Static Detection of Second-Order Vulnerabilities in Web Applications

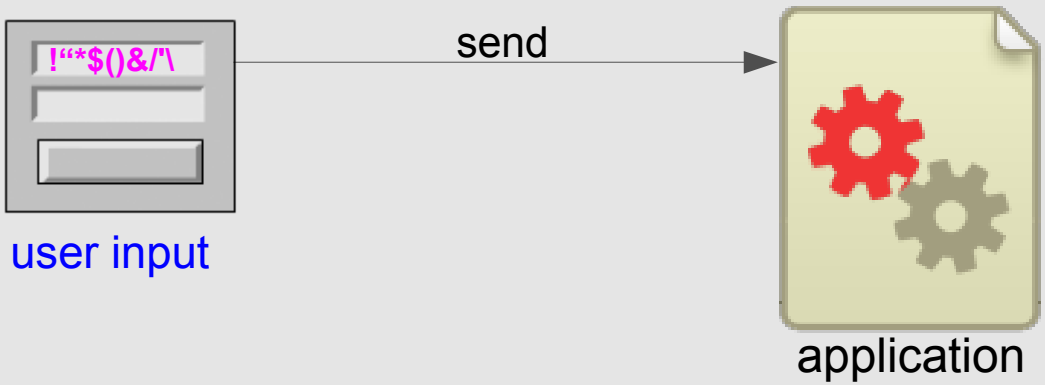
Johannes Dahse and Thorsten Holz
Ruhr-University Bochum

USENIX Security ’14, 20-22 August 2014, San Diego, CA, USA

„First-Order“ Vulnerabilities

- **SQL injection**

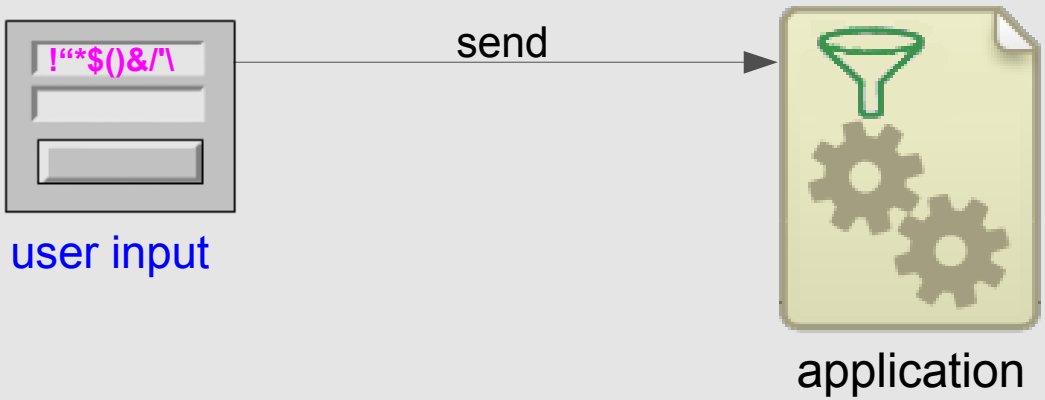
```
<?php
    $name = $_POST['name']; // ', 1), (version(), 1)-- -
    $sql = “INSERT INTO users VALUES ('$name', '$pwd')“;
    mysql_query($sql);
?>
```



