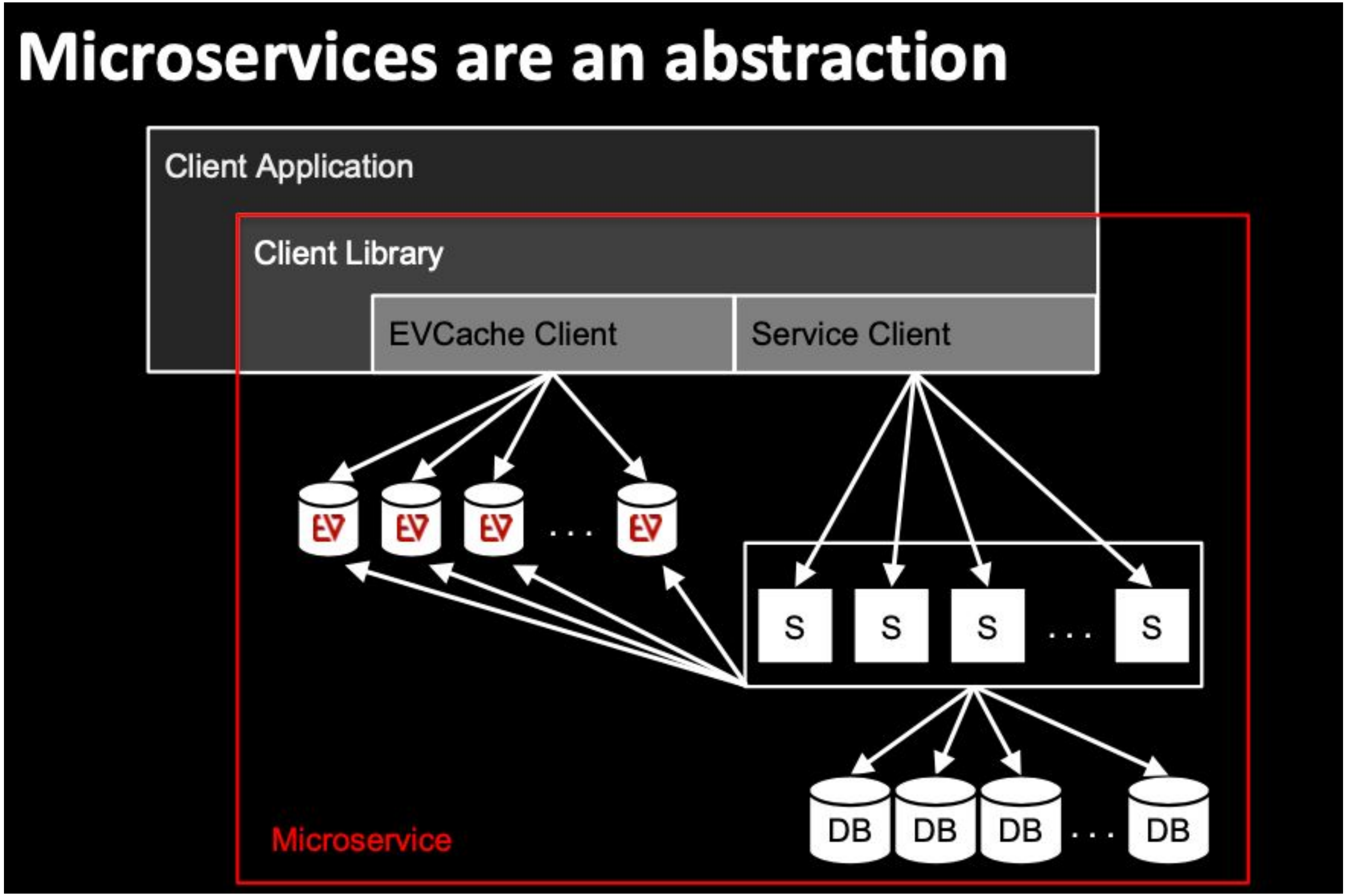


https://www.infoq.com/presentations/twitter-scalability-culture/

@raffi
qcon SF 2013



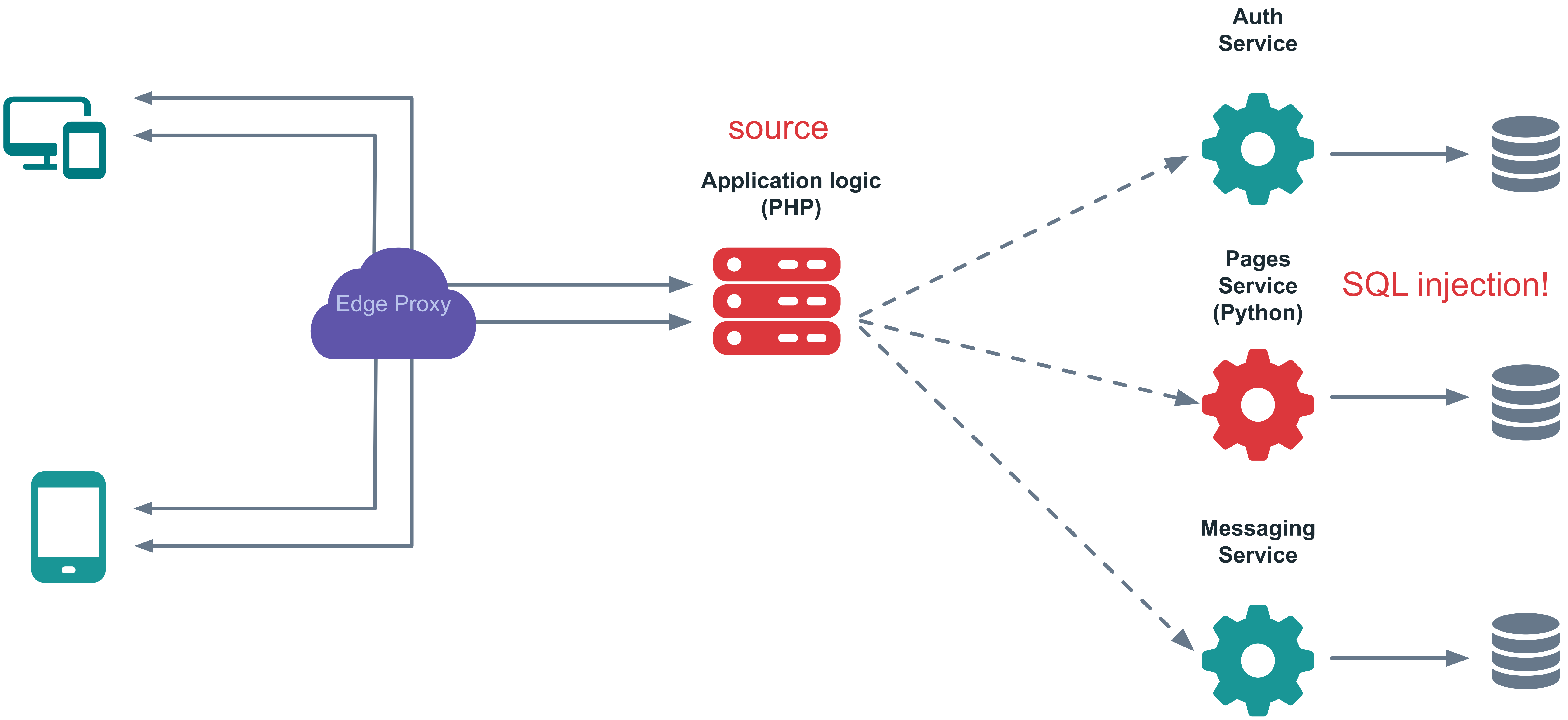
https://www.infoq.com/presentations/netflix-chaos-microservices/



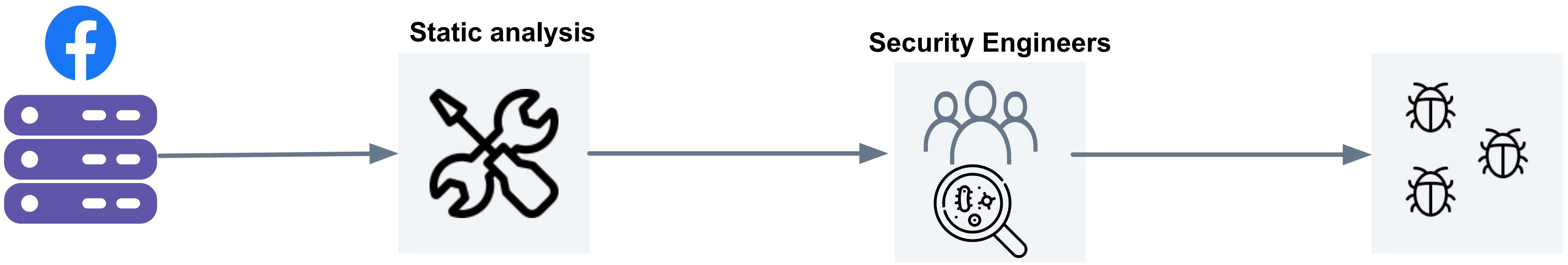
Microservices are an abstraction

https://www.infoq.com/presentations/netflix-chaos-microservices/

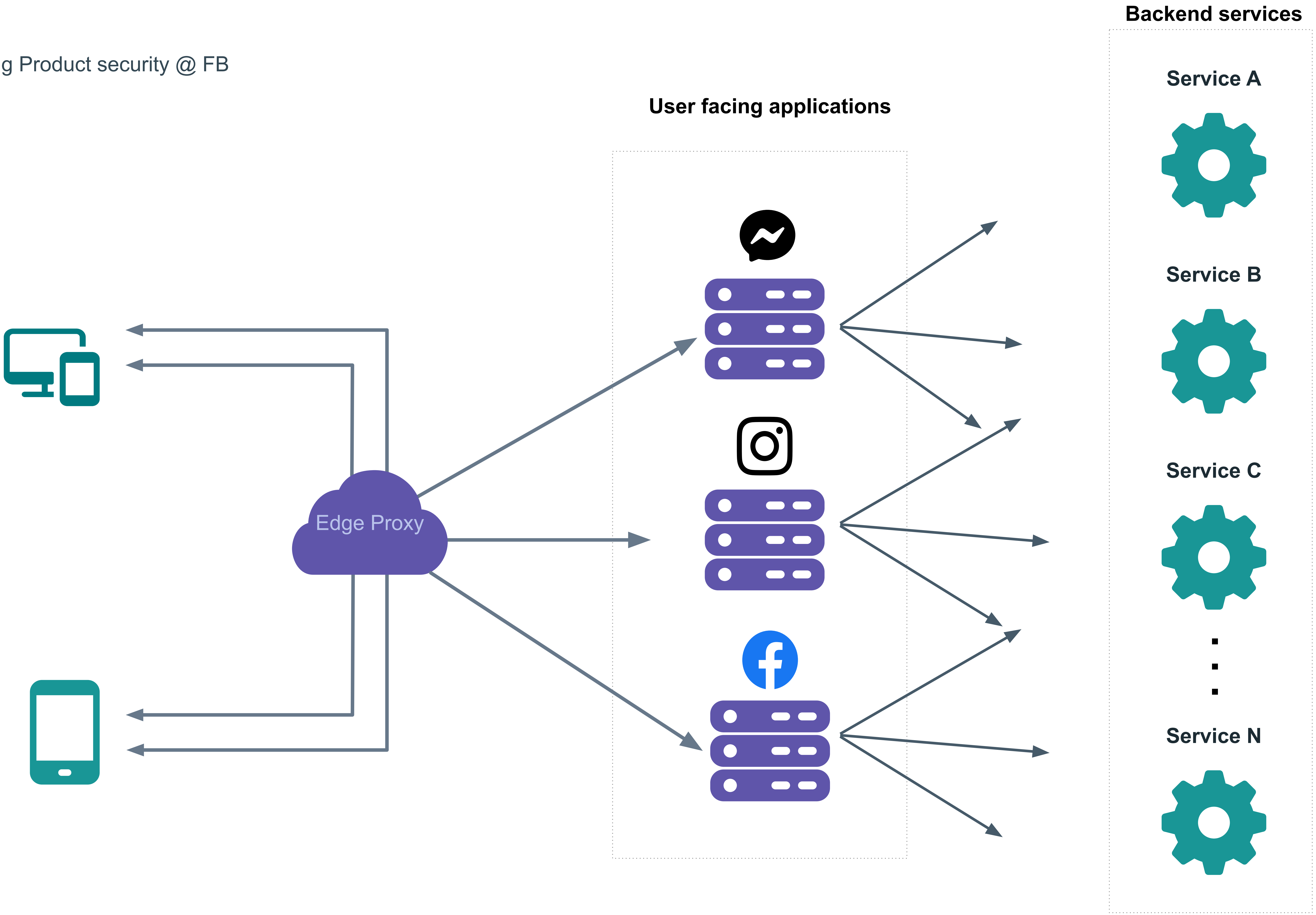
Motivation: Service-oriented architecture



Scaling Product security @ FB



Scaling Product security @ FB



Static Analysis @ Facebook

Taint-flow analysis

```
class ViewPage extends FacebookEndpoint {
 function getResponse() {
   $x = wrapGetPageName();
   $clause = ‘name = ‘ . $x;
   return wrapFetchData($clause);
 }
}
```

Taint-flow analysis

```
function wrapGetPageName(): string {
  return getPageName();
}
```

1

```
class ViewPage extends FacebookEndpoint {
  function getResponse(): void {
    $x = wrapGetPageName();
    $clause = 'name = ' . $x;
    wrapFetchData($clause);
  }
}
```

Taint-flow analysis

```
function getPageName(): string {
  return request()['page_name'];
}
```

2

```
function wrapGetPageName(): string {
  return getPageName();
}
```

1

```
class ViewPage extends FacebookEndpoint {
 function getResponse(): void {
   $x = wrapGetPageName();
   $clause = 'name = ' . $x;
   wrapFetchData($clause);
 }
}
```

Taint-flow analysis

```
request(): {
  return $_REQUEST;
}
```

3

```
function getPageName(): string {
  return request()[‘page_name’];
}
```

2

```
function wrapGetPageName(): string {
  return getPageName();
}
```

1

```
class ViewPage extends FacebookEndpoint {
  function getResponse(): void {
    $x = wrapGetPageName();
    $clause = ‘name = ‘ . $x;
    wrapFetchData($clause);
  }
}
```

Taint-flow analysis

```
function wrapFetchData(
  string $clause,
): {
  return fetchData($clause);
}
```

1

```
class ViewPage extends FacebookEndpoint {
  function getResponse(): void {
    $x = wrapGetPageName();
    $clause = 'name = ' . $x;
    wrapFetchData($clause);
  }
}
```

Taint-flow analysis

```
function fetchData(
  string $clause,
): {
  return mysql_query(
    'select …' . $clause
  );
}
```

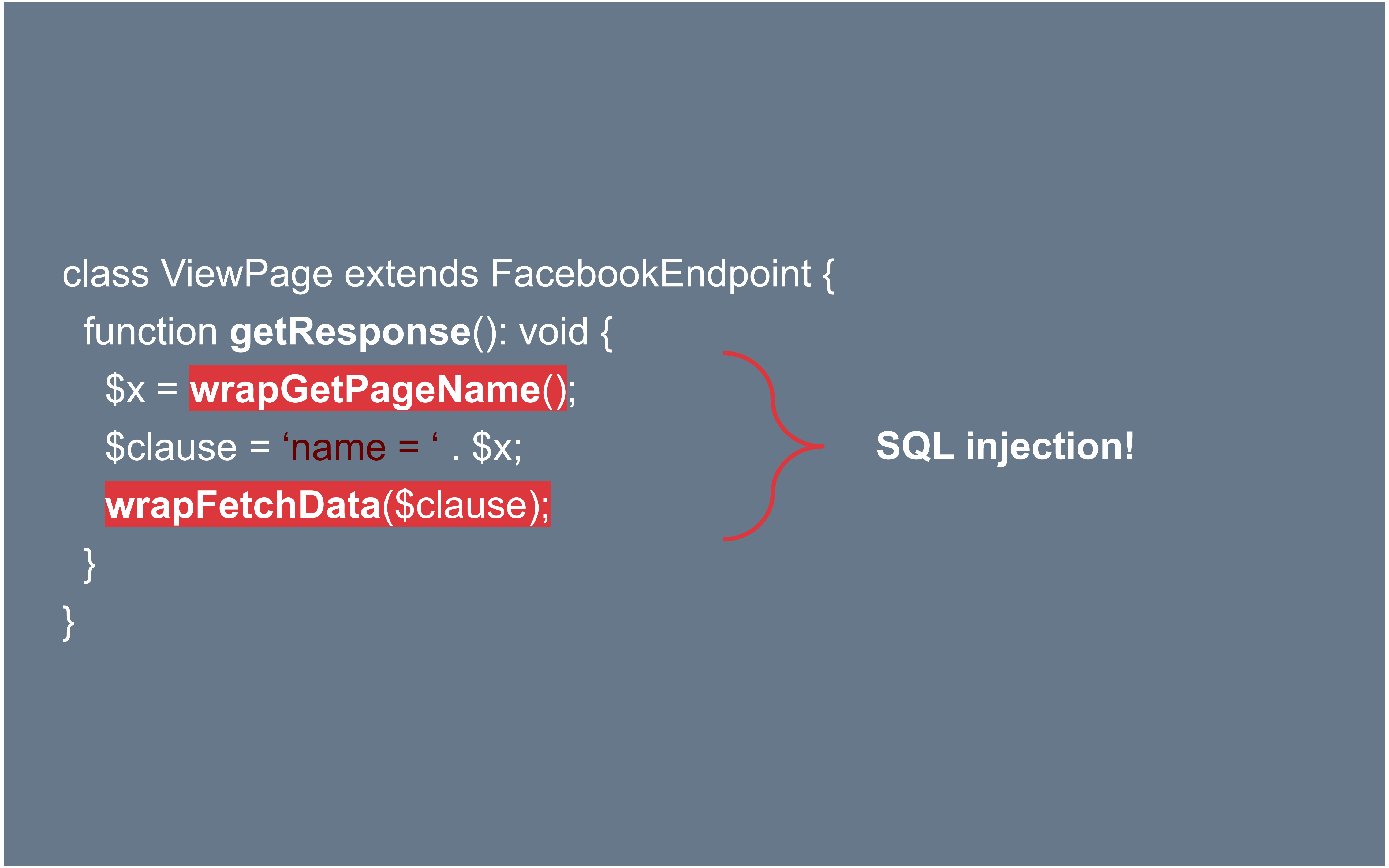
2

```
function wrapFetchData(
  string $clause,
): {
  return fetchData($clause);
}
```

1

```
class ViewPage extends FacebookEndpoint {
  function getResponse(): void {
    $x = wrapGetPageName();
    $clause = 'name = ' . $x;
    wrapFetchData($clause);
  }
}
```

Taint-flow analysis



```
class ViewPage extends FacebookEndpoint {
  function getResponse(): void {
    $x = wrapGetPageName();
    $clause = ‘name = ‘ . $x;
    wrapFetchData($clause);
  }
}
```

SQL injection!