Sanitization

- **SQL injection (prevented)**

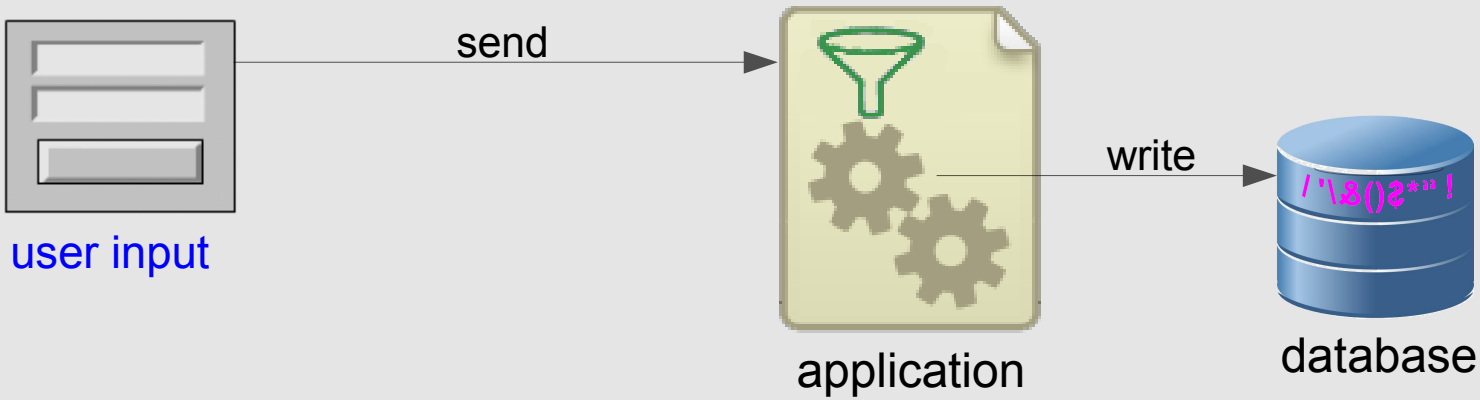
```
<?php
    $name = mysql_real_escape_string($_POST['name']);
    $sql = "INSERT INTO users VALUES ('$name', '$pwd')";
    mysql_query($sql);
?>
```



Second-Order Vulnerability (1)

- **Database Write**

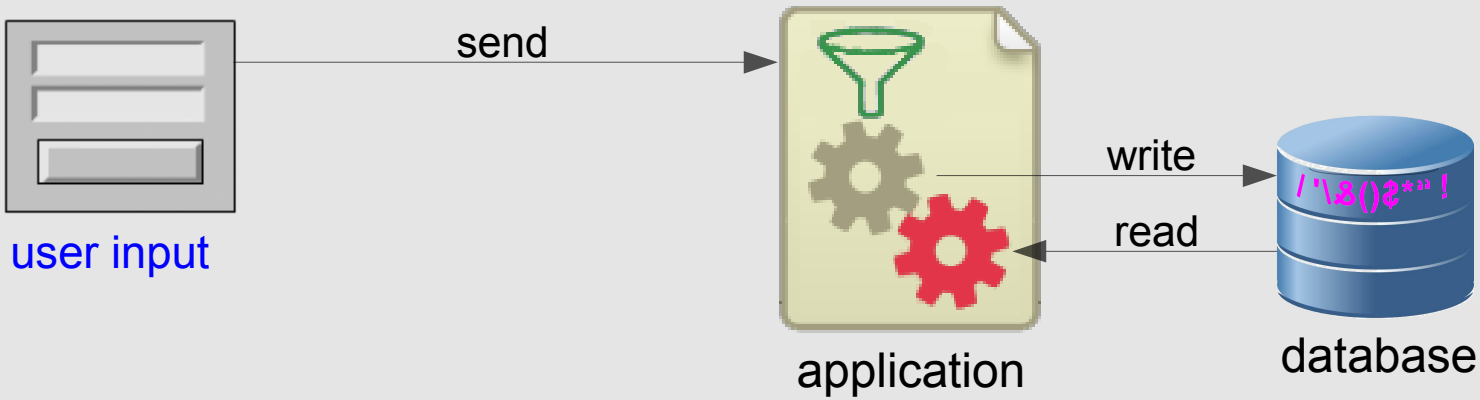
```
<?php
    $name = mysql_real_escape_string($_POST['name']);
    $sql = "INSERT INTO users VALUES ('$name', '$pwd')";
    mysql_query($sql);
?>
```



Second-Order Vulnerability (2)

- **Database Read**

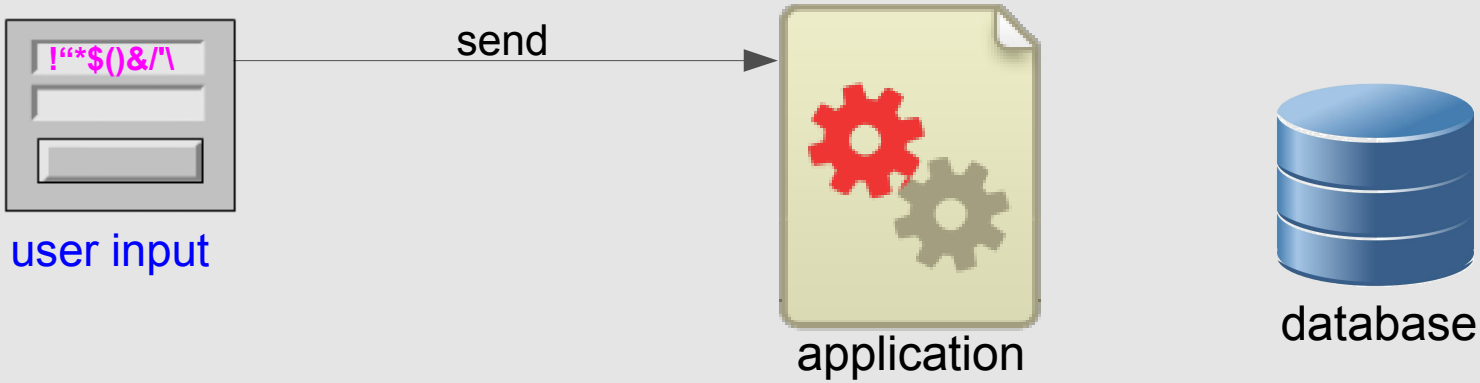
```
<?php
    $result = mysql_query('SELECT * FROM users');
    $row = mysql_fetch_assoc($result);
    echo $row['name'];
?>
```



Multi-Step Exploit (1)

- **First-Order SQL injection**

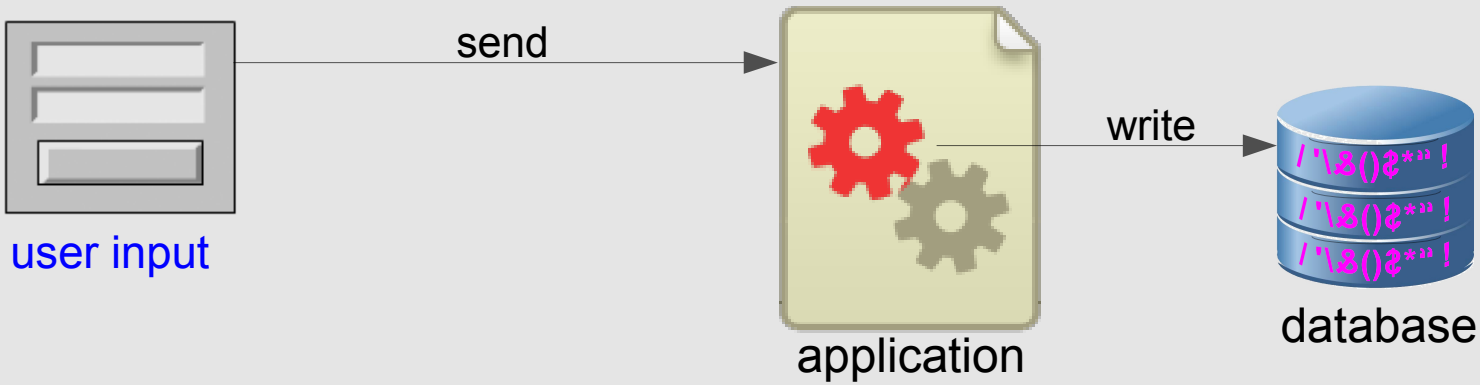
```
<?php
    $name = $_POST['name']; // ', 'payload')-- -
    $sql = "INSERT INTO users VALUES ('$name', '$pwd')";
    mysql_query($sql);
?>
```



Multi-Step Exploit (1)