Taint-flow analysis

```
function request() {
  return $_REQUEST;
}
 request():UserControlled source
```

3

```
class ViewPage extends FacebookEndpoint {
  function getResponse(): void {
                            PageName();
                            ne = ' . $x;
                            ta($clause);

}
```

sources: functions/methods that **return tainted data** which the static analysis tool **should track across the call graph**.

sources usually map to places where **untrusted input is returned**

Taint-flow analysis

3

```
function request() {
  return $_REQUEST;
}
 request():UserControlled source
```

2

```
function getPageName(): string {
  return request()['page_name'];
}
  getPageName():UserControlled source
```

```
class ViewPage extends FacebookEndpoint {
  function getResponse(): void {
    $x = wrapGetPageName();
    $clause = 'name = ' . $x;
    wrapFetchData($clause);
  }
}
```

Taint-flow analysis

3

```
function request() {
  return $_REQUEST;
}
 request():UserControlled source
```

2

```
function getPageName(): string {
  return request()['page_name'];
}
  getPageName():UserControlled source
```

1

```
function wrapGetPageName(): string {
  return getPageName();
}
 wrapGetPageName():UserControlled source
```

```
class ViewPage extends FacebookEndpoint {
  function getResponse(): void {
    $x = wrapGetPageName();
    $clause = 'name = ' . $x;
    wrapFetchData($clause);
  }
}
```

2

```
function fetchData(
  string $clause,
) {
  return mysql_query(
    'select ... ' . $clause
  );
}

fetchD
```

```
class ViewPage extends FacebookEndpoint {
  function getResponse(): void {
                         PageName();
                      ne = ' . $x;
                      ta($clause);

}
```

sinks: functions/methods that if **tainted data flows into**, we want to **create issues** and see the full flow

Taint-flow analysis

2

```
function fetchData(
  string $clause,
) {
  return mysql_query(
    'select …' . $clause,
  );
}
 fetchData($clause:SQLi sink)
```

1

```
function wrapFetchData(
  string $clause,
) {
  return fetchData($clause);
}
 wrapFetchData($clause:SQLi sink)
```

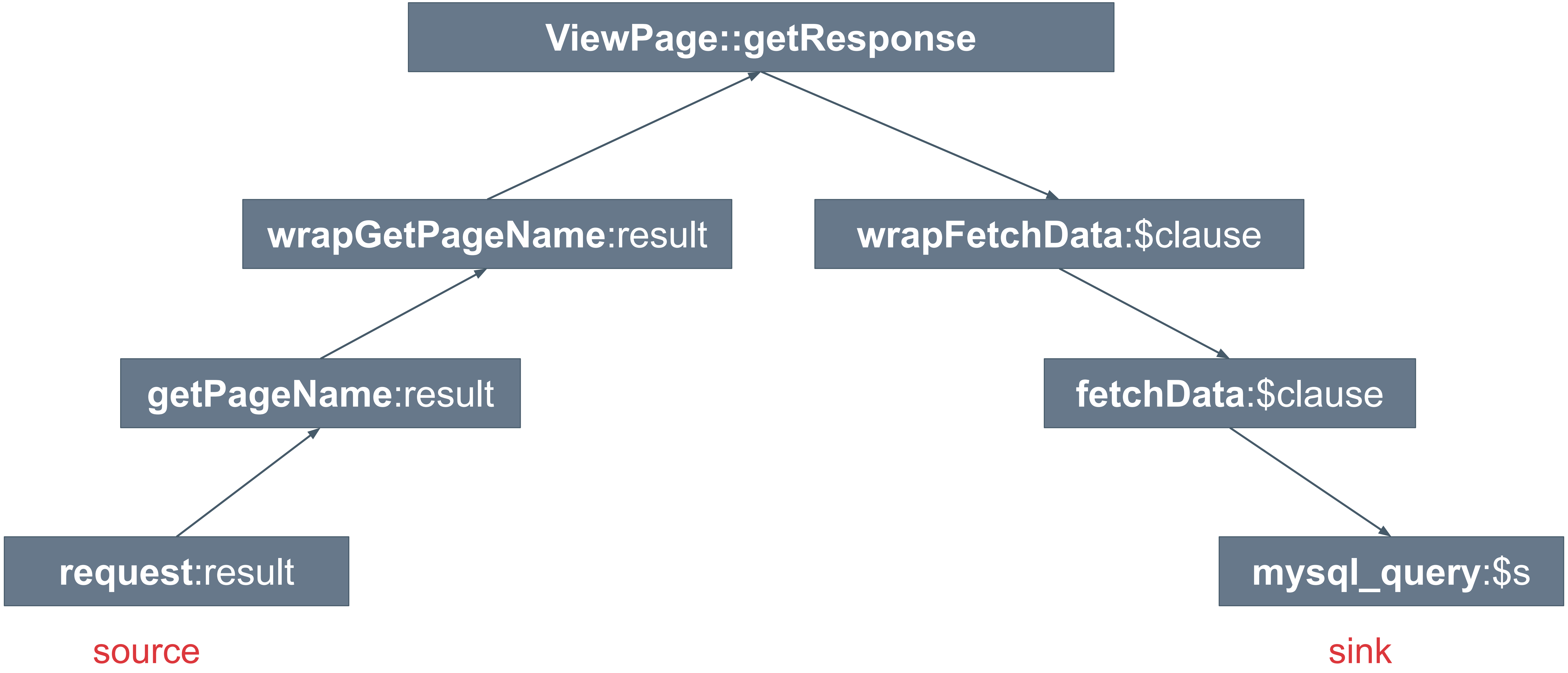
```
class ViewPage extends FacebookEndpoint {
  function getResponse(): void {
    $x = wrapGetPageName();
    $clause = 'name = ' . $x;
    wrapFetchData($clause);
  }
}
```

Taint-flow analysis

```
rule: {
  sources: [UserControlled],
  sinks: [SQLi],
  message: “SQL injection”
}
```

```
class ViewPage extends FacebookEndpoint {
  function getResponse(): void {
    $x = wrapGetPageName();
    $clause = ‘name = ‘ . $x;
    wrapFetchData($clause);
  }
}
```

Trace in our Example



Facebook Taint-flow analysis engines

Zoncolan

Taint-flow analysis for **Hack**

Pysa

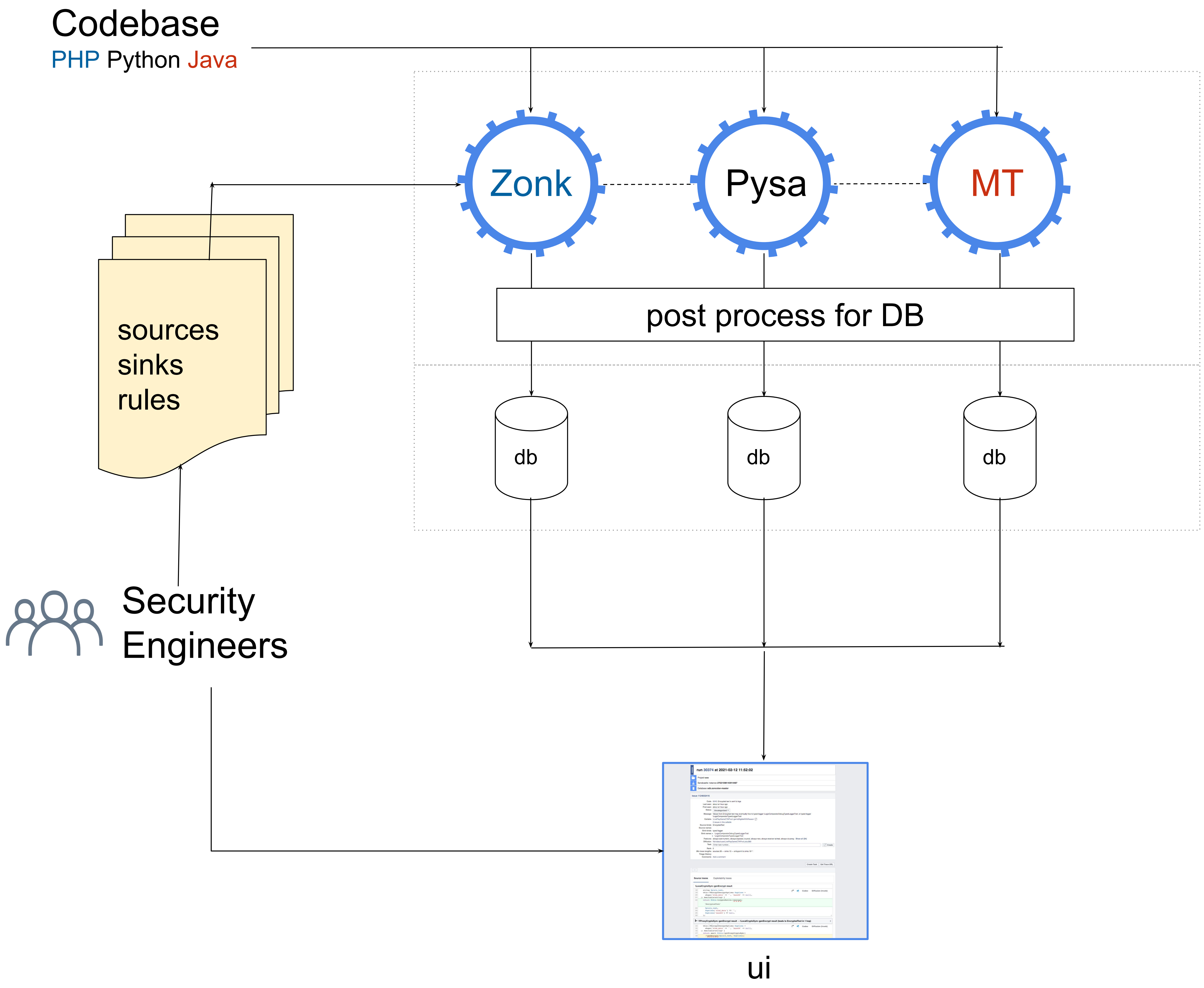
Taint-flow analysis for **Python**

https://pyre-check.org/docs/pysa-basics/

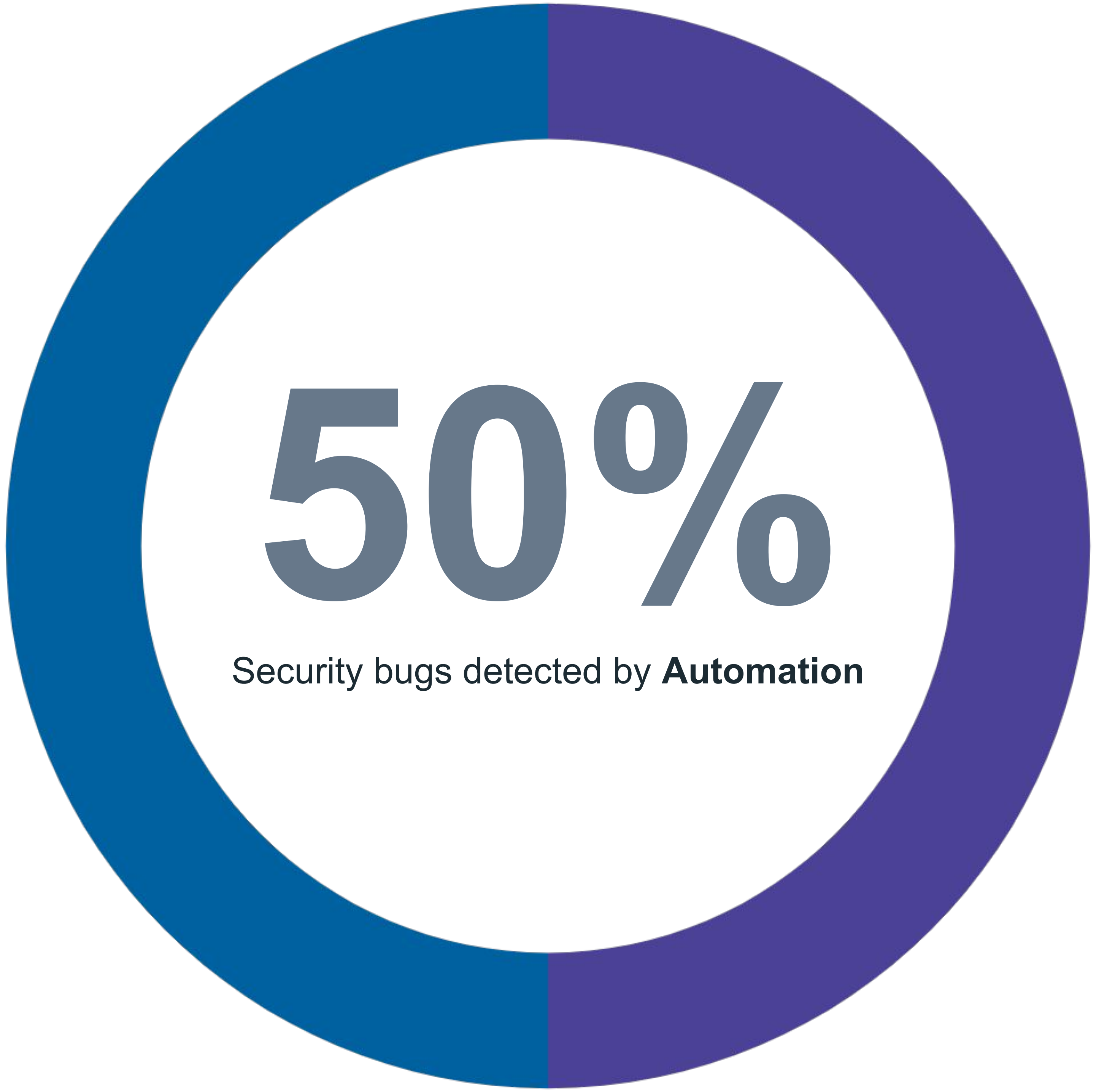
Mariana Trench (MT)

Taint-flow analysis for **Java** and **Android** code

https://github.com/facebook/mariana-trench


Codebase
PHP Python Java
Zonk
Pysa
MT
post process for DB
sources
sinks
rules
db
db
db
Security
Engineers
ui

Taint-flow analysis @ FB



50%

Security bugs detected by **Automation**

Cross-repo taint-flow analysis

SQL injection!

```
class ViewPage extends FacebookEndpoint {
    function getResponse() {
      $x = wrapGetPageName();
      $clause = ‘name = ‘ . $x;
      ...
      $client = new PageServiceClient(...);
      $client->fetchData($clause);
    }
}
```

```
class PageServiceHandler:
    def __init__(self):
        self.db = MySQLdb.connect(...)

    def fetchData(self, clause):
        cursor = db.cursor()
        query = “select * from pages” + clause
        return cursor.execute(query).fetchone()

if __name__ == '__main__':
    server = TSimpleServer(PageServiceHandler)
    print('Starting the server...')
    server.serve()
```

```
class ViewPage extends FacebookEndpoint {
   function getResponse() {
    $x = wrapGetPageName();
    $clause = 'name = ' . $x;
```

Thrift: lightweight, language-independent software stack for point-to-point RPC implementation. Thrift provide abstractions for data transport, data serialization, and application level processing.

src: https://github.com/apache/thrift

```
    $client = new PageServiceAsyncClient($protocol);

    $client->fetchData($clause);
   }
}
```

RETURN data from $_REQUEST

Call to a thrift service

Python server implementation

```
Import MySQLdb
class PageServiceHandler:
    def __init__(self):
        self.log = {}
        self.db = MySQLdb.connect("host","user","pwd","db")

    def fetchData(self, clause):
        cursor = db.cursor()
        query = “select * from pages where” + clause
        return cursor.execute(query).fetchone()

if __name__ == '__main__':
    handler = PageServiceHandler()
    processor = PageService.Processor(handler)
    server = TSimpleServer(processor)
    print('Starting the server...')
    server.serve()
```

Python server implementation

```
...
def fetchData(self, clause):
    cursor = db.cursor()
    query = “select * from pages where” + clause
    return cursor.execute(query).fetchone()
  ...
```

THRIFT arguments

Argument flows into SQLi sink

SQL injection!

```
class ViewPage extends FacebookEndpoint {
    function getResponse() {
      $x = wrapGetPageName();
      $clause = ‘name = ‘ . $x;
      ...
      $client = new PageServiceClient(...);
      $client->fetchData($clause);
    }
}
```

```
class PageServiceHandler:
    def __init__(self):
        self.db = MySQLdb.connect(...)

    def fetchData(self, clause):
        cursor = db.cursor()
        query = “select * from pages” + clause
        return cursor.execute(query).fetchone()

if __name__ == '__main__':
    server = TSimpleServer(PageServiceHandler)
    print('Starting the server...')
    server.serve()
```

How to find this with static analysis?

- If we have PHP static analysis tool - Zoncolan!
  - Review the code
  - Identify calls to thrift services
- If we have Python static analysis tool - Pysa!
  - Review the code
  - Identify thrift server implementation
- Automagically make both share the information to find the SQLi

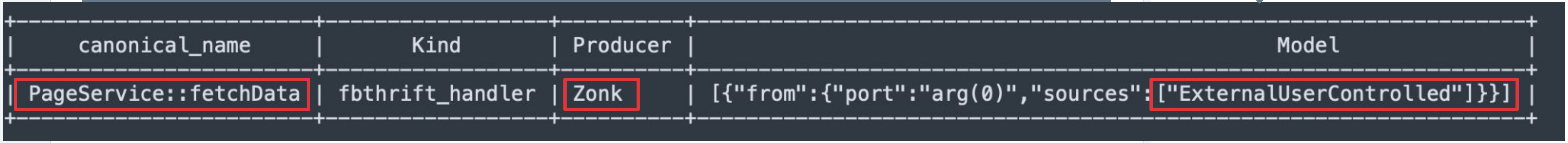
PHP client implementation

```
class ViewPage extends FacebookEndpoint {
    function getResponse() {
      $x = wrapGetPageName();
      $clause = ‘name = ‘ . $x;
      ...
      $client = new PageServiceClient(...);
      $client->fetchData($clause);
    }
}
```

1. Analyze code normally

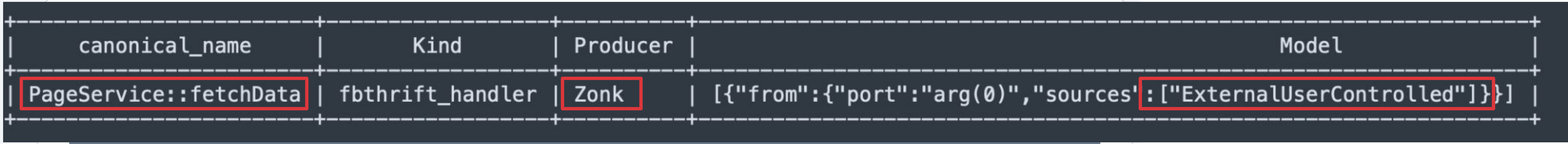
2. Identify flows to thrift RPC calls

3. resolve canonical connection point and store partial flow



```
+------------------------+------------------+----------+------------------------------------------------------------------+
|     canonical_name     |       Kind       | Producer |                              Model                               |
+------------------------+------------------+----------+------------------------------------------------------------------+
| PageService::fetchData | fbthrift_handler | Zonk     | [{"from":{"port":"arg(0)","sources":["ExternalUserControlled"]}}] |
+------------------------+------------------+----------+------------------------------------------------------------------+
```

Python server implementation



```
+-----------------------+------------------+----------+-------------------------------------------------------------------+
|    canonical_name     |       Kind       | Producer |                               Model                               |
+-----------------------+------------------+----------+-------------------------------------------------------------------+
| PageService::fetchData | fbthrift_handler | Zonk     | [{"from":{"port":"arg(0)","sources":["ExternalUserControlled"]}}] |
+-----------------------+------------------+----------+-------------------------------------------------------------------+
```

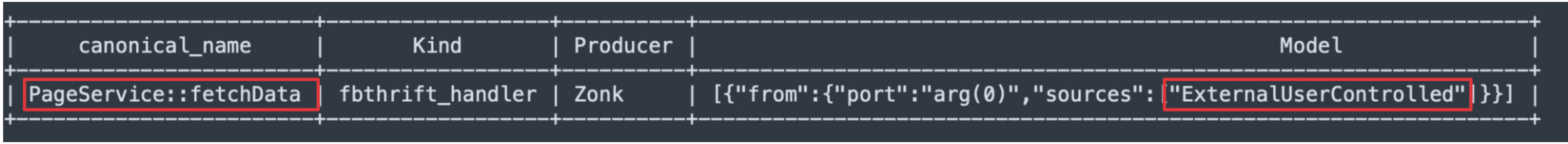
```
class PageServiceHandler:
    ...
    ...
    def fetchData(self, clause):
        cursor = db.cursor()
        query = “select * from pages where” + clause
        return cursor.execute(query).fetchone()
      ...
```

4. Identity thrift service implementations

5. Look up canonical point information PageService:FetchData

6. Engine augments initial models with producers information

Cross-repo Taint flows - storing partial flows



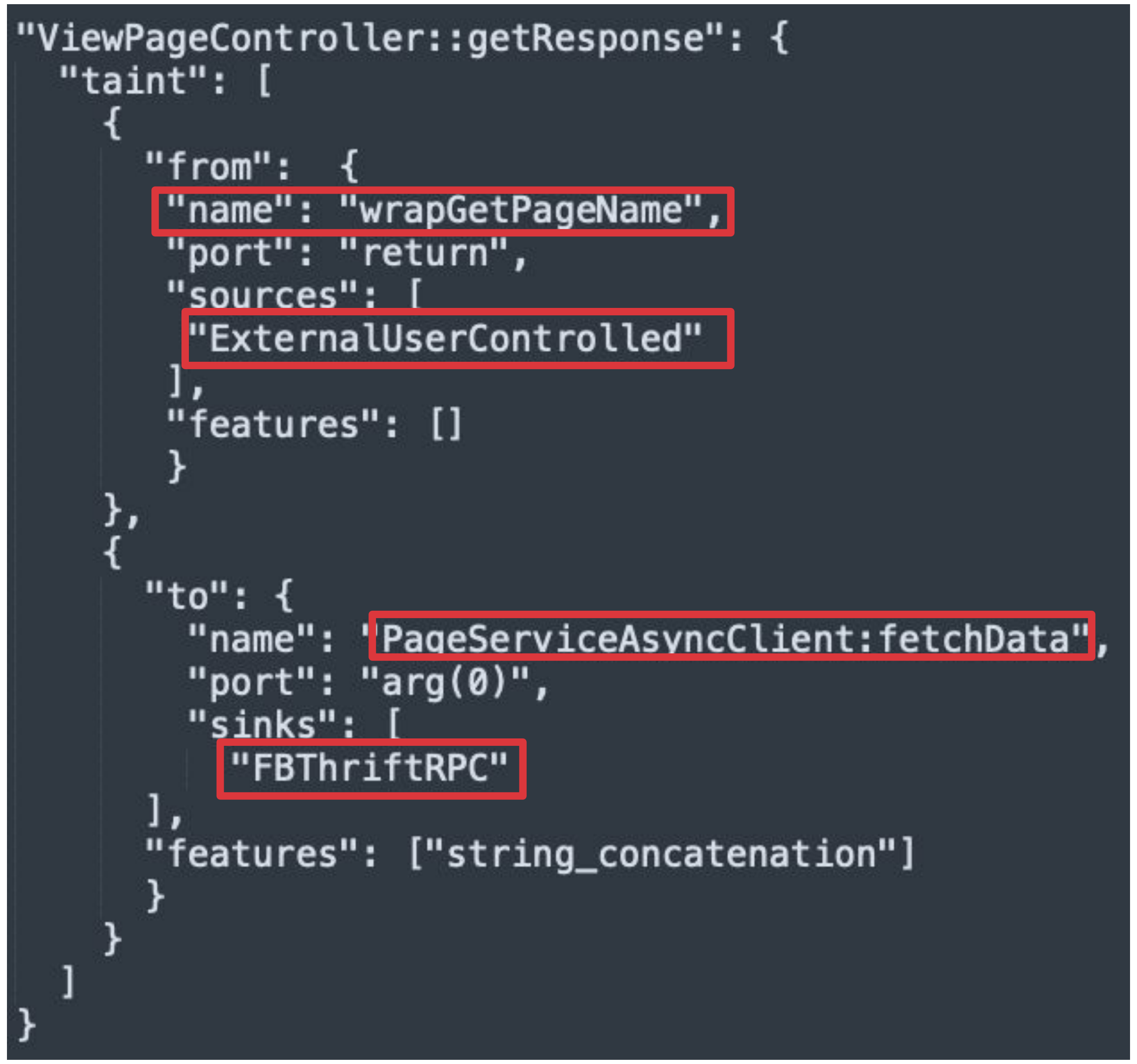
```
+------------------------+------------------+----------+----------------------------------------------------------------+
|     canonical_name     |       Kind       | Producer |                              Model                             |
+------------------------+------------------+----------+----------------------------------------------------------------+
| PageService::fetchData | fbthrift_handler | Zonk     | [{"from":{"port":"arg(0)","sources":["ExternalUserControlled"]}}] |
+------------------------+------------------+----------+----------------------------------------------------------------+
```

```
...
def fetchData(self, clause):
  cursor = db.cursor()
  query = “select * from pages where” +
          clause;
  return cursor.execute(query)
...
```