- **Exploit First-Order SQL injection**

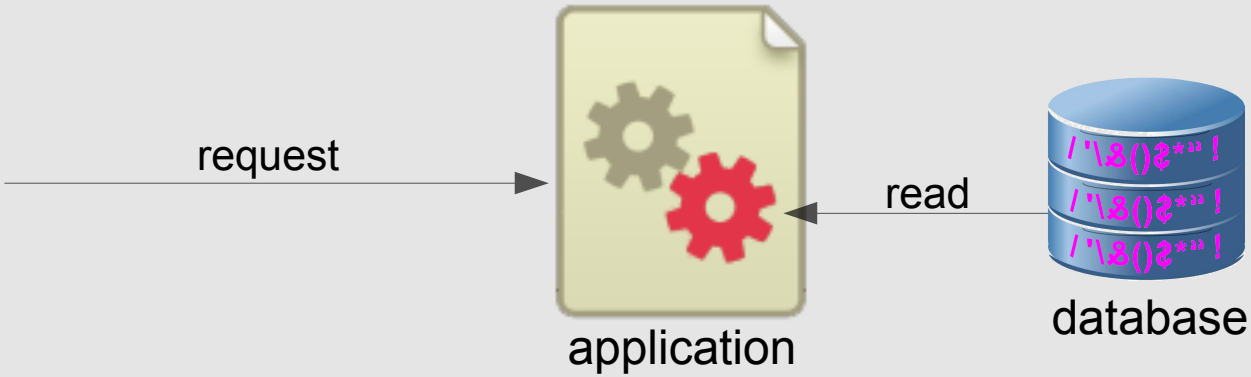
```
<?php
    $name = $_POST['name']; // ', 'payload')-- -
    $sql = "INSERT INTO users VALUES ('$name', '$pwd')";
    mysql_query($sql);
?>
```



Multi-Step Exploit (2)

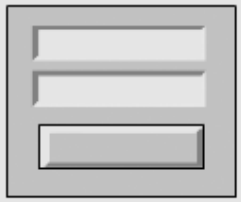
- **Second-Order Command Execution**

```
<?php
    $result = mysql_query('SELECT * FROM users');
    $row = mysql_fetch_assoc($result);
    system('htpasswd -b .htpasswd Admin '.$row['pwd']);
?>
```



Second-Order Vulnerabilities

User input



- $_GET
- $_POST
- $_COOKIE
- $_FILES
- $_SERVER

...

1.

Persistent Data Store (PDS)

- Databases
- File Names
- $_SESSION (File Content)

...

2.

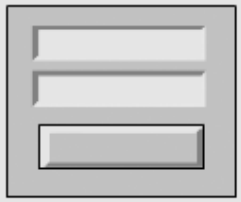
Sensitive Sink

- Cross-Site Scripting
- SQL Injection
- Code Execution
- File Inclusion
- File Disclosure

...

Second-Order Vulnerabilities

User input



- $_GET
- **$_POST**
- $_COOKIE
- $_FILES
- $_SERVER

...

1.

Persistent Data Store (PDS)

- **Databases**
- File Names
- $_SESSION (File Content)

...

2.

Sensitive Sink

- **Cross-Site Scripting**
- SQL Injection
- Code Execution
- File Inclusion
- File Disclosure

...