3. We see thrift handler, look up db and augment initial models

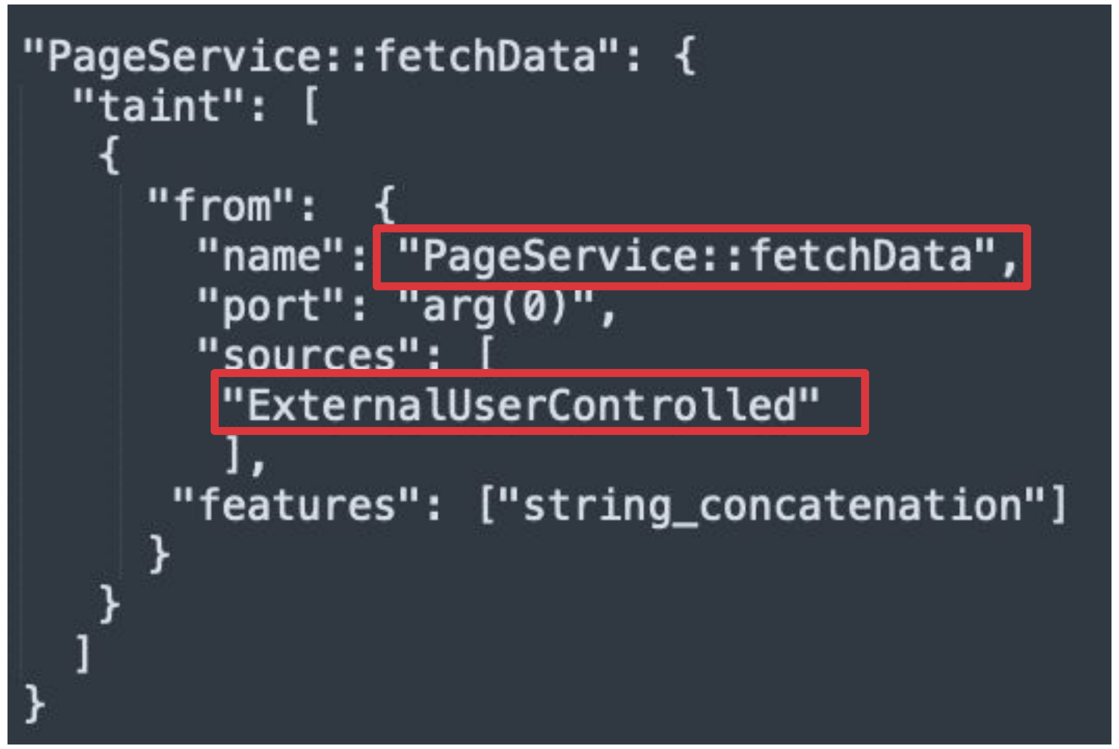
4. Cross-language SQL injection!

Producer run - storing partial flows

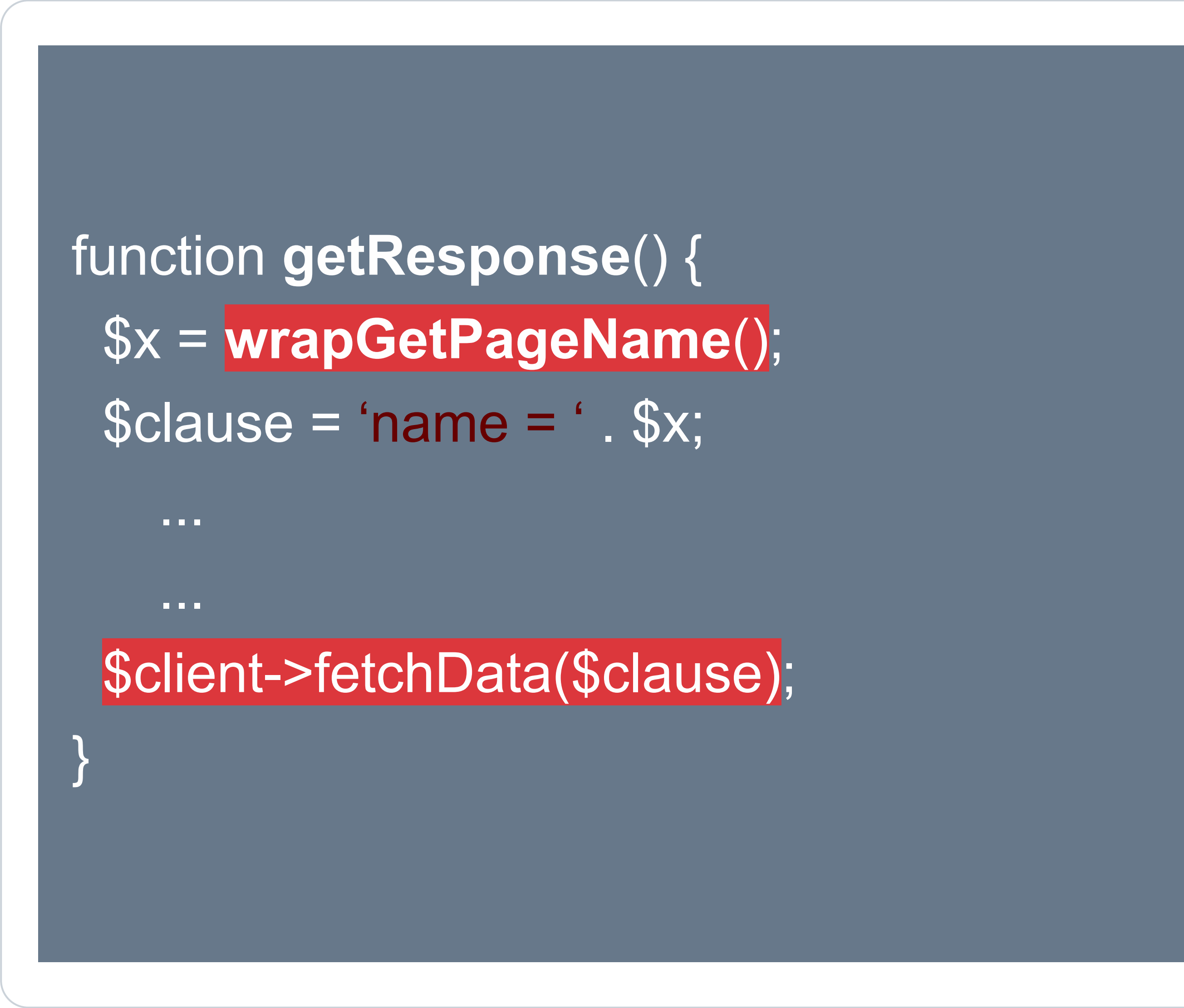


```
"ViewPageController::getResponse": {
  "taint": [
    {
      "from":  {
        "name": "wrapGetPageName",
        "port": "return",
        "sources": [
          "ExternalUserControlled"
        ],
        "features": []
      }
    },
    {
      "to": {
        "name": "PageServiceAsyncClient:fetchData",
        "port": "arg(0)",
        "sinks": [
          "FBThriftRPC"
        ],
      "features": ["string_concatenation"]
      }
    }
  ]
}
```

1

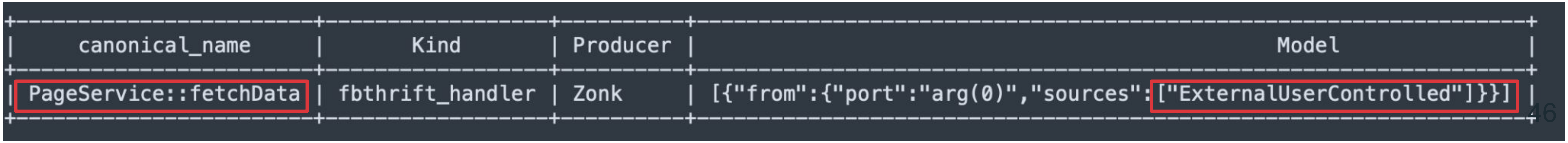


```
"PageService::fetchData": {
  "taint": [
   {
     "from":  {
       "name": "PageService::fetchData",
       "port": "arg(0)",
       "sources": [
         "ExternalUserControlled"
       ],
      "features": ["string_concatenation"]
     }
   }
  ]
}
```



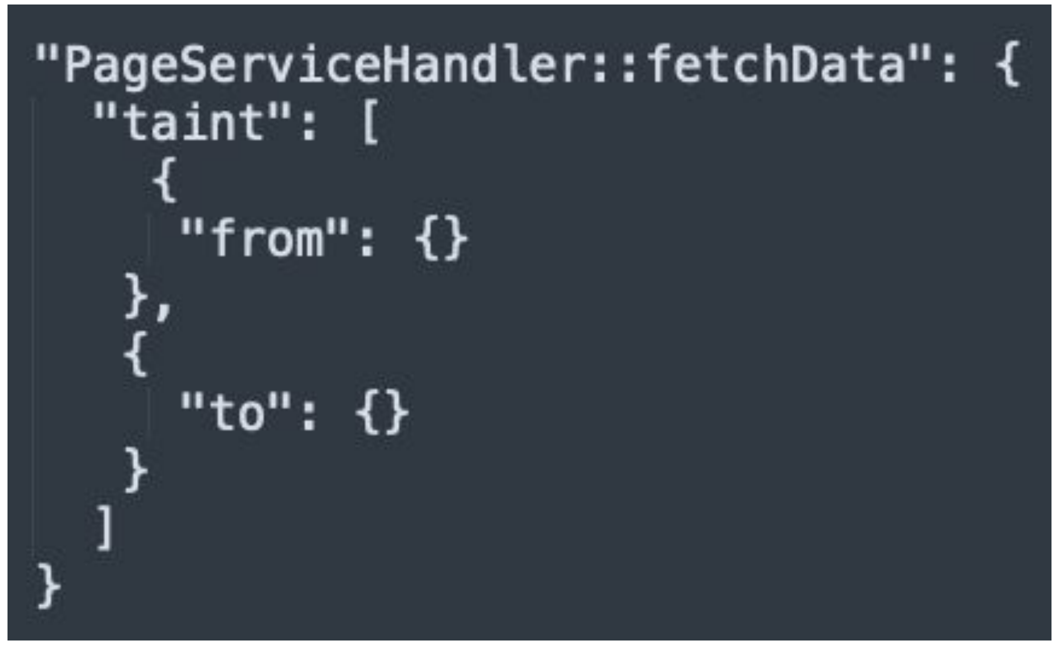
```
function getResponse() {
  $x = wrapGetPageName();
  $clause = 'name = ' . $x;
    ...
    ...
  $client->fetchData($clause);
}
```

2



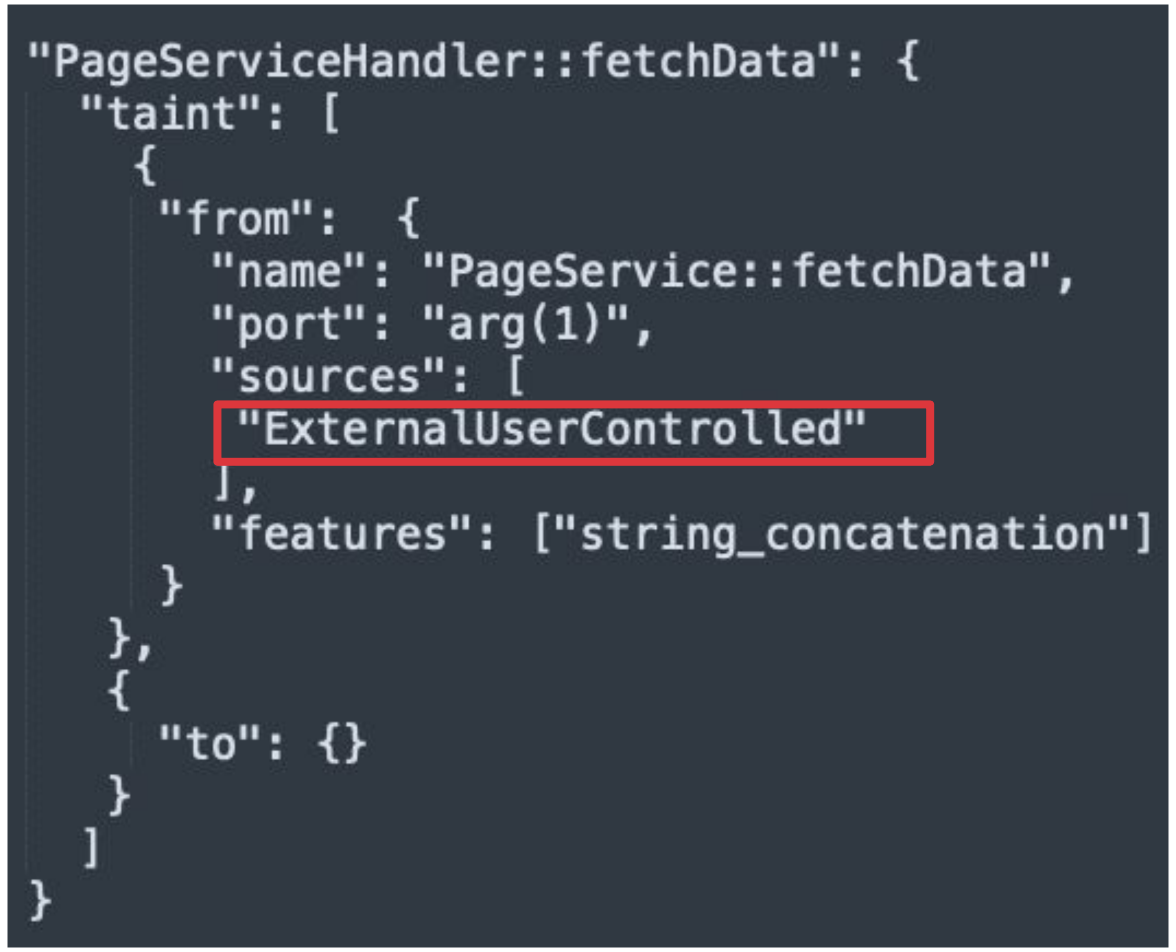
```
+----------------------+-----------------+----------+------------------------------------------------------------------+
|    canonical_name    |      Kind       | Producer |                              Model                               |
+----------------------+-----------------+----------+------------------------------------------------------------------+
| PageService::fetchData | fbthrift_handler | Zonk     | [{"from":{"port":"arg(0)","sources":["ExternalUserControlled"]}}] |
+----------------------+-----------------+----------+------------------------------------------------------------------+
```

Cross-repo Taint flows - Loading partial flows



```
"PageServiceHandler::fetchData": {
  "taint": [
    {
     "from": {}
    },
    {
     "to": {}
    }
  ]
}
```

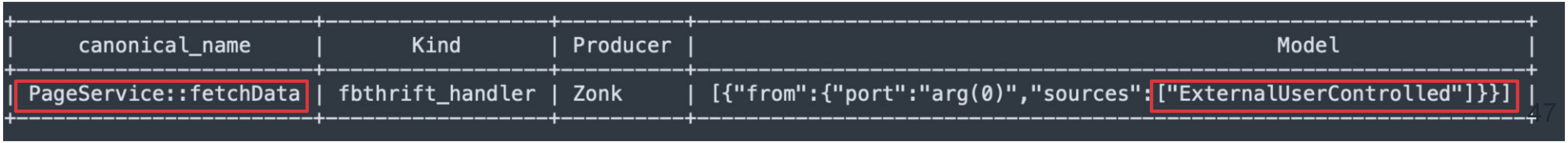
2



```
"PageServiceHandler::fetchData": {
  "taint": [
    {
      "from":  {
        "name": "PageService::fetchData",
        "port": "arg(1)",
        "sources": [
         "ExternalUserControlled"
        ],
        "features": ["string_concatenation"]
      }
    },
    {
      "to": {}
    }
  ]
}
```

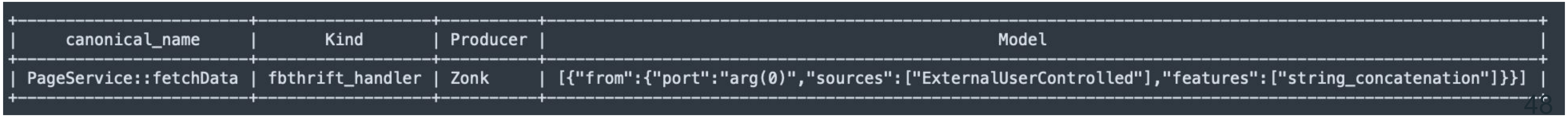
```
...
def fetchData(self, clause):
  cursor = db.cursor()
  query = “select * from pages where” + claus
  return cursor.execute(query)
...
```

1



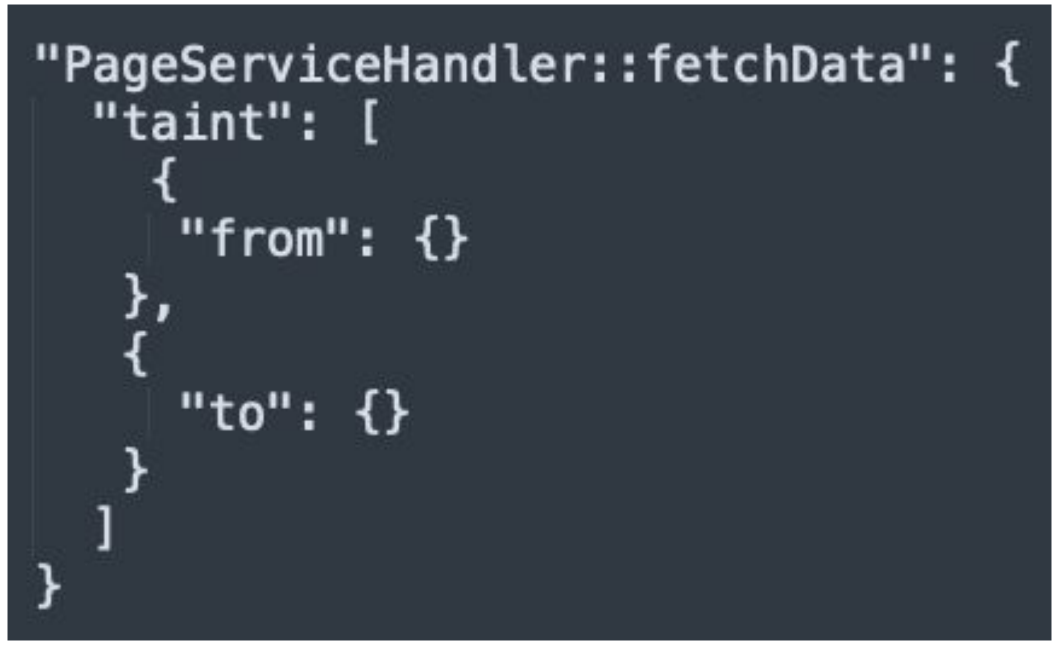
```
+--------------------------+-------------------+-----------+----------------------------------------------------------------------+
|      canonical_name      |       Kind        | Producer  |                                Model                                 |
+--------------------------+-------------------+-----------+----------------------------------------------------------------------+
| PageService::fetchData   | fbthrift_handler  | Zonk      | [{"from":{"port":"arg(0)","sources":["ExternalUserControlled"]}}]   |
+--------------------------+-------------------+-----------+----------------------------------------------------------------------+
```

Cross-repo Taint flows - Full analysis view



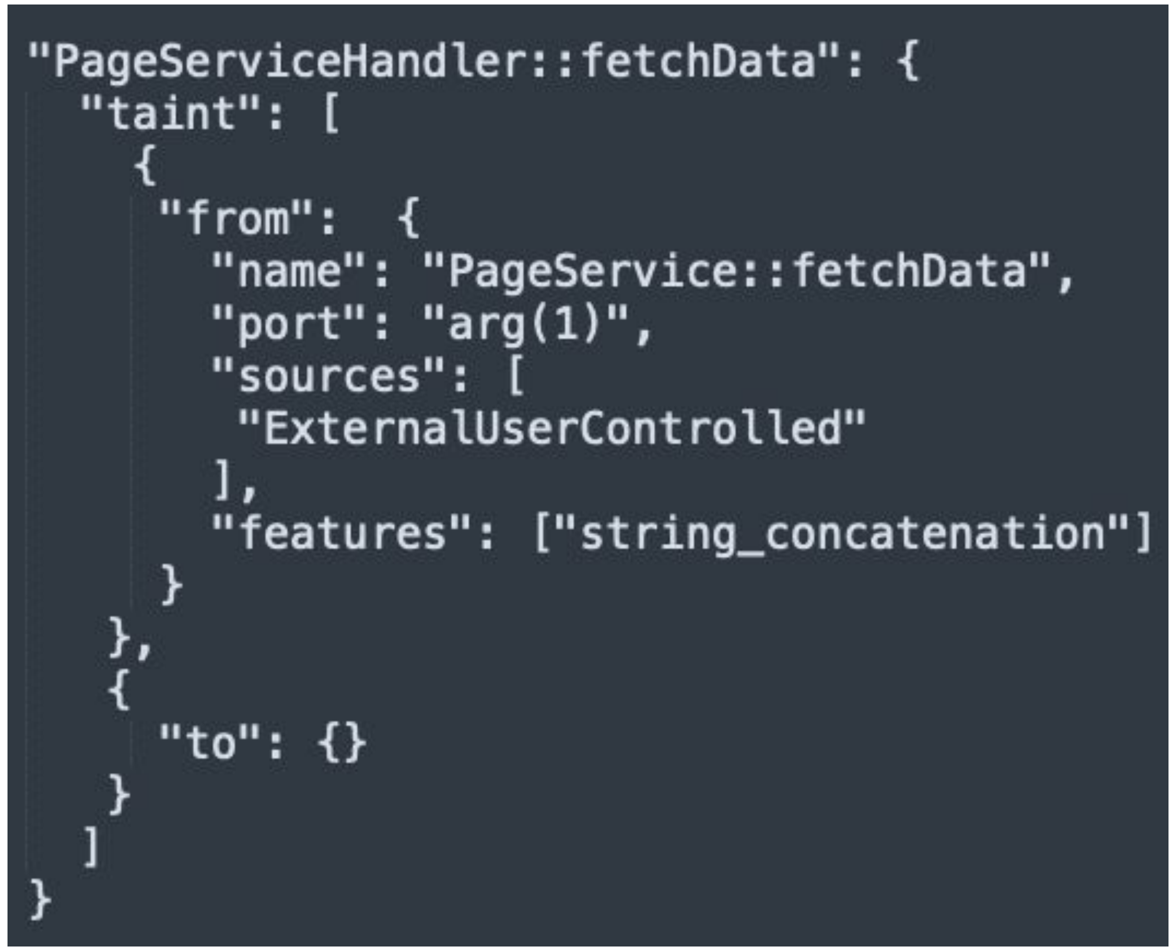
```
+-------------------------+-------------------+----------+--------------------------------------------------------------------------------------------------+
|      canonical_name     |        Kind       | Producer |                                              Model                                               |
+-------------------------+-------------------+----------+--------------------------------------------------------------------------------------------------+
| PageService::fetchData  | fbthrift_handler  | Zonk     | [{"from":{"port":"arg(0)","sources":["ExternalUserControlled"],"features":["string_concatenation"]}}] |
+-------------------------+-------------------+----------+--------------------------------------------------------------------------------------------------+
```

1



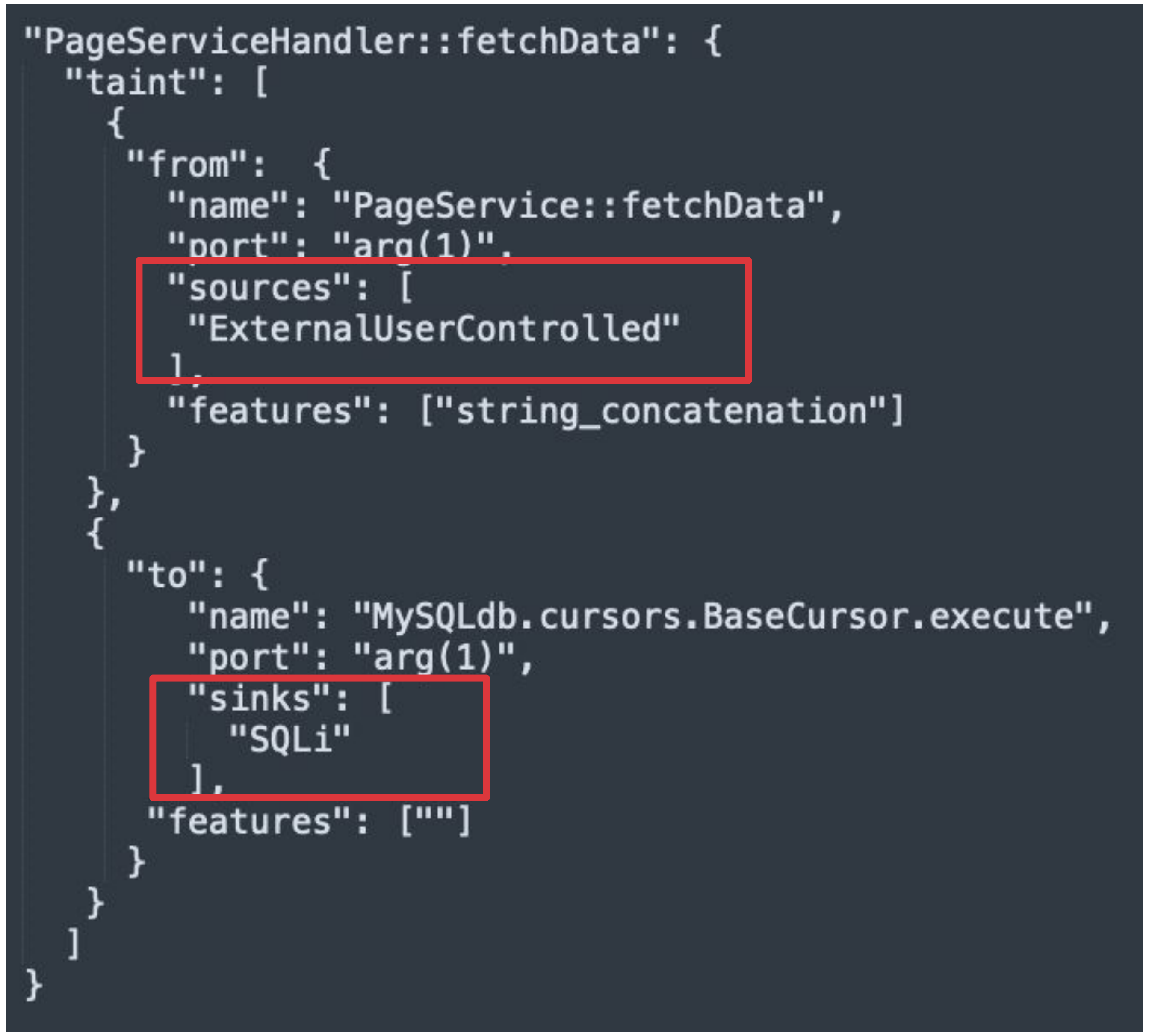
```
"PageServiceHandler::fetchData": {
  "taint": [
    {
     "from": {}
   },
   {
     "to": {}
   }
  ]
}
```

2



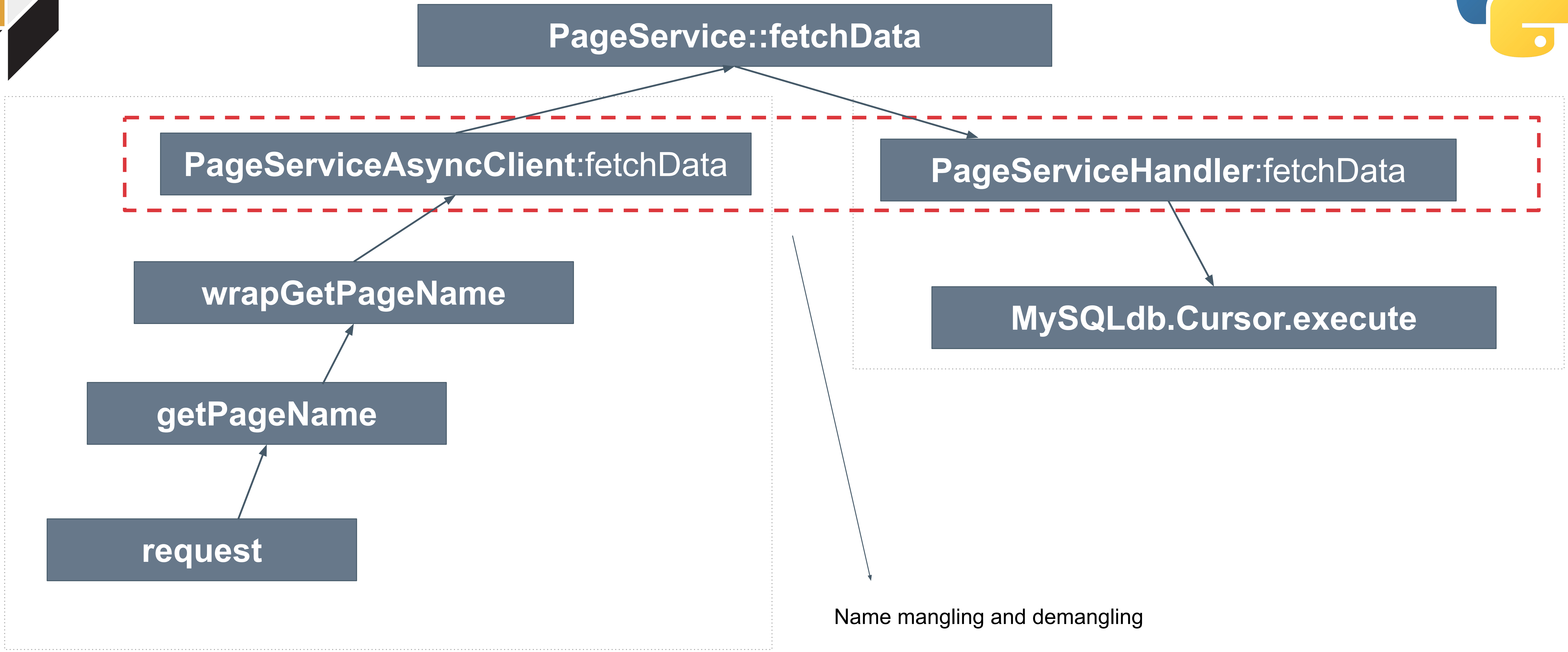
```
"PageServiceHandler::fetchData": {
  "taint": [
    {
      "from":  {
        "name": "PageService::fetchData",
        "port": "arg(1)",
        "sources": [
         "ExternalUserControlled"
        ],
        "features": ["string_concatenation"]
      }
    },
    {
      "to": {}
    }
  ]
}
```

3



```
"PageServiceHandler::fetchData": {
  "taint": [
    {
      "from":  {
        "name": "PageService::fetchData",
        "port": "arg(1)",
        "sources": [
         "ExternalUserControlled"
        ],
        "features": ["string_concatenation"]
      }
    },
    {
      "to": {
         "name": "MySQLdb.cursors.BaseCursor.execute",
         "port": "arg(1)",
         "sinks": [
           "SQLi"
         ],
        "features": [""]
      }
    }
  ]
}
```

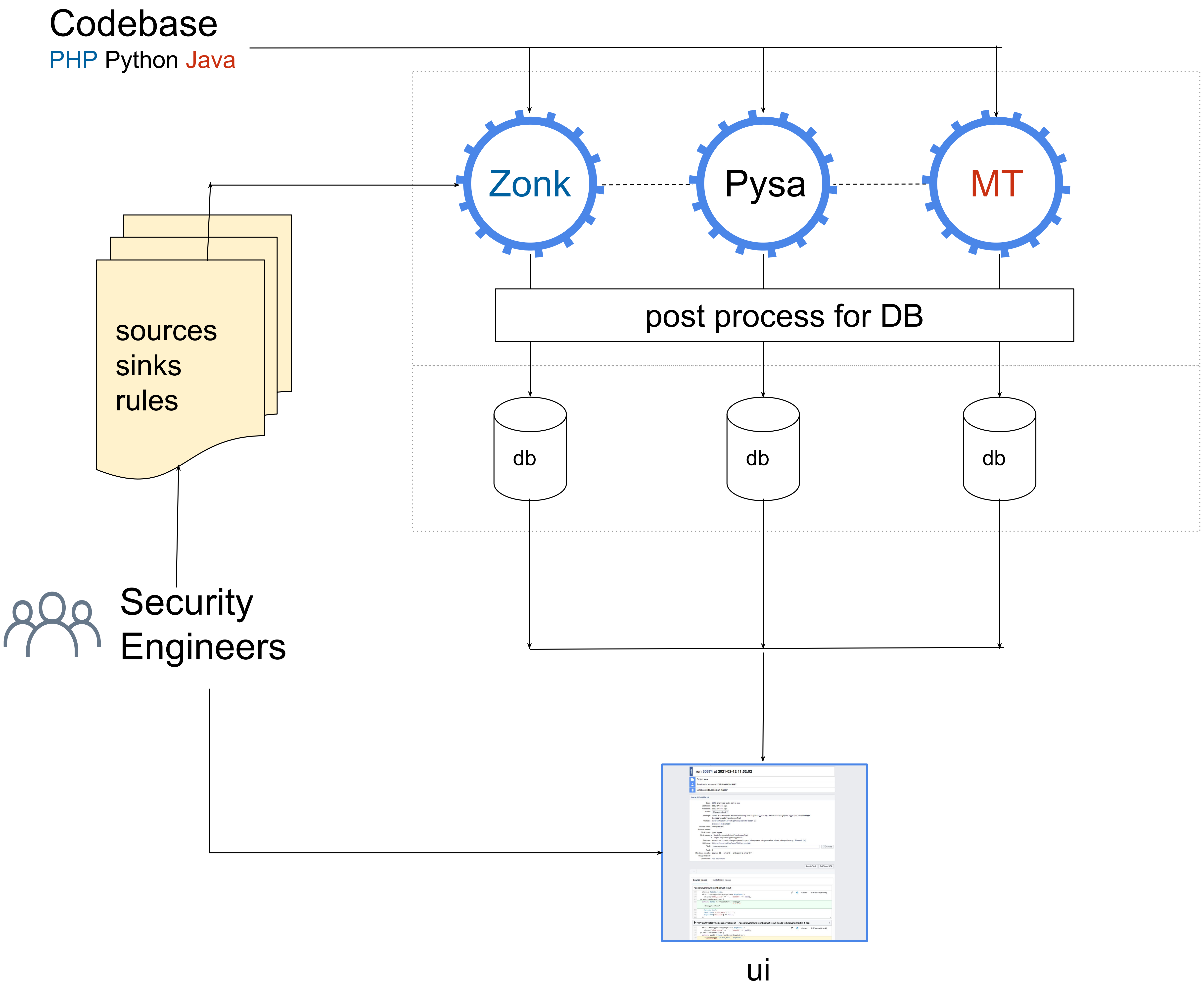
Trace in our Example



Name mangling and demangling

Cross-Repo Taint Analysis

- Mark all RPC calls as sinks
- Defines canonical connection points
  (e.g. Fbthrift, Thrift, gRPC)
- Allow engines to store partial flows
  (e.g. UserControlled to Thrift) - Producers
- Allow engines to load partial flows augmenting initial models
  (e.g. UserControlled via Thrift) - Consumers
- Define format to visualize cross-repo traces



Codebase

PHP Python Java

Zonk

Pysa

MT

post process for DB

db

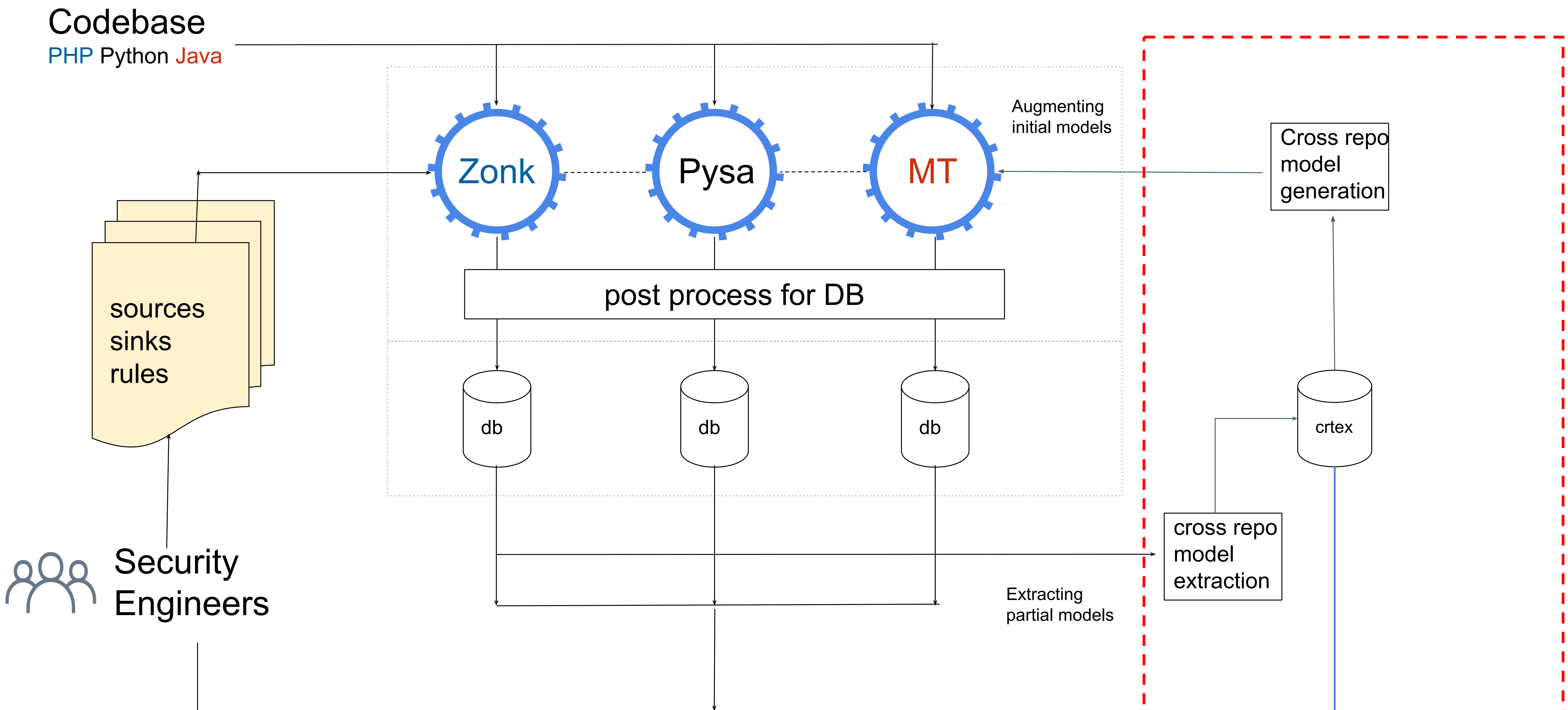
db

db

sources
sinks
rules

Security
Engineers

ui



Codebase

PHP Python Java

Augmenting initial models

Zonk

Pysa

MT

Cross repo model generation

sources sinks rules

post process for DB

db

db

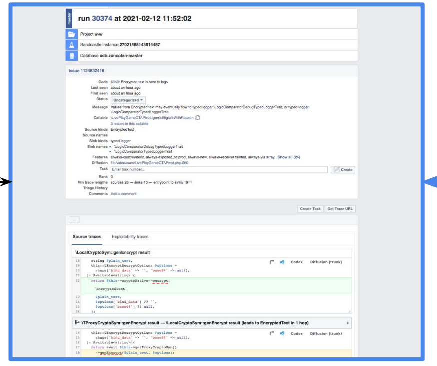
db

crtex

cross repo model extraction

Security Engineers

Extracting partial models



run 30374 at 2021-02-12 11:52:02

Project www

Sandcastle instance 27021598143914487

Database xdb.zoncolan-master

Issue 1124832416

Code 6343: Encrypted text is sent to logs

Last seen about an hour ago

First seen about an hour ago

Status Uncategorized

Message Values from Encrypted text may eventually flow to typed logger \LogicComparatorDebugTypedLoggerTrait, or typed logger \LogicComparatorTypedLoggerTrait

Callable \LivePlayGameCTAPivot::genIsEligibleWithReason

3 issues in this callable

Source kinds EncryptedText

Source names

Sink kinds typed logger

Sink names
- \LogicComparatorDebugTypedLoggerTrait
- \LogicComparatorTypedLoggerTrait

Features always-cast:numeric, always-exposed_to:prod, always-new, always-receiver tainted, always-via:array Show all (24)

Diffusion flib/video/cues/LivePlayGameCTAPivot.php:$60

Task Enter task number...