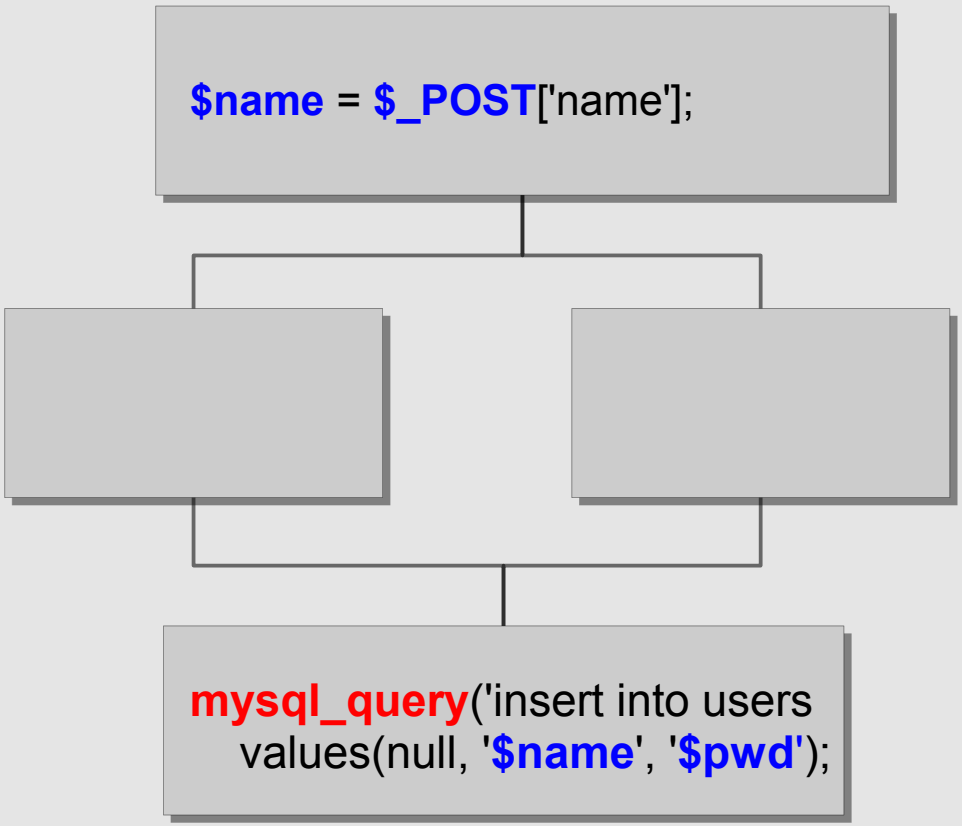
Our Approach

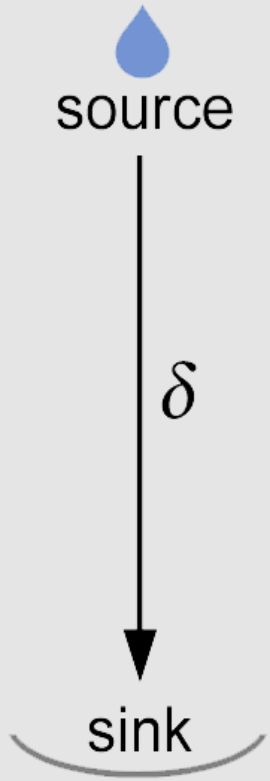
- Static Code Analysis (no access to environment)
- Analyze *writes* and *reads* to persistent data stores
- Connect input and output points at the end of the analysis to detect second-order and multi-step vulnerabilities