Rank 0

Min trace lengths sources 28 --- sinks 13 --- entrypoint to sinks 19

Triage History

Comments Add a comment

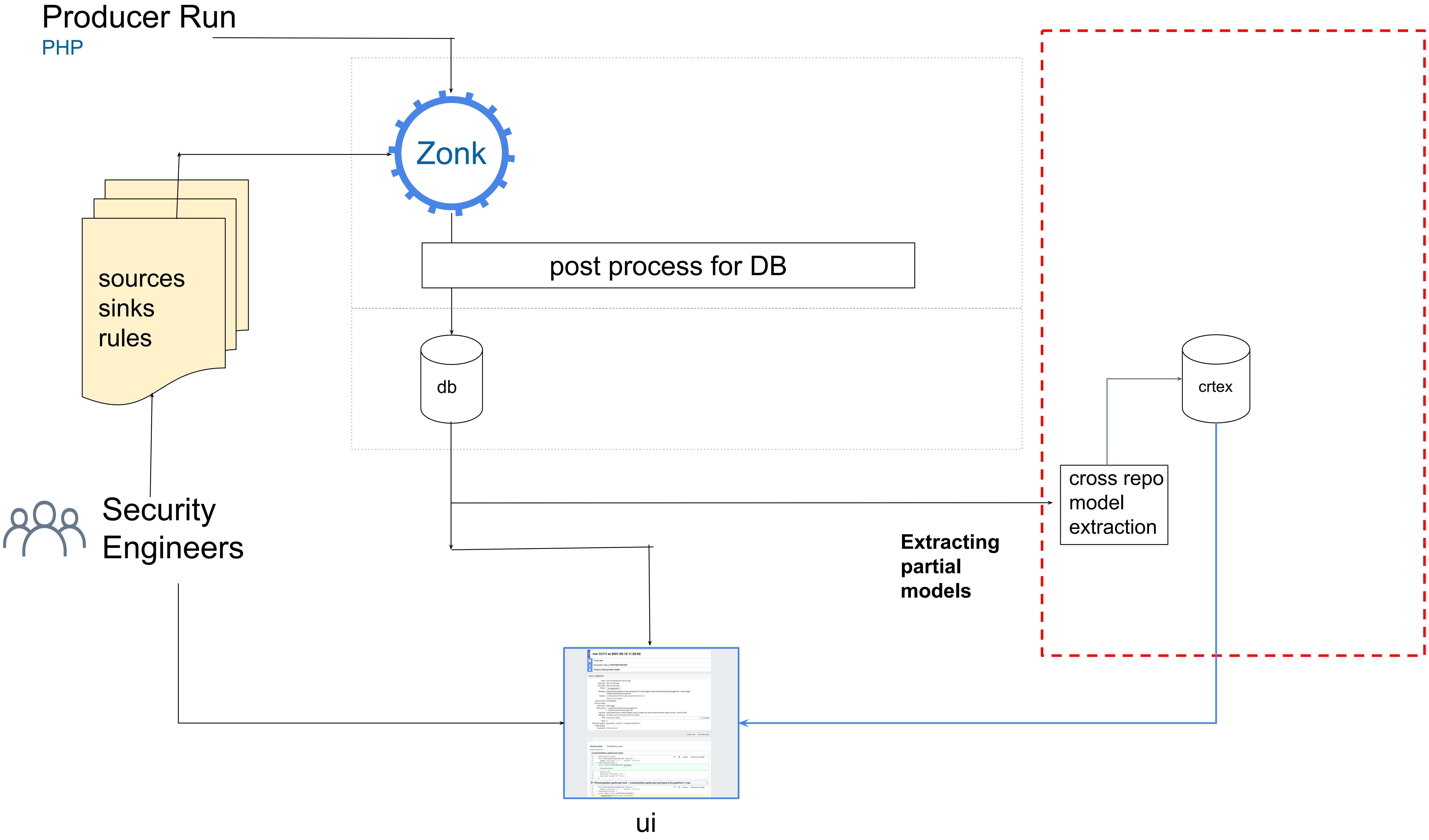
\LocalCryptoSym::genEncrypt result

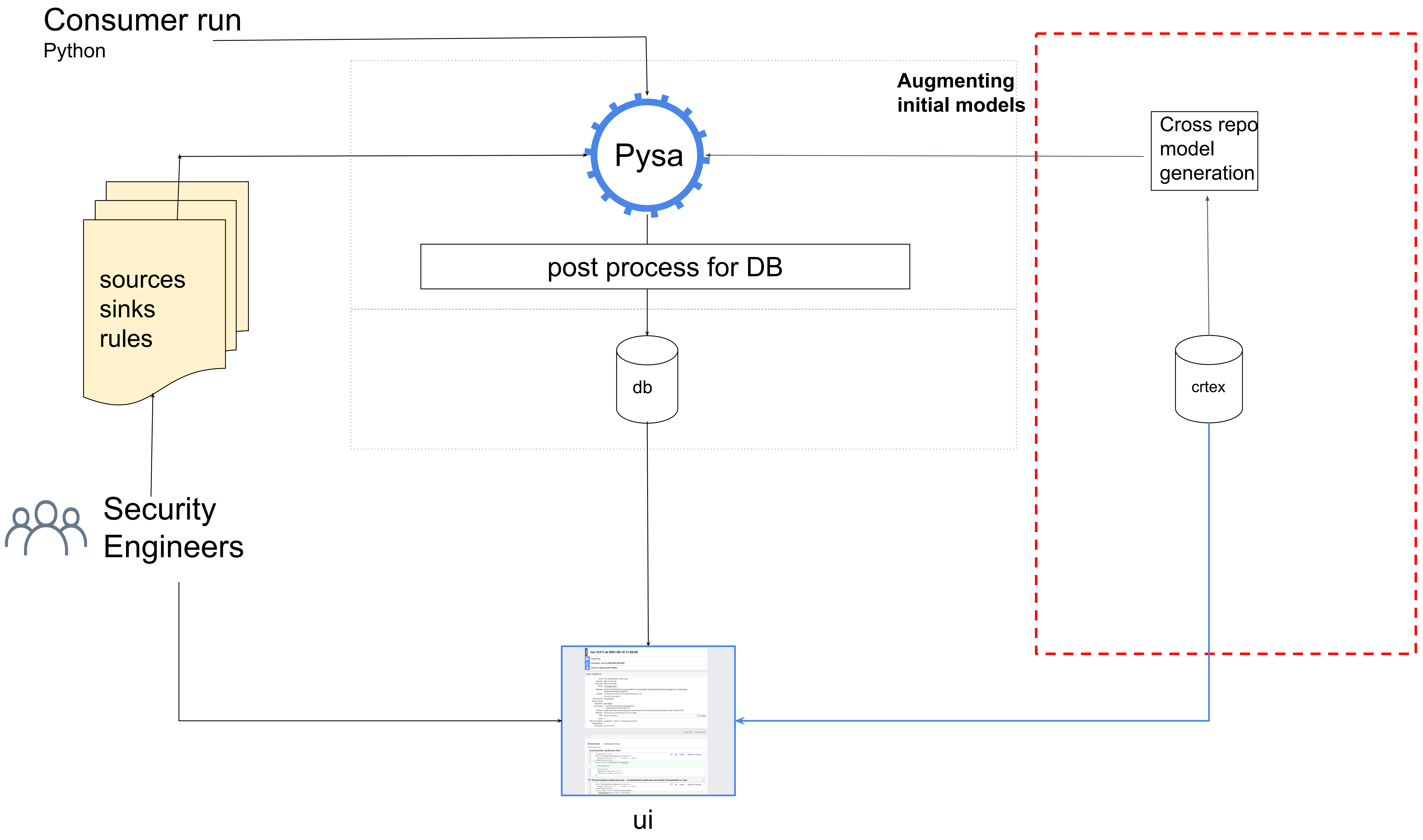
```
    string $plain_text,
    this::TEncryptDecryptOptions $options =
      shape('bind_data' => '', 'base64' => null),
  ): Awaitable<string> {
    return $this->cryptoNative->encrypt(

     'EncryptedText'

      $plain_text,
      $options['bind_data'] ?? '',
      $options['base64'] ?? null,
    );
```

\TProxyCryptoSym::genEncrypt result → \LocalCryptoSym::genEncrypt result (leads to EncryptedText in 1 hop)

```
    this::TEncryptDecryptOptions $options =
      shape('bind_data' => '', 'base64' => null),
  ): Awaitable<string> {
    return await $this->getProxyCryptoSym()
      ->genEncrypt($plain_text, $options);
```

ui


Producer Run
PHP
Zonk
post process for DB
db
sources
sinks
rules
Security
Engineers
crtex
cross repo
model
extraction
Extracting
partial
models
ui


Consumer run
Python
Augmenting
initial models
Pysa
Cross repo
model
generation
sources
sinks
rules
post process for DB
db
crtex
Security
Engineers
ui

Cross-Repo Taint-Exchange (CRTEX)

A **tool-independent** store of **taint information** (in a tool agnostic format). The store provides a **push/pull model** which static analysis tools can use to extend their capabilities and analyze flows cross-language

Viewing traces

Viewing single repo traces

1

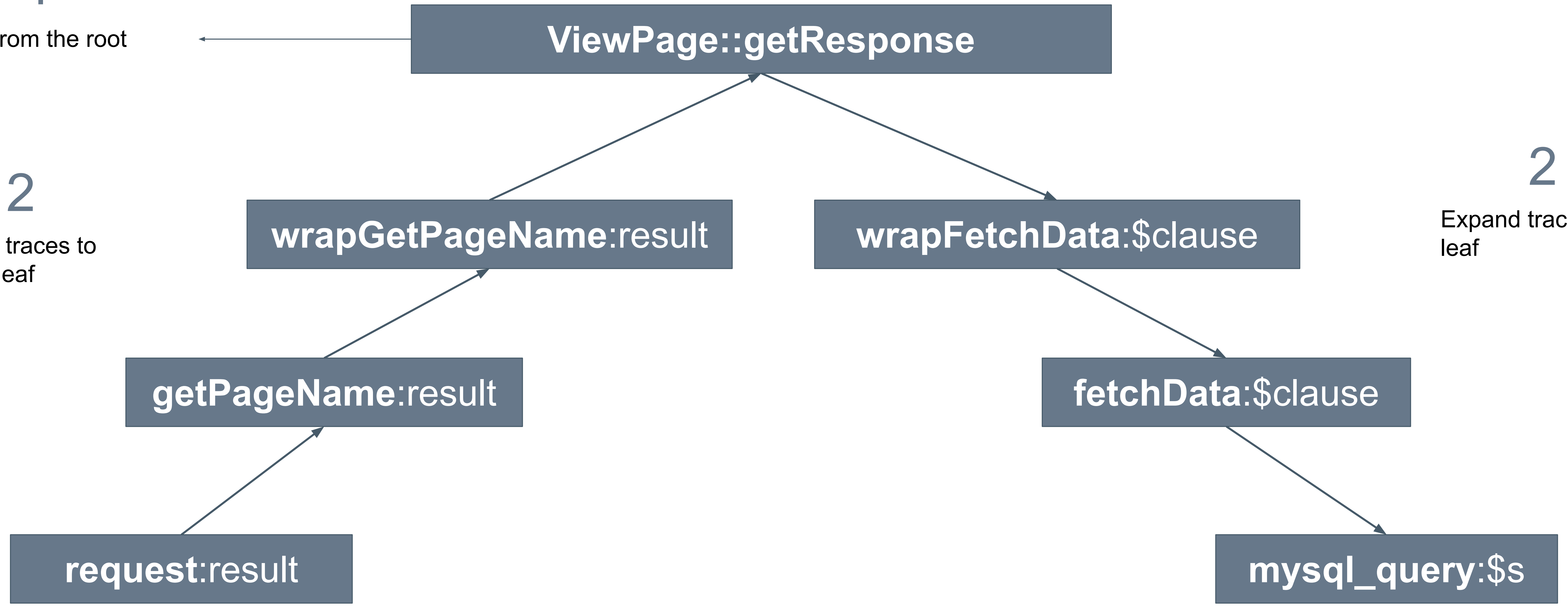
Start from the root

2

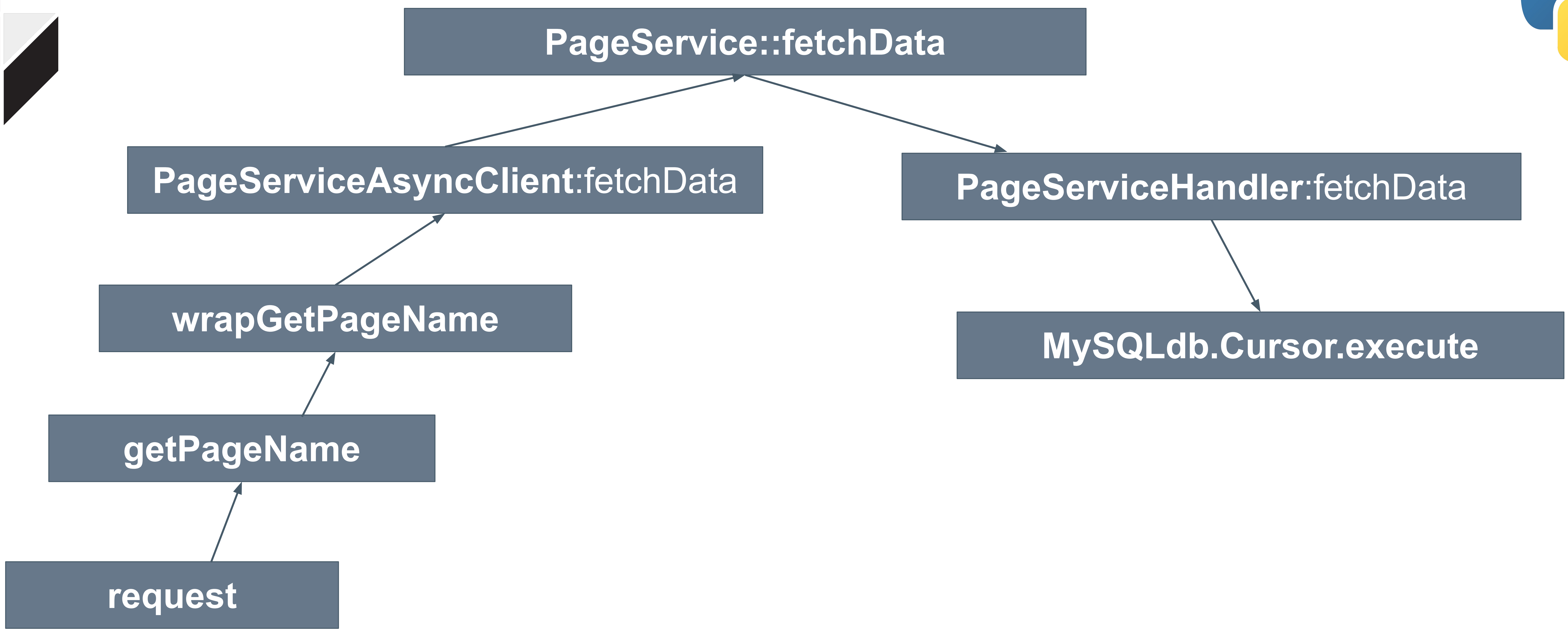
Expand traces to source leaf

2

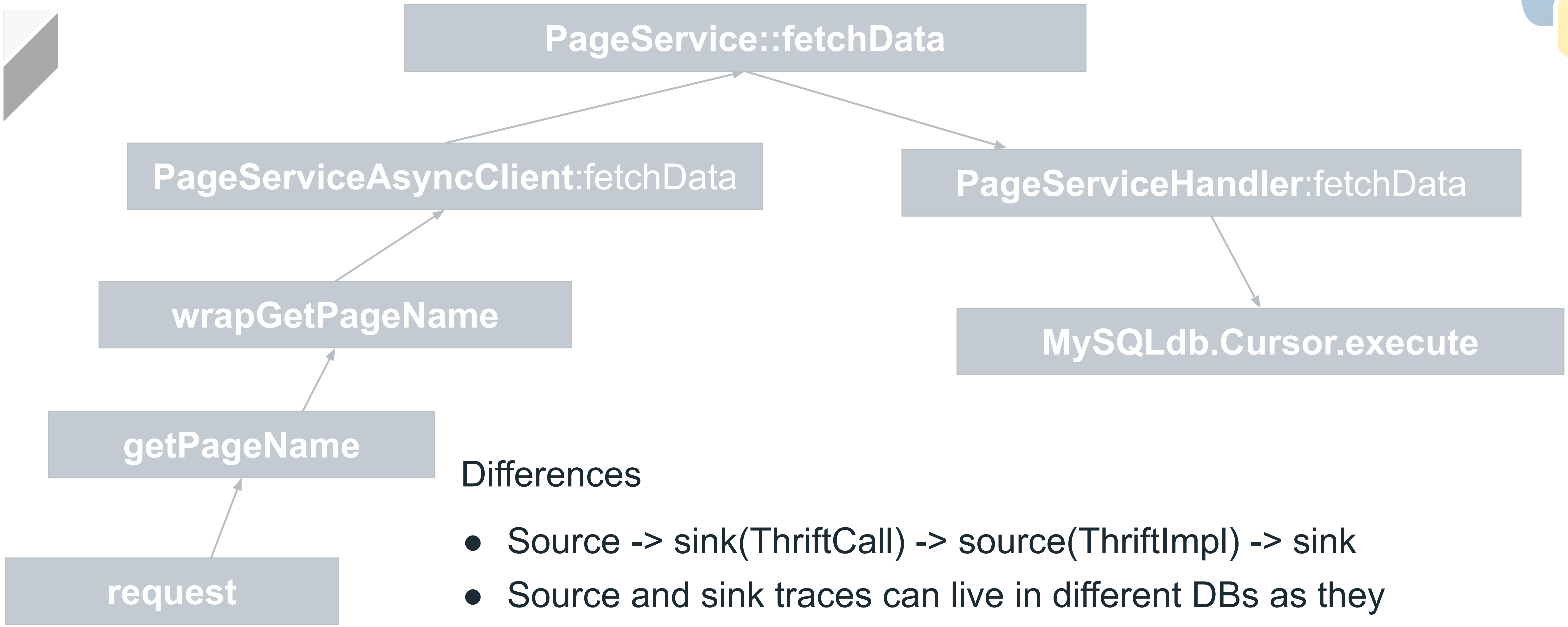
Expand traces to sink leaf



Viewing the cross-repo traces



Viewing the cross-repo traces



PageService::fetchData

PageServiceAsyncClient:fetchData

PageServiceHandler:fetchData

wrapGetPageName

MySQLdb.Cursor.execute

getPageName

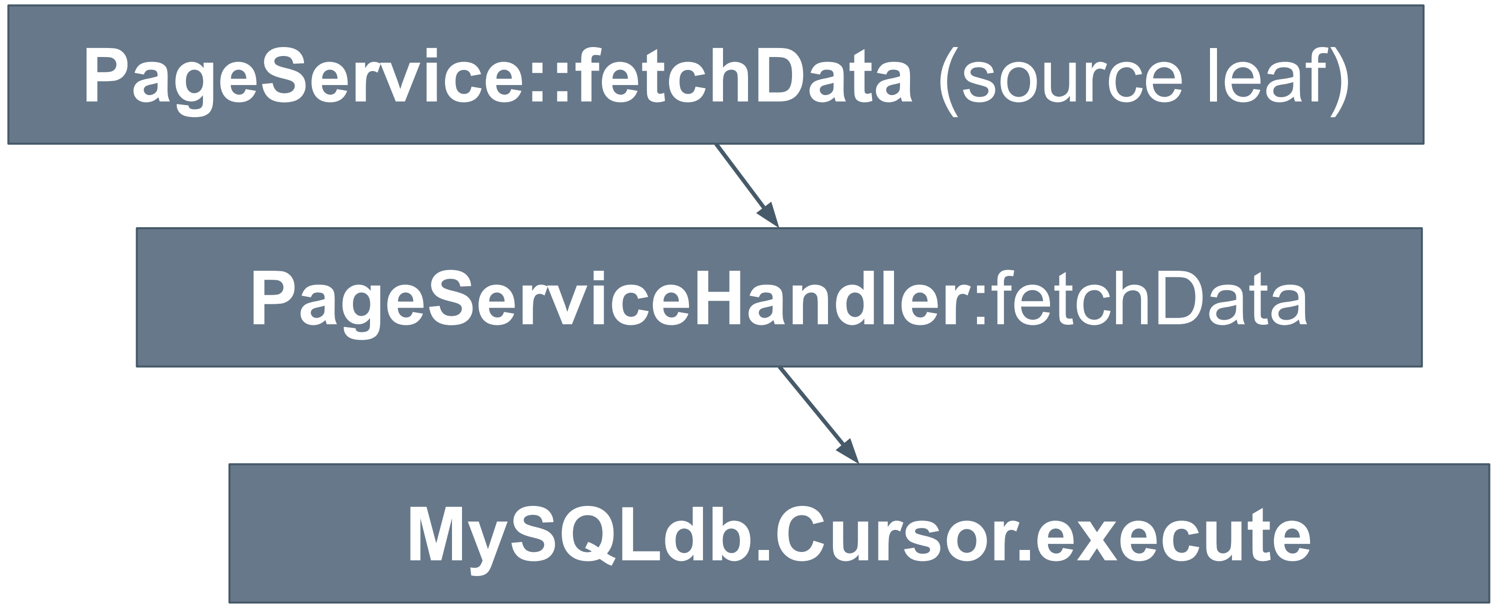
request

Differences

- Source -> sink(ThriftCall) -> source(ThriftImpl) -> sink
- Source and sink traces can live in different DBs as they belong to different engines

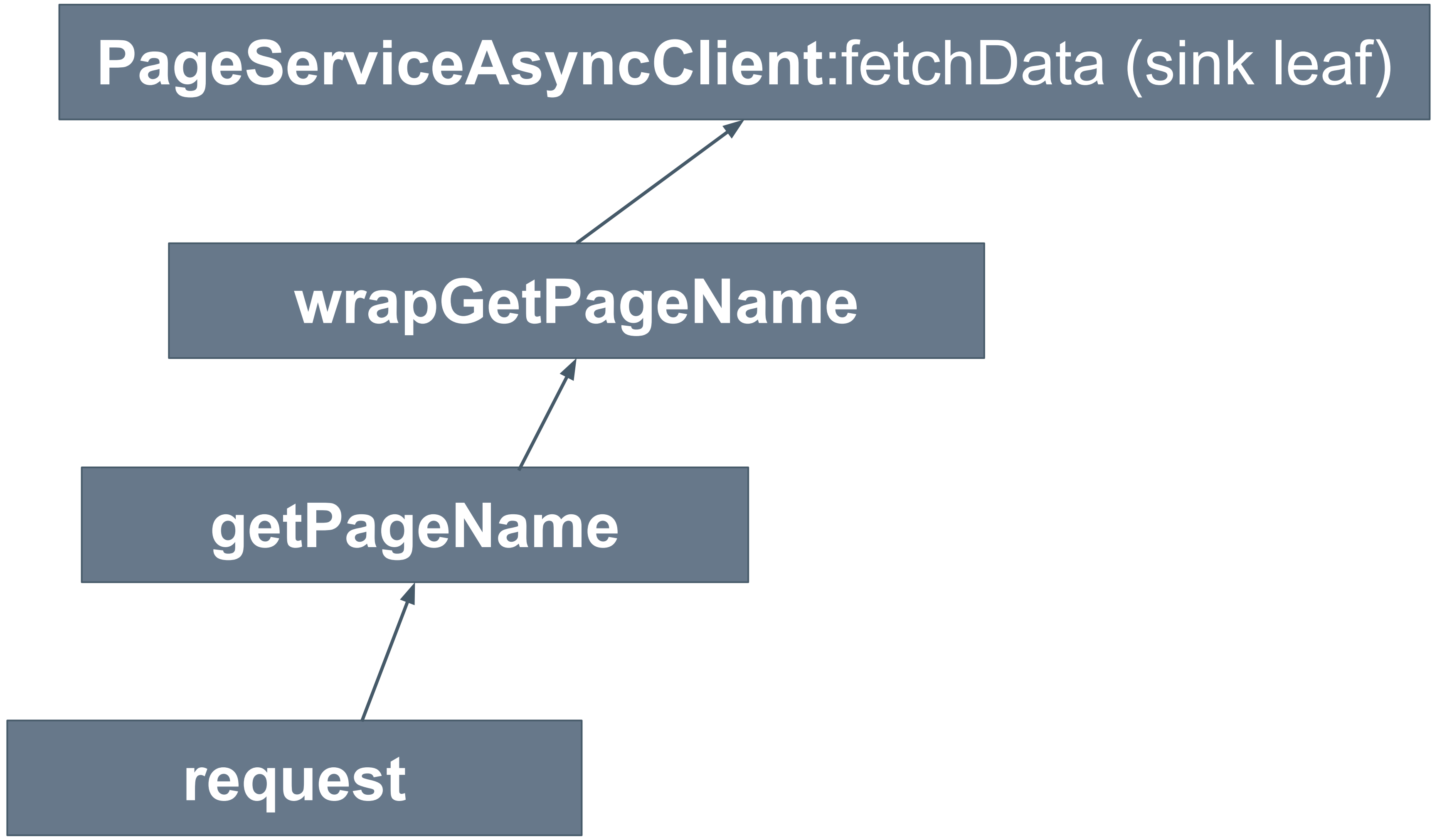
Viewing the cross-language traces

- Expand the source/sink traces normally
- Once you hit the source leaf
  - query CRTEX with canonical points for information about producer runs
- That can be a list of traces


PageService::fetchData (source leaf)
PageServiceHandler:fetchData
MySQLdb.Cursor.execute

Viewing the cross-language traces

- Switch to the right tool's db, and run based on CRTEX
- Start with the thrift sink leaf
- Traverse backward


PageServiceAsyncClient:fetchData (sink leaf)

wrapGetPageName

getPageName

request

Deployment at facebook

Producers

- facebook.com (WWW)
  - Zoncolan
- instagram.com
  - Pysa
- Android mobile apps
  - Mariana Trench

Consumers

- Backend fbthrift services
  - PHP (Zoncolan)
  - Python (Pysa)
  - Java (Mariana Trench)

Deployment at Facebook - analysis schedule

Master

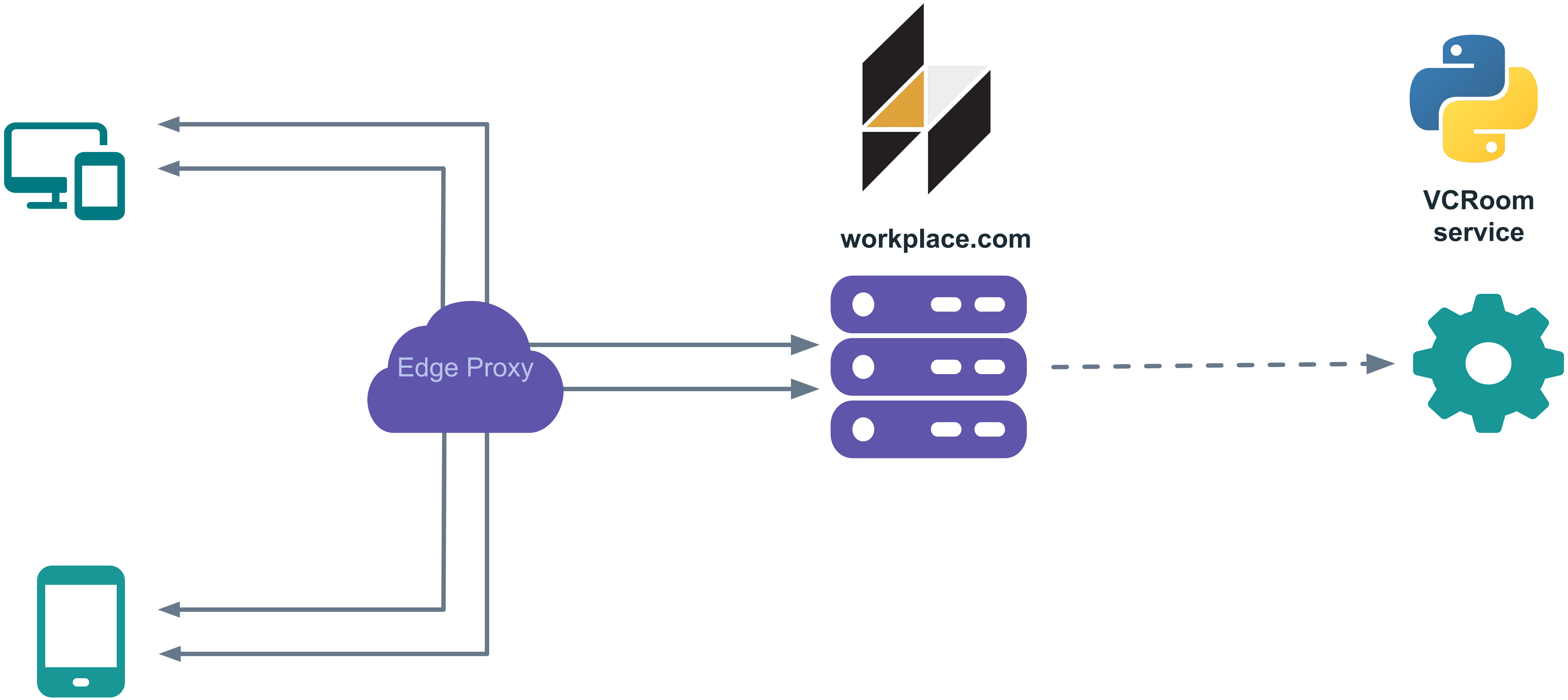
- Periodically - multiple times a day
- File tasks for new findings

Pull requests

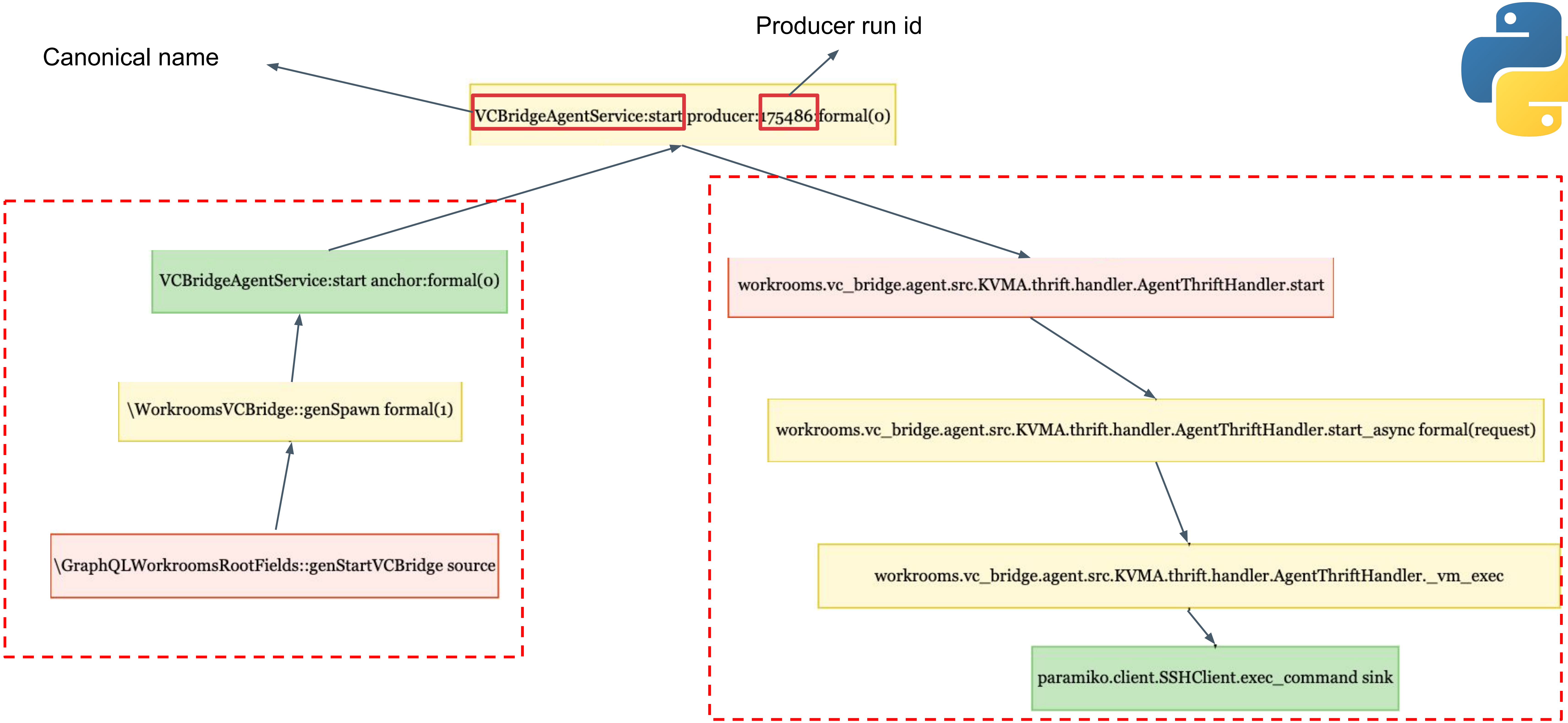
- Analyze the codebase with/without the pull request
- Check for findings
- new?
  - High confidence -> auto-comment
  - Lower confidence -> security oncall

Example finding

Example - Remote command execution



Example - Remote Command Execution



Canonical name

Producer run id

VCBridgeAgentService:start producer:175486:formal(0)

VCBridgeAgentService:start anchor:formal(0)

\WorkroomsVCBridge::genSpawn formal(1)

\GraphQLWorkroomsRootFields::genStartVCBridge source

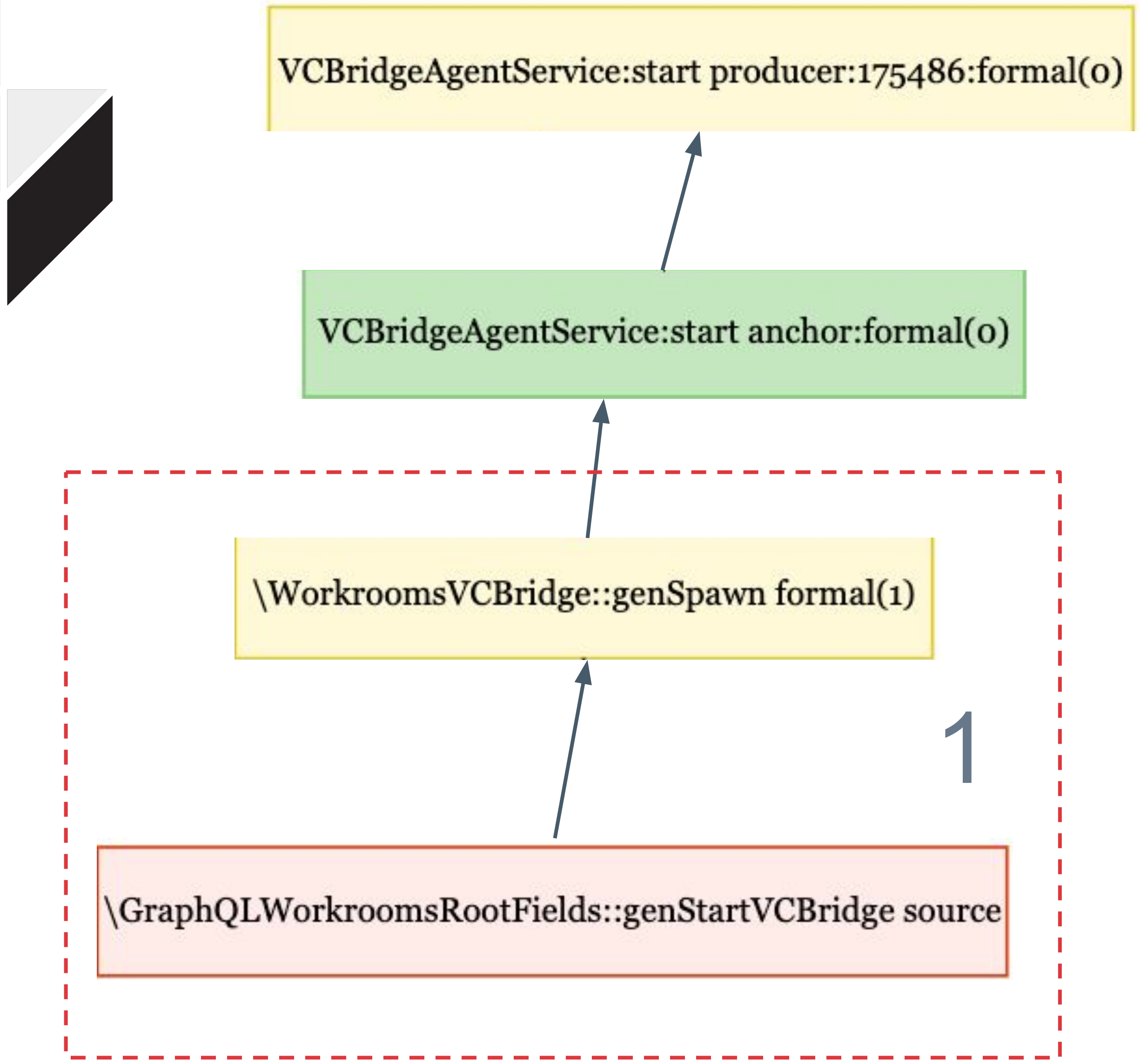
workrooms.vc_bridge.agent.src.KVMA.thrift.handler.AgentThriftHandler.start

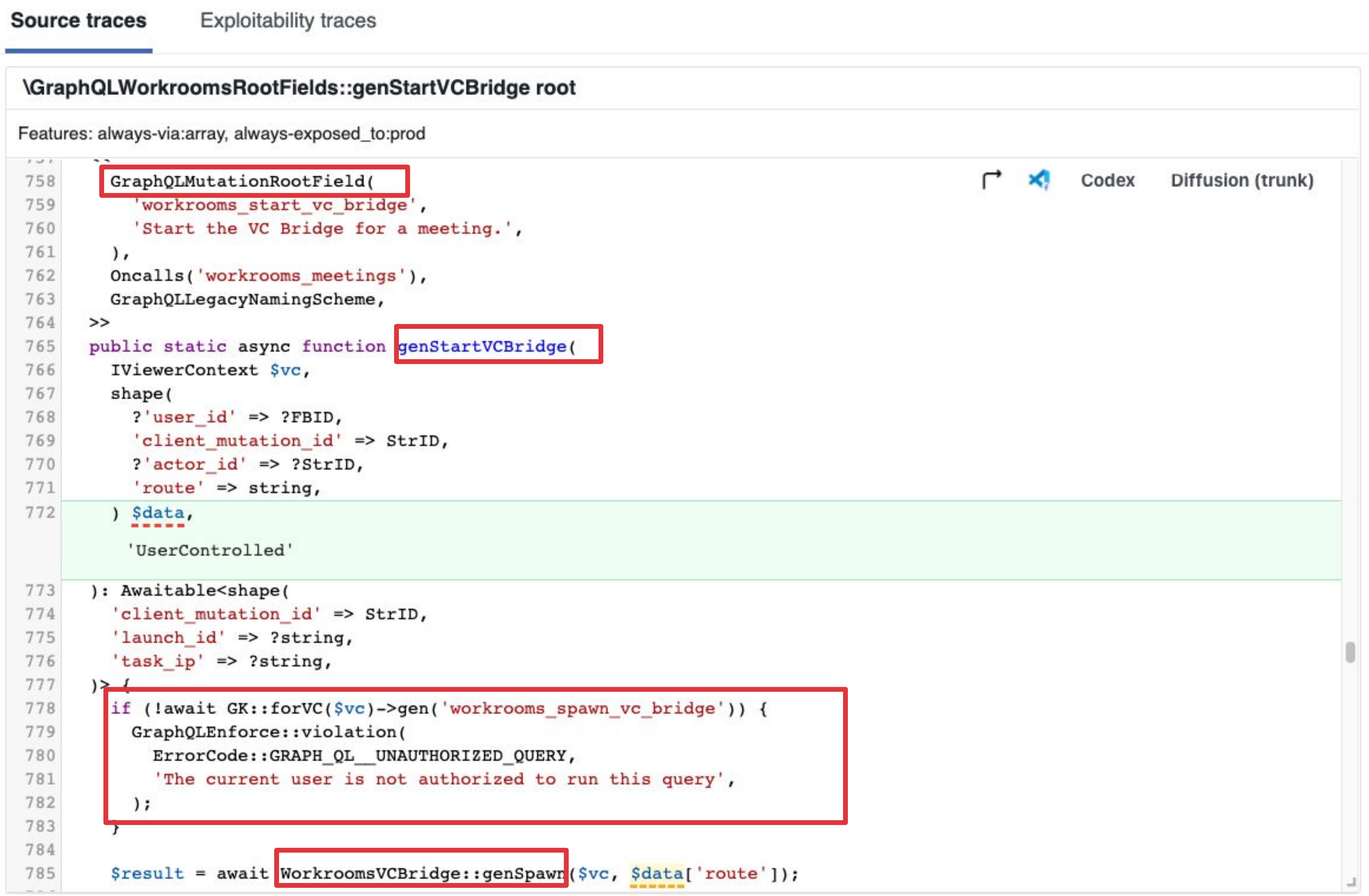
workrooms.vc_bridge.agent.src.KVMA.thrift.handler.AgentThriftHandler.start_async formal(request)

workrooms.vc_bridge.agent.src.KVMA.thrift.handler.AgentThriftHandler._vm_exec

paramiko.client.SSHClient.exec_command sink

Example - Remote command execution


VCBridgeAgentService:start producer:175486:formal(0)
VCBridgeAgentService:start anchor:formal(0)
\WorkroomsVCBridge::genSpawn formal(1)
1
\GraphQLWorkroomsRootFields::genStartVCBridge source