2. Implementation
(Overview)

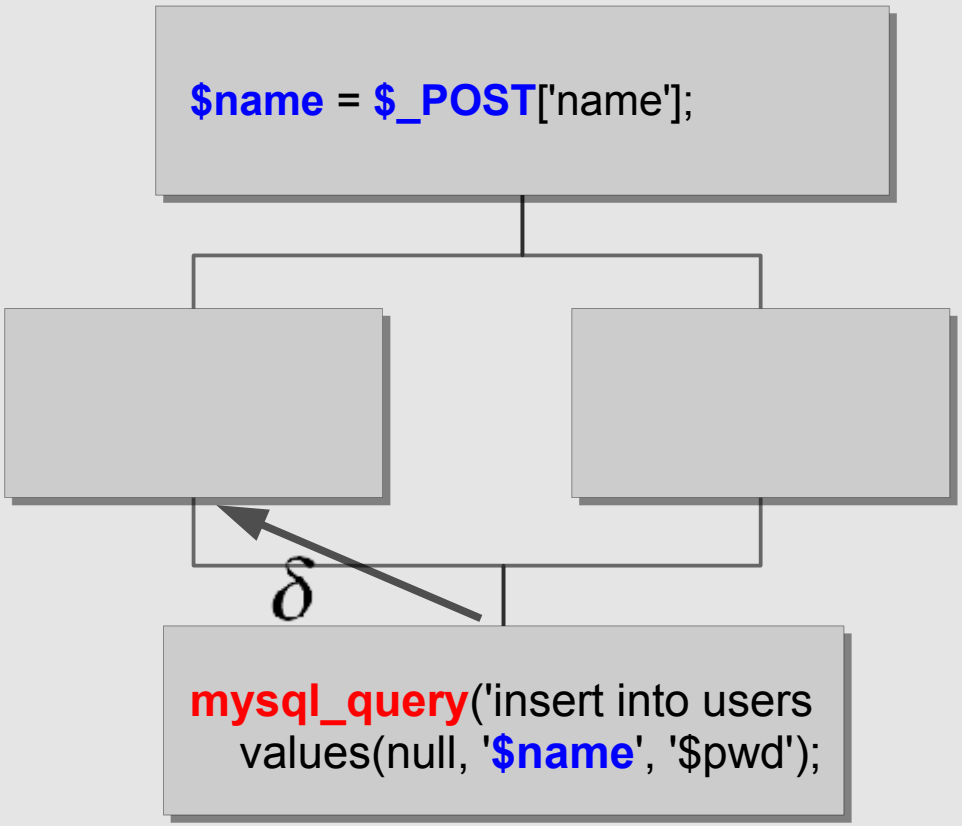
Source: http://rewalls.com

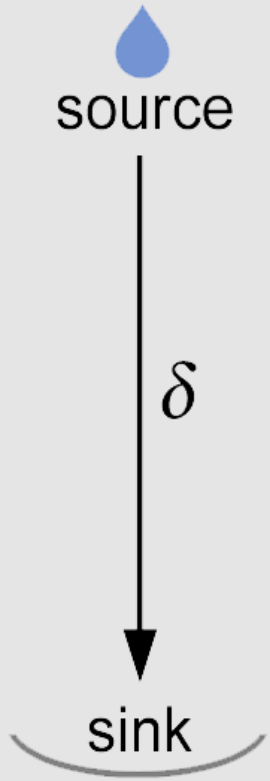
First-Order Taint Analysis



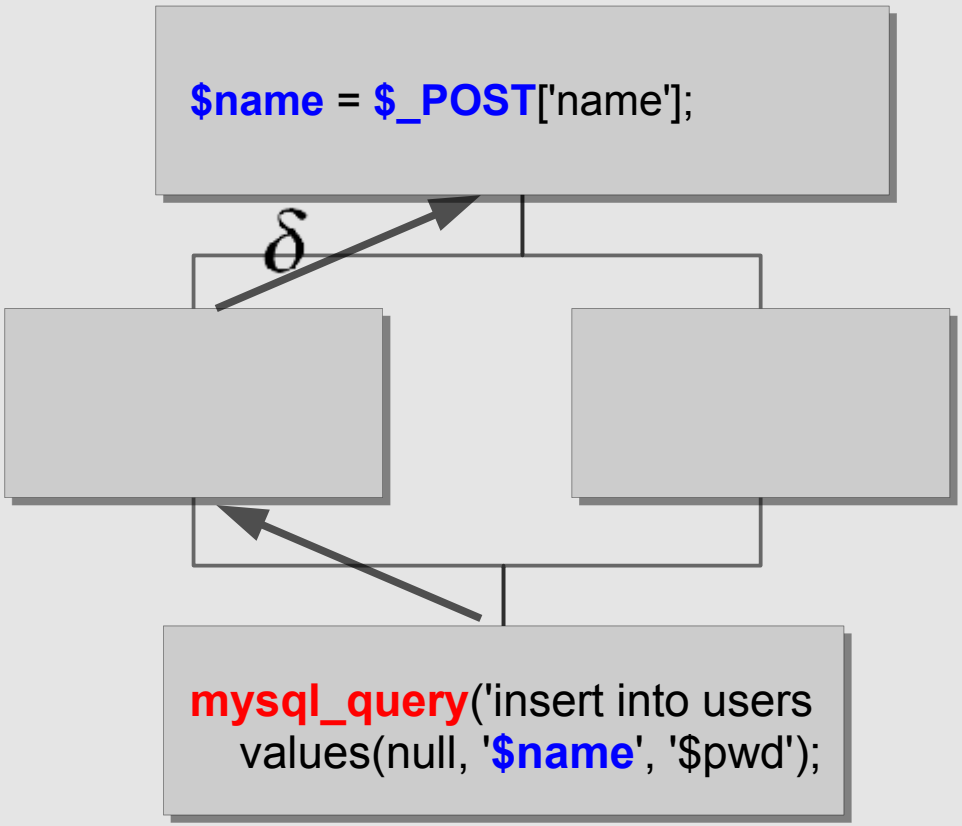


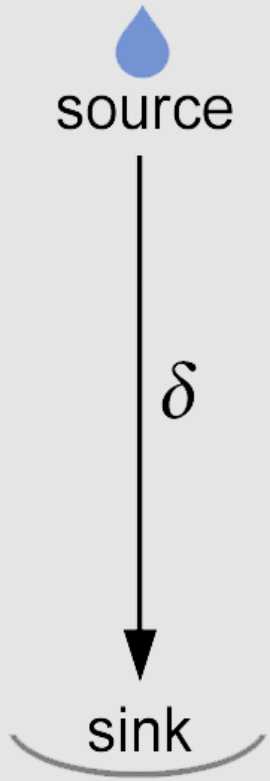
First-Order Taint Analysis



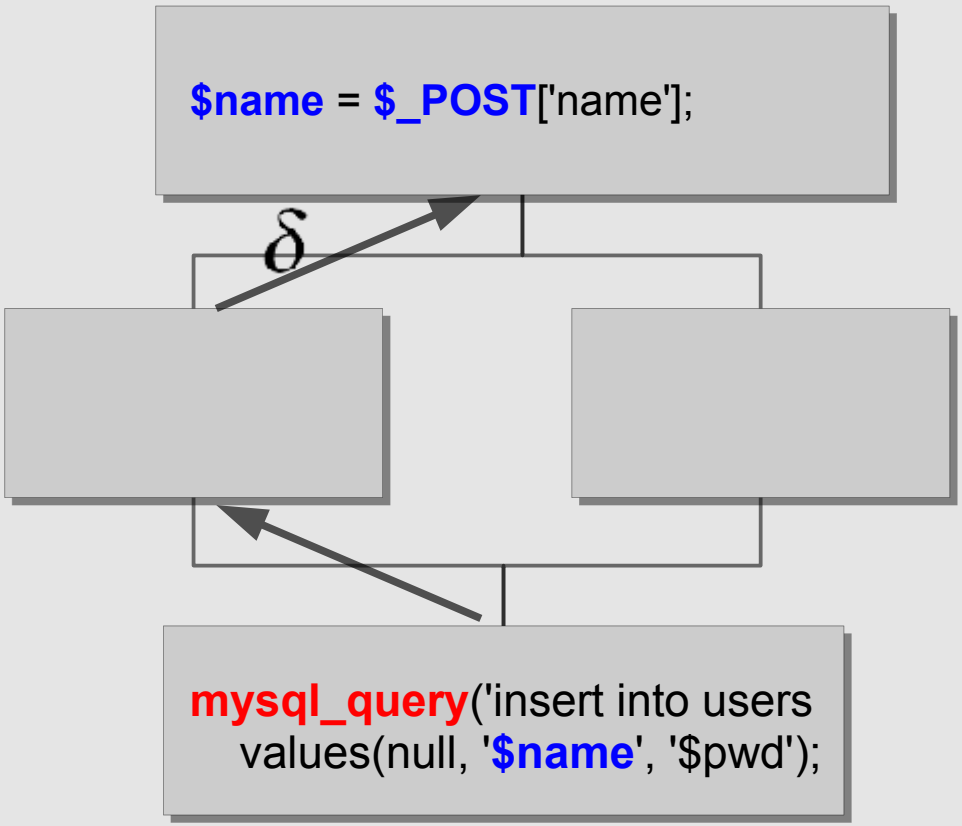


First-Order Taint Analysis

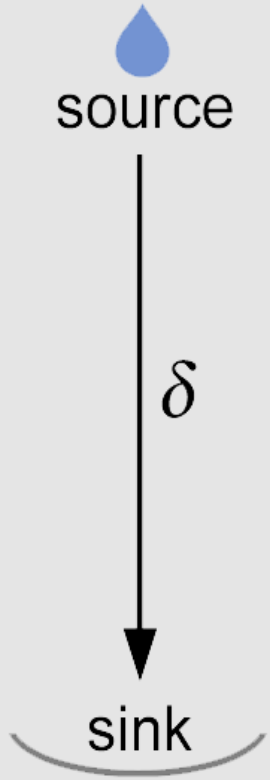