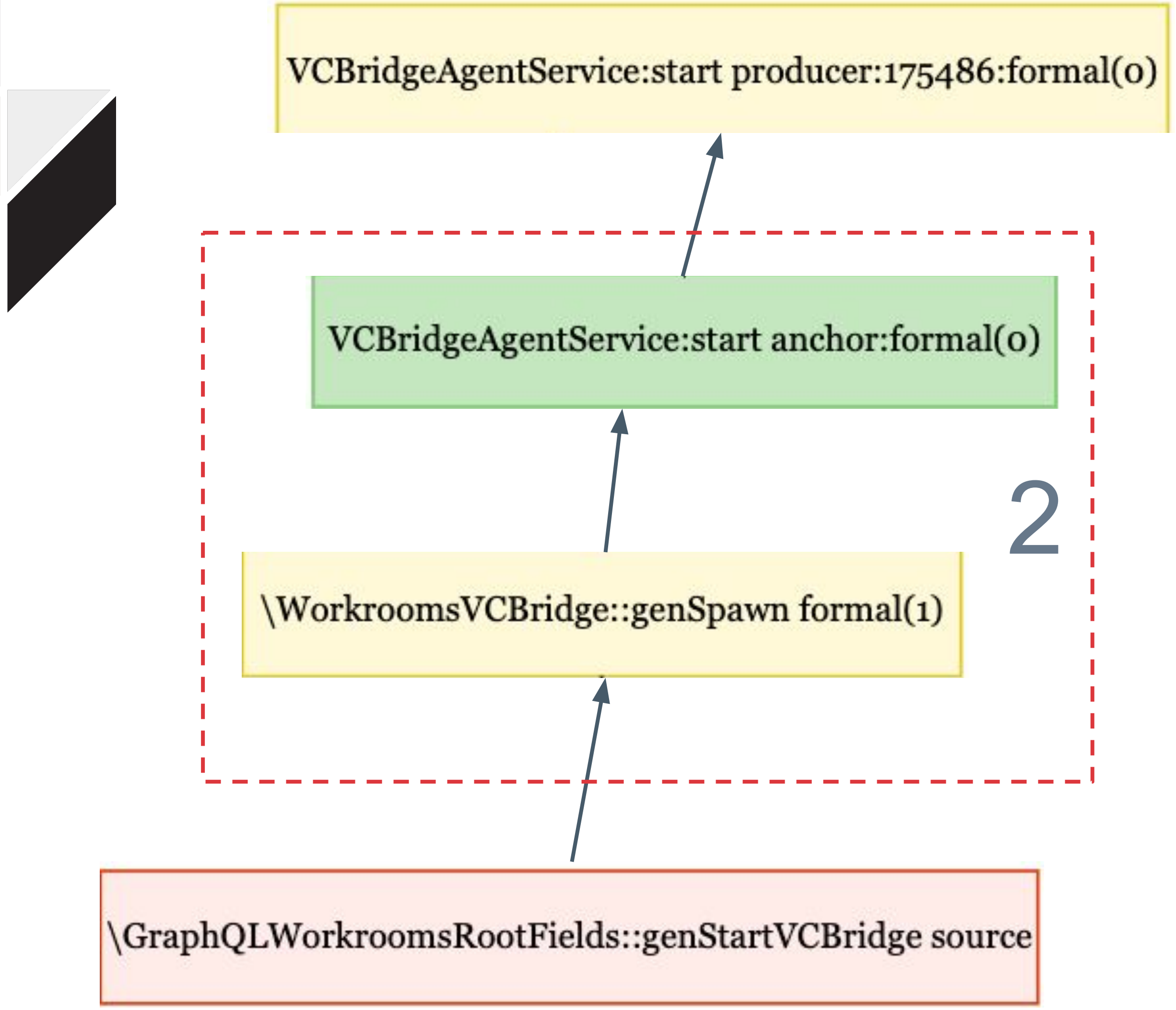
\GraphQLWorkroomsRootFields::genStartVCBridge root

Features: always-via:array, always-exposed_to:prod

```
  GraphQLMutationRootField(
    'workrooms_start_vc_bridge',
    'Start the VC Bridge for a meeting.',
  ),
  Oncalls('workrooms_meetings'),
  GraphQLLegacyNamingScheme,
>>
public static async function genStartVCBridge(
  IViewerContext $vc,
  shape(
    ?'user_id' => ?FBID,
    'client_mutation_id' => StrID,
    ?'actor_id' => ?StrID,
    'route' => string,
  ) $data,
    'UserControlled'
): Awaitable<shape(
  'client_mutation_id' => StrID,
  'launch_id' => ?string,
  'task_ip' => ?string,
)> {
  if (!await GK::forVC($vc)->gen('workrooms_spawn_vc_bridge')) {
    GraphQLEnforce::violation(
      ErrorCode::GRAPH_QL__UNAUTHORIZED_QUERY,
      'The current user is not authorized to run this query',
    );
  }

  $result = await WorkroomsVCBridge::genSpawn($vc, $data['route']);
```

Primary run issue 2127615841

Example - Remote command execution



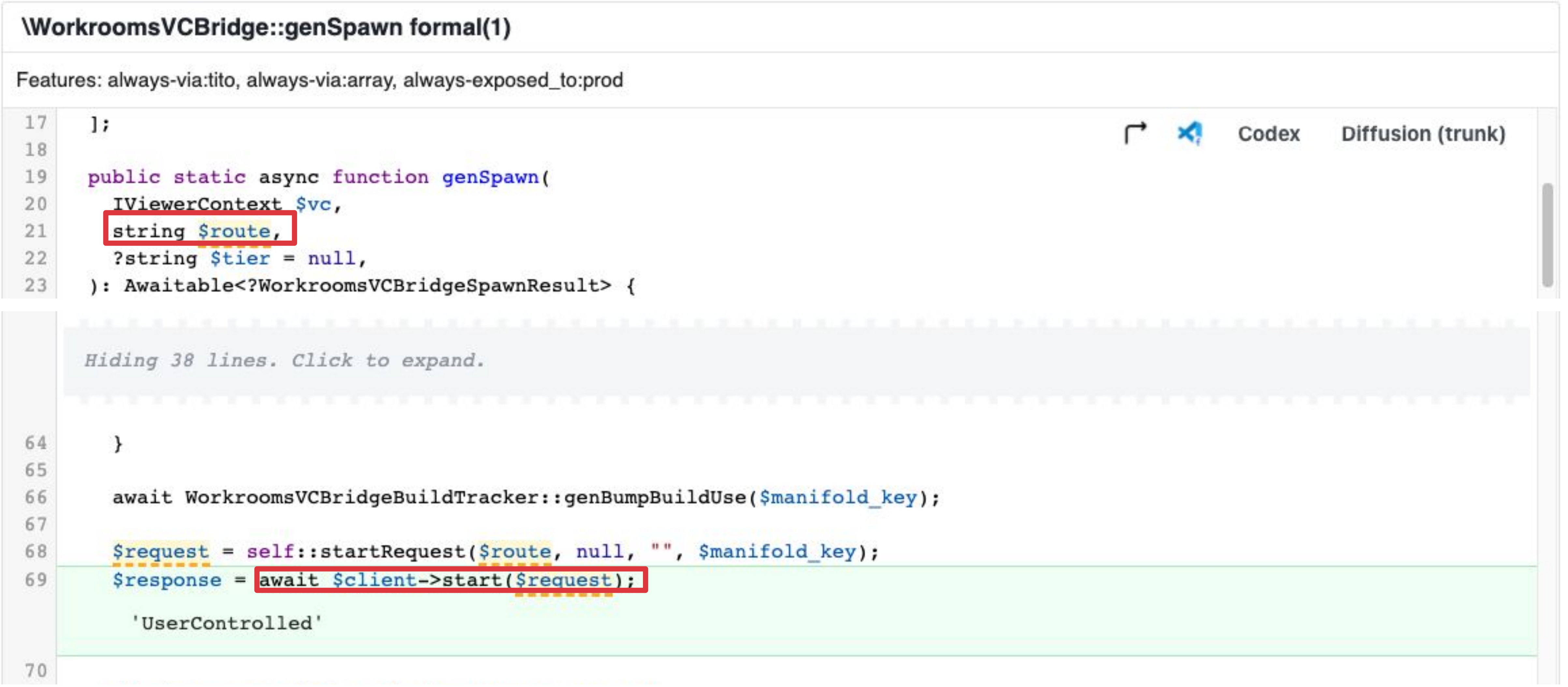
VCBridgeAgentService:start producer:175486:formal(0)

VCBridgeAgentService:start anchor:formal(0)

\WorkroomsVCBridge::genSpawn formal(1)

2

\GraphQLWorkroomsRootFields::genStartVCBridge source



\WorkroomsVCBridge::genSpawn formal(1)

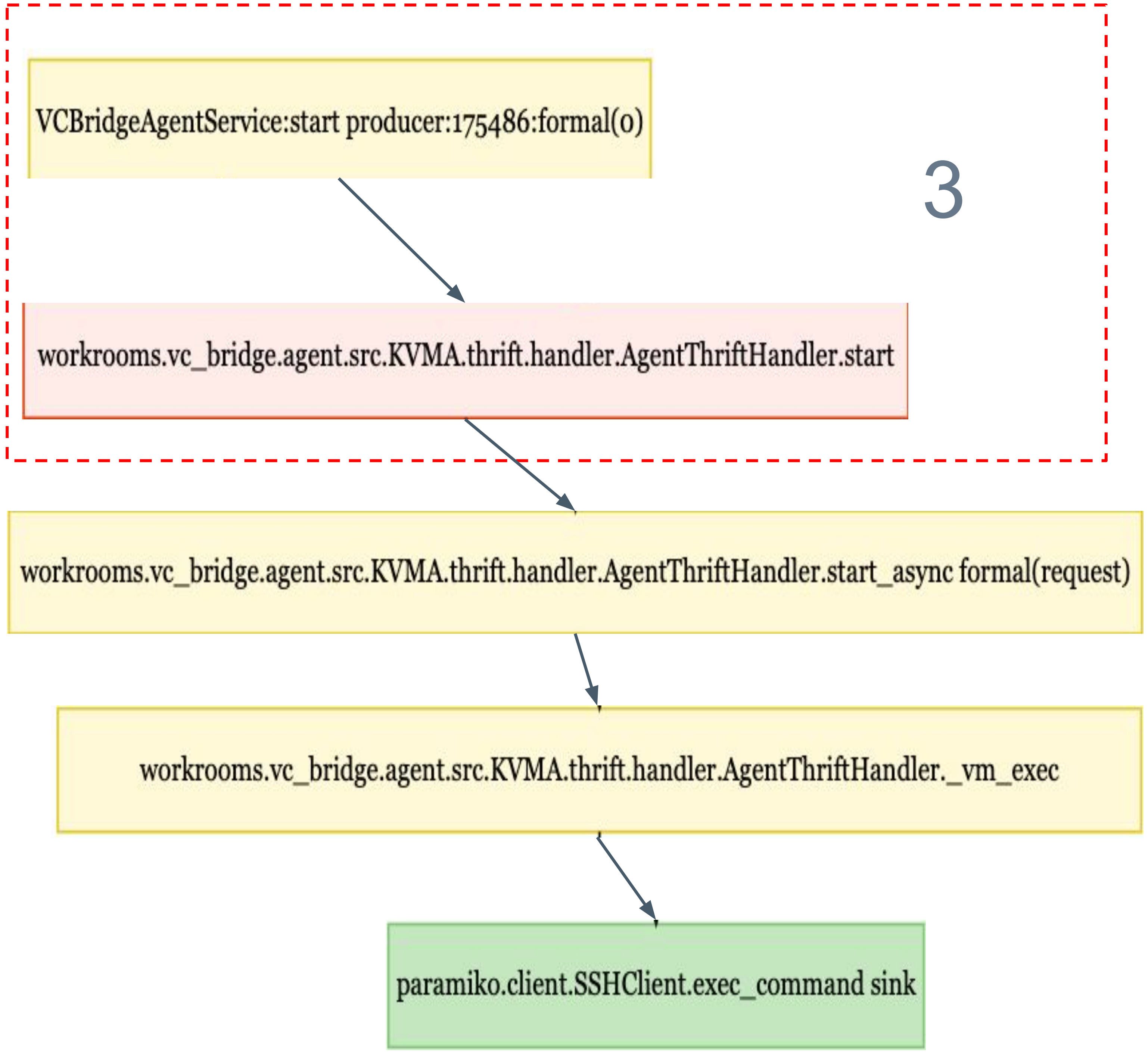
Features: always-via:tito, always-via:array, always-exposed_to:prod

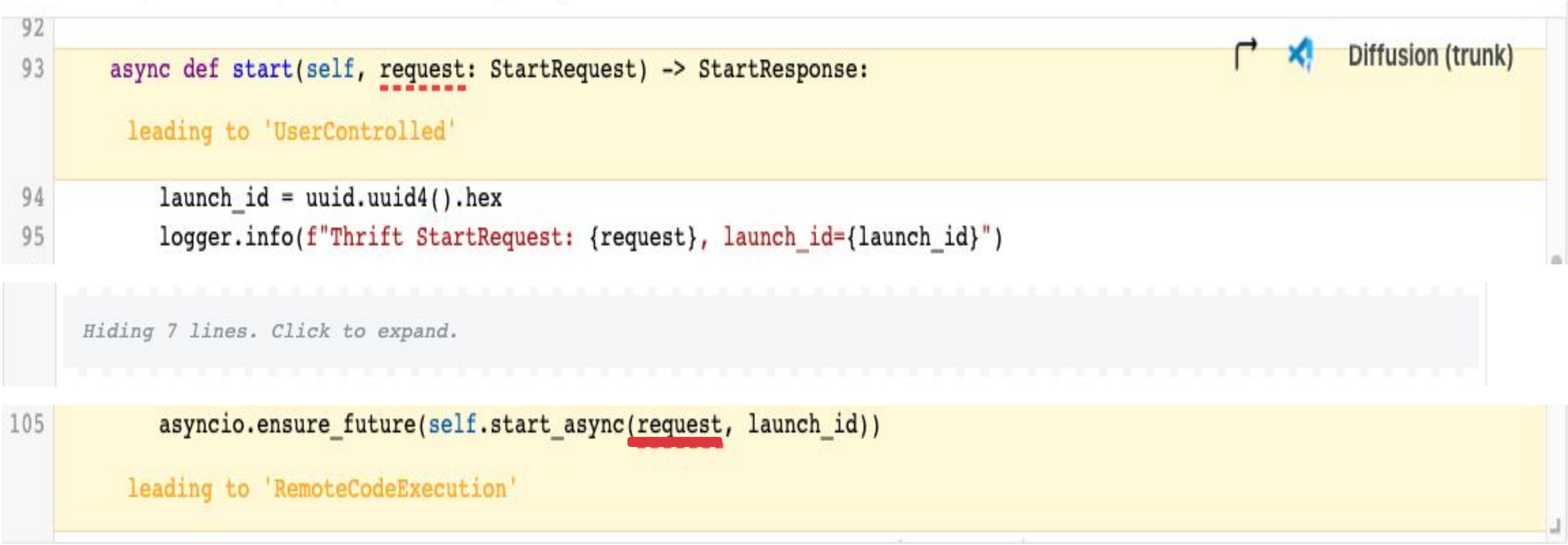
```
];

public static async function genSpawn(
  IViewerContext $vc,
  string $route,
  ?string $tier = null,
): Awaitable<?WorkroomsVCBridgeSpawnResult> {
```

Hiding 38 lines. Click to expand.

```
  }

  await WorkroomsVCBridgeBuildTracker::genBumpBuildUse($manifold_key);

  $request = self::startRequest($route, null, "", $manifold_key);
  $response = await $client->start($request);
   'UserControlled'
```

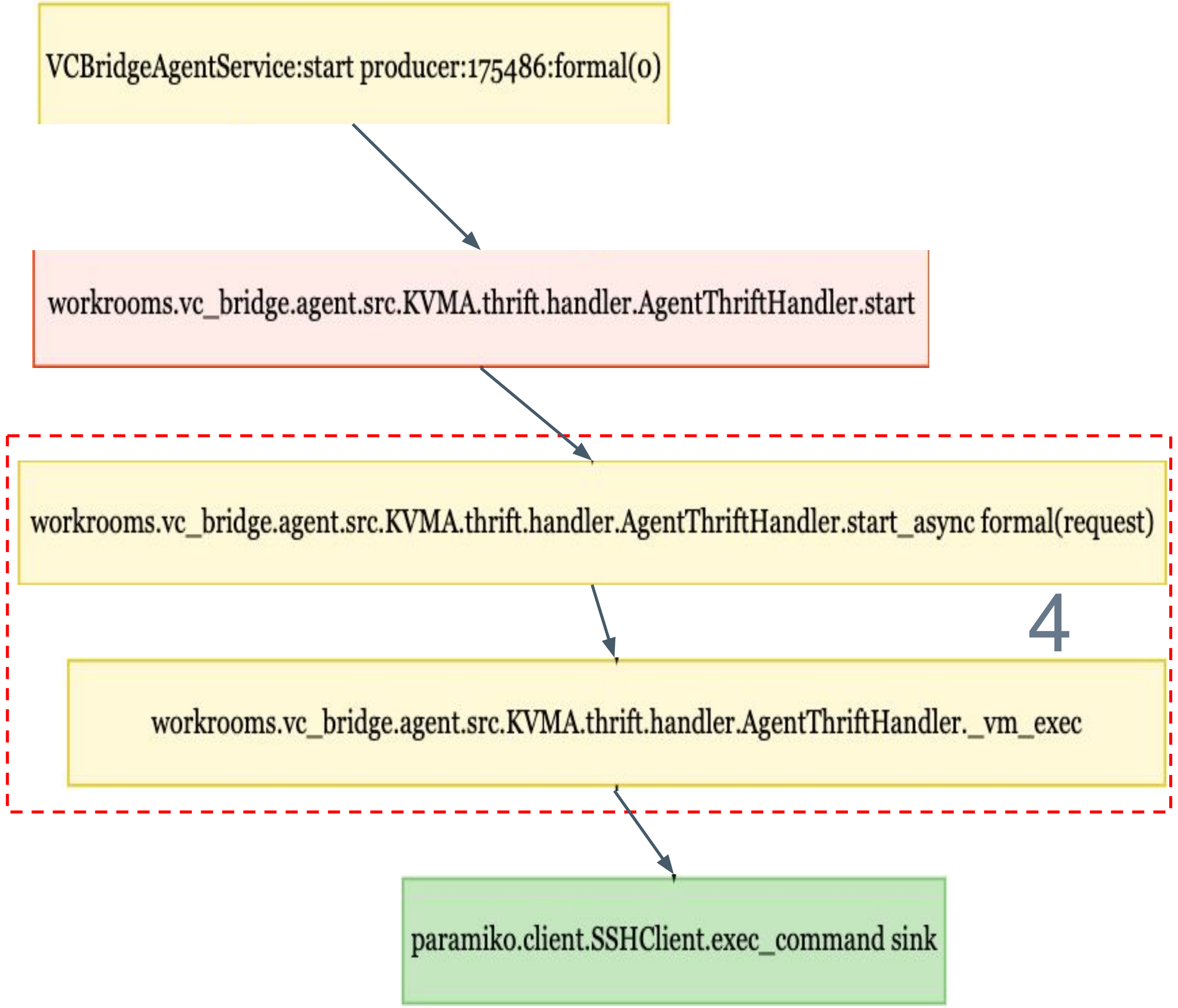
Example - Remote command execution

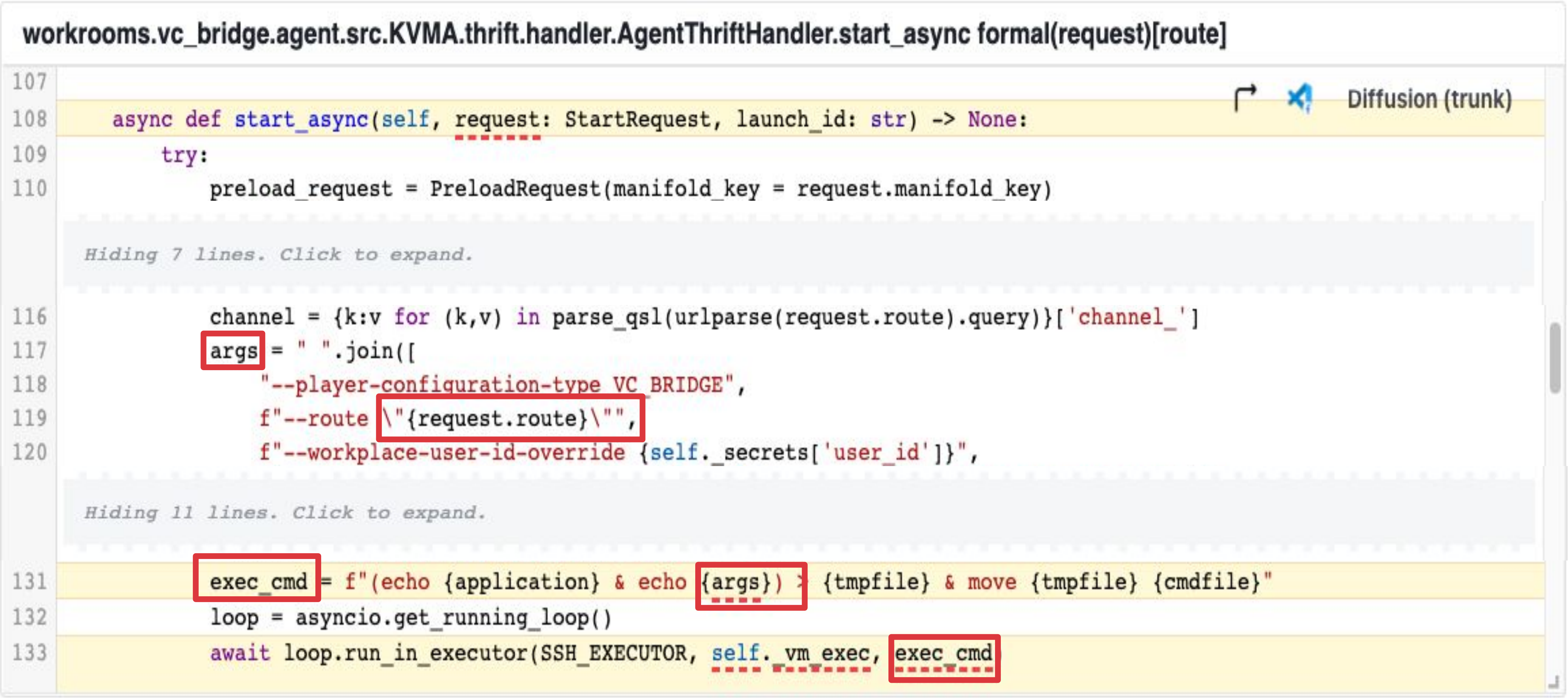

VCBridgeAgentService:start producer:175486:formal(0)
3
workrooms.vc_bridge.agent.src.KVMA.thrift.handler.AgentThriftHandler.start
workrooms.vc_bridge.agent.src.KVMA.thrift.handler.AgentThriftHandler.start_async formal(request)
workrooms.vc_bridge.agent.src.KVMA.thrift.handler.AgentThriftHandler._vm_exec
paramiko.client.SSHClient.exec_command sink



```
async def start(self, request: StartRequest) -> StartResponse:
 leading to 'UserControlled'
    launch_id = uuid.uuid4().hex
    logger.info(f"Thrift StartRequest: {request}, launch_id={launch_id}")
Hiding 7 lines. Click to expand.
    asyncio.ensure_future(self.start_async(request, launch_id))
 leading to 'RemoteCodeExecution'
```

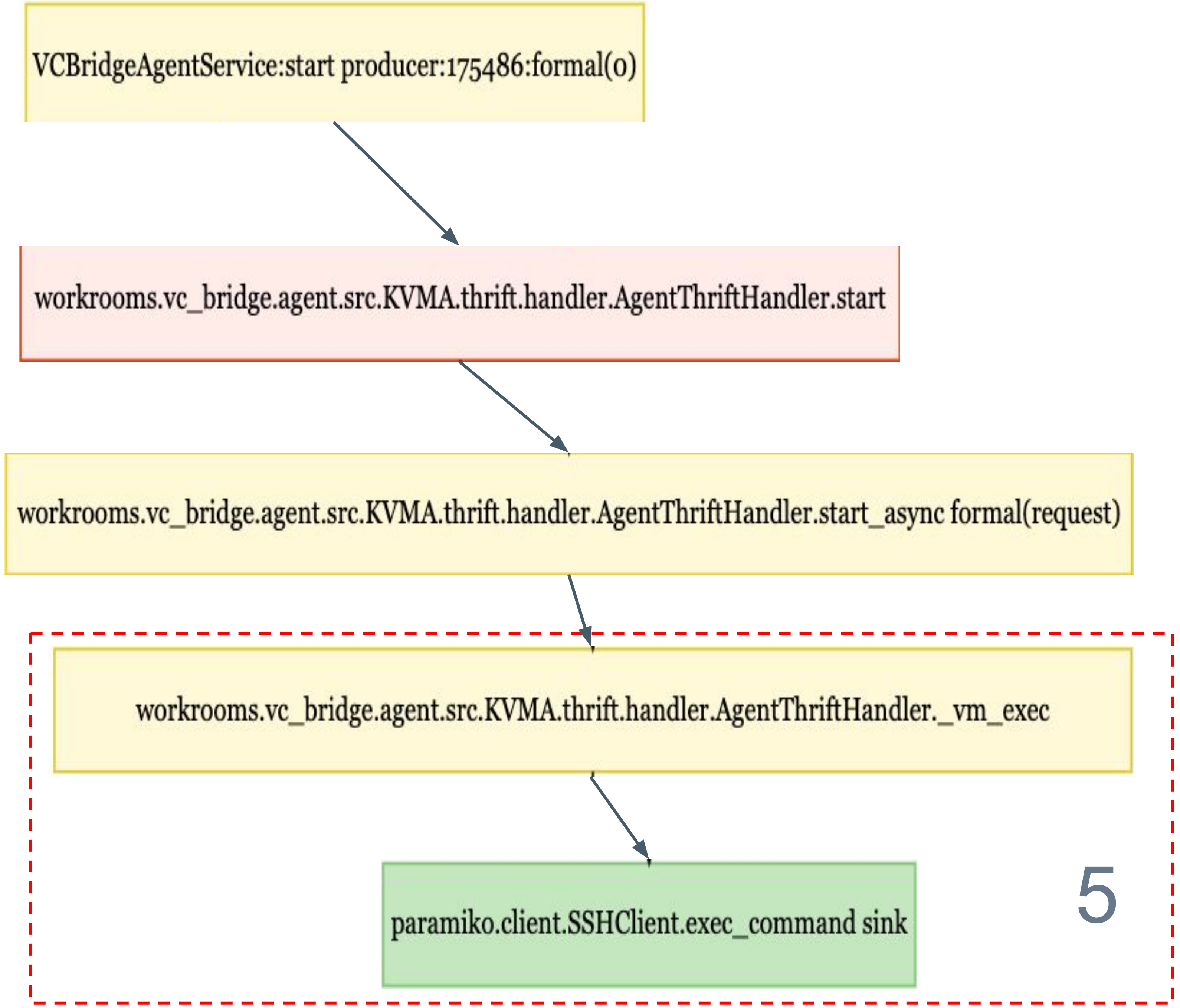
Example - Remote command execution

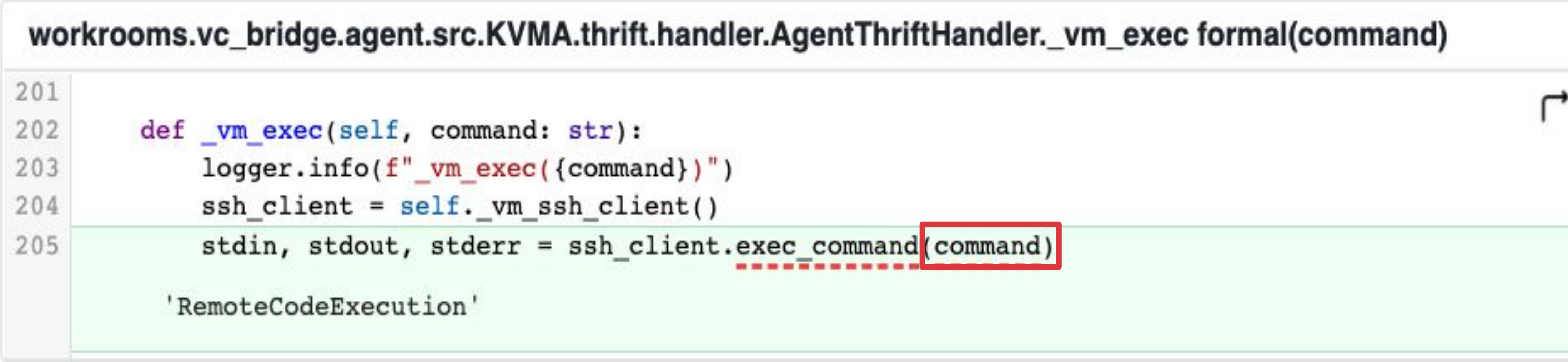

VCBridgeAgentService:start producer:175486:formal(0)
workrooms.vc_bridge.agent.src.KVMA.thrift.handler.AgentThriftHandler.start
workrooms.vc_bridge.agent.src.KVMA.thrift.handler.AgentThriftHandler.start_async formal(request)
4
workrooms.vc_bridge.agent.src.KVMA.thrift.handler.AgentThriftHandler._vm_exec
paramiko.client.SSHClient.exec_command sink


workrooms.vc_bridge.agent.src.KVMA.thrift.handler.AgentThriftHandler.start_async formal(request)[route]

```
    async def start_async(self, request: StartRequest, launch_id: str) -> None:
        try:
            preload_request = PreloadRequest(manifold_key = request.manifold_key)
 Hiding 7 lines. Click to expand.
            channel = {k:v for (k,v) in parse_qsl(urlparse(request.route).query)}['channel_']
            args = " ".join([
                "--player-configuration-type VC_BRIDGE",
                f"--route \"{request.route}\"",
                f"--workplace-user-id-override {self._secrets['user_id']}",
 Hiding 11 lines. Click to expand.
            exec_cmd = f"(echo {application} & echo {args}) > {tmpfile} & move {tmpfile} {cmdfile}"
            loop = asyncio.get_running_loop()
            await loop.run_in_executor(SSH_EXECUTOR, self._vm_exec, exec_cmd
```

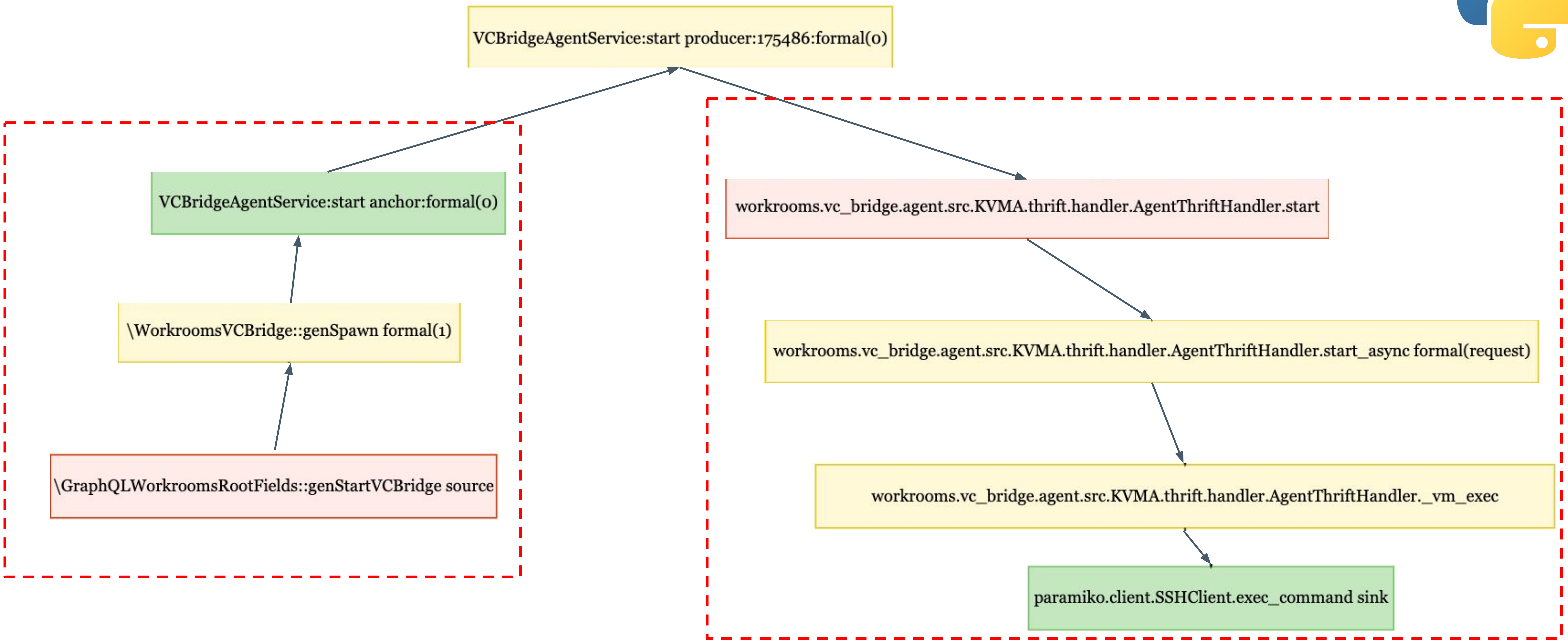
Example - Remote command execution


VCBridgeAgentService:start producer:175486:formal(0)
workrooms.vc_bridge.agent.src.KVMA.thrift.handler.AgentThriftHandler.start
workrooms.vc_bridge.agent.src.KVMA.thrift.handler.AgentThriftHandler.start_async formal(request)
workrooms.vc_bridge.agent.src.KVMA.thrift.handler.AgentThriftHandler._vm_exec
paramiko.client.SSHClient.exec_command sink
5


workrooms.vc_bridge.agent.src.KVMA.thrift.handler.AgentThriftHandler._vm_exec formal(command)

```

    def _vm_exec(self, command: str):
        logger.info(f"_vm_exec({command})")
        ssh_client = self._vm_ssh_client()
        stdin, stdout, stderr = ssh_client.exec_command(command)

  'RemoteCodeExecution'
```

Example - Remote Command Execution



VCBridgeAgentService:start producer:175486:formal(0)

VCBridgeAgentService:start anchor:formal(0)

\WorkroomsVCBridge::genSpawn formal(1)

\GraphQLWorkroomsRootFields::genStartVCBridge source

workrooms.vc_bridge.agent.src.KVMA.thrift.handler.AgentThriftHandler.start

workrooms.vc_bridge.agent.src.KVMA.thrift.handler.AgentThriftHandler.start_async formal(request)

workrooms.vc_bridge.agent.src.KVMA.thrift.handler.AgentThriftHandler._vm_exec

paramiko.client.SSHClient.exec_command sink

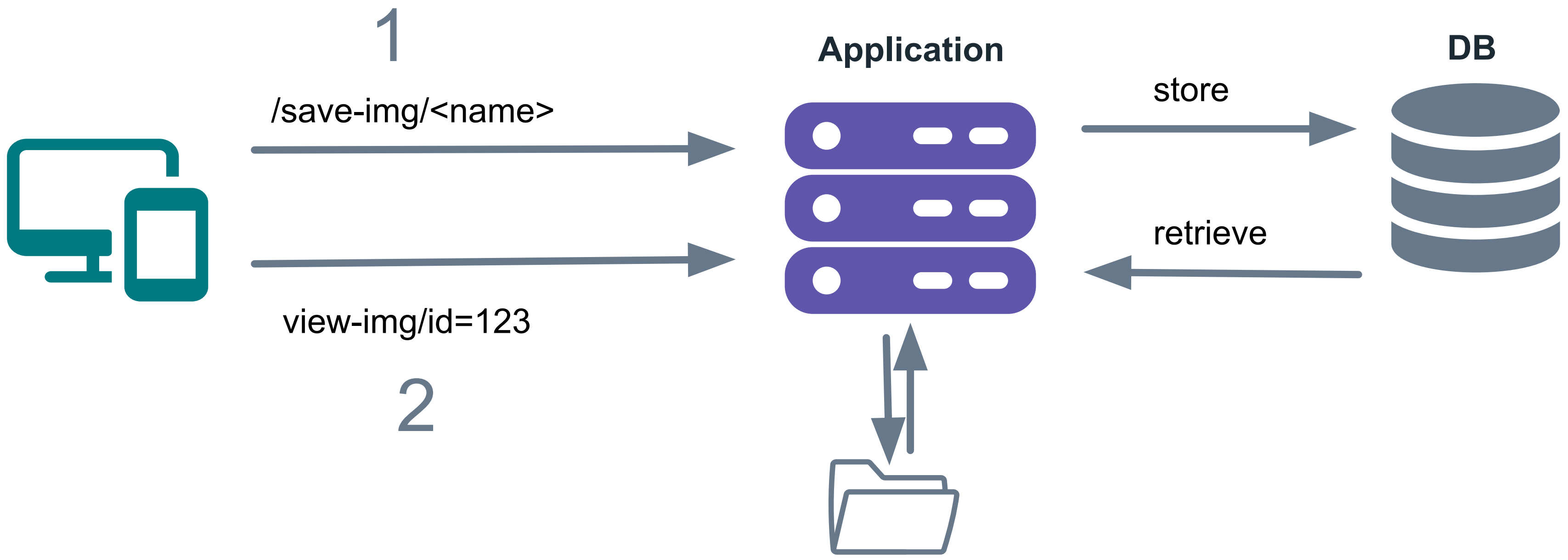
RCE in 2021 ლ(ò_óˇ)

Challenges and improvements areas

- False positives due to sanitization/validation in one language
- Simplify the creation of connection points
- Simplifying complex and long traces for security and software engineers
- Fix ownership

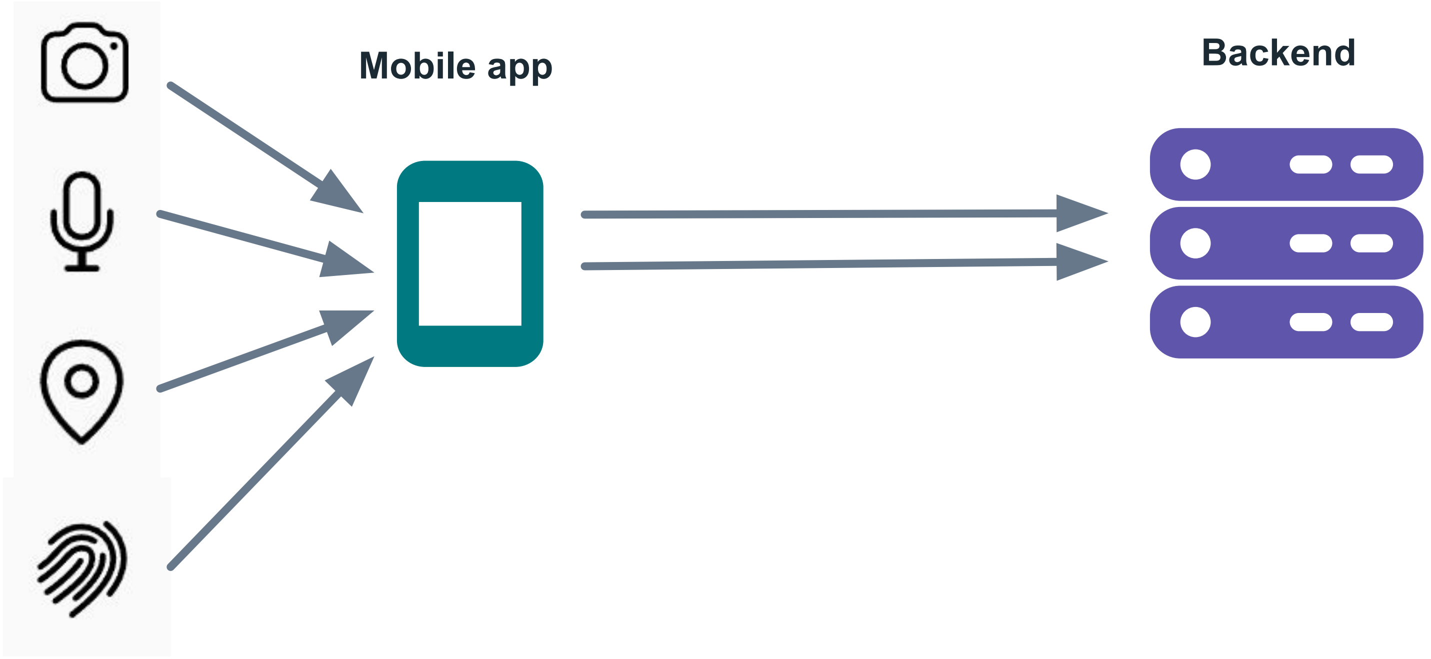
Looking forward

- Research standardise the taint-flow summaries - let’s all speak the same language!
- Expand mobile attack surface
  - Exported components
  - Requests from Backend
- Backend storages



Looking forward

- Privacy-relevant flows
  - Better understanding for clients (mobile applications)
  - Marking sensors e.g. GPS, fingerprint, camera as sources
  - Find flows that go to the backend?



TAKEAWAYS

Application security teams

- Scale through static analysis
  - **Pysa** for Python applications github.com/facebook/pyre-check
  - **Mariana-Trench** for Android/Java github.com/facebook/mariana-trench
- Go deeper with Cross-repo analysis

Security consultants

- Optimize your security reviews with our **open source configurations**
  - Pysa
  - Mariana trench
- Found more ways to get SQLi/RCE?
  - Contribute to our configurations!
- Want to see everything in action? **Come join us! (fb.com/careers)**

Static analysis researchers

- Research on tool-agnostic taint summaries
- Our tools are open source!
  - github.com/facebook/mariana-trench
  - github.com/facebook/pyre-check
  - github.com/facebook/sapp

Thank you

Dominik Gabi

Manuel Fahndrich

Otto Ebeling

David Molnar

Graham Bleaney

Jim O’Leary

Dan Gurfinkel

Chris Rohlf

Sinan Cepel

Questions?

THANK YOU FOR YOUR TIME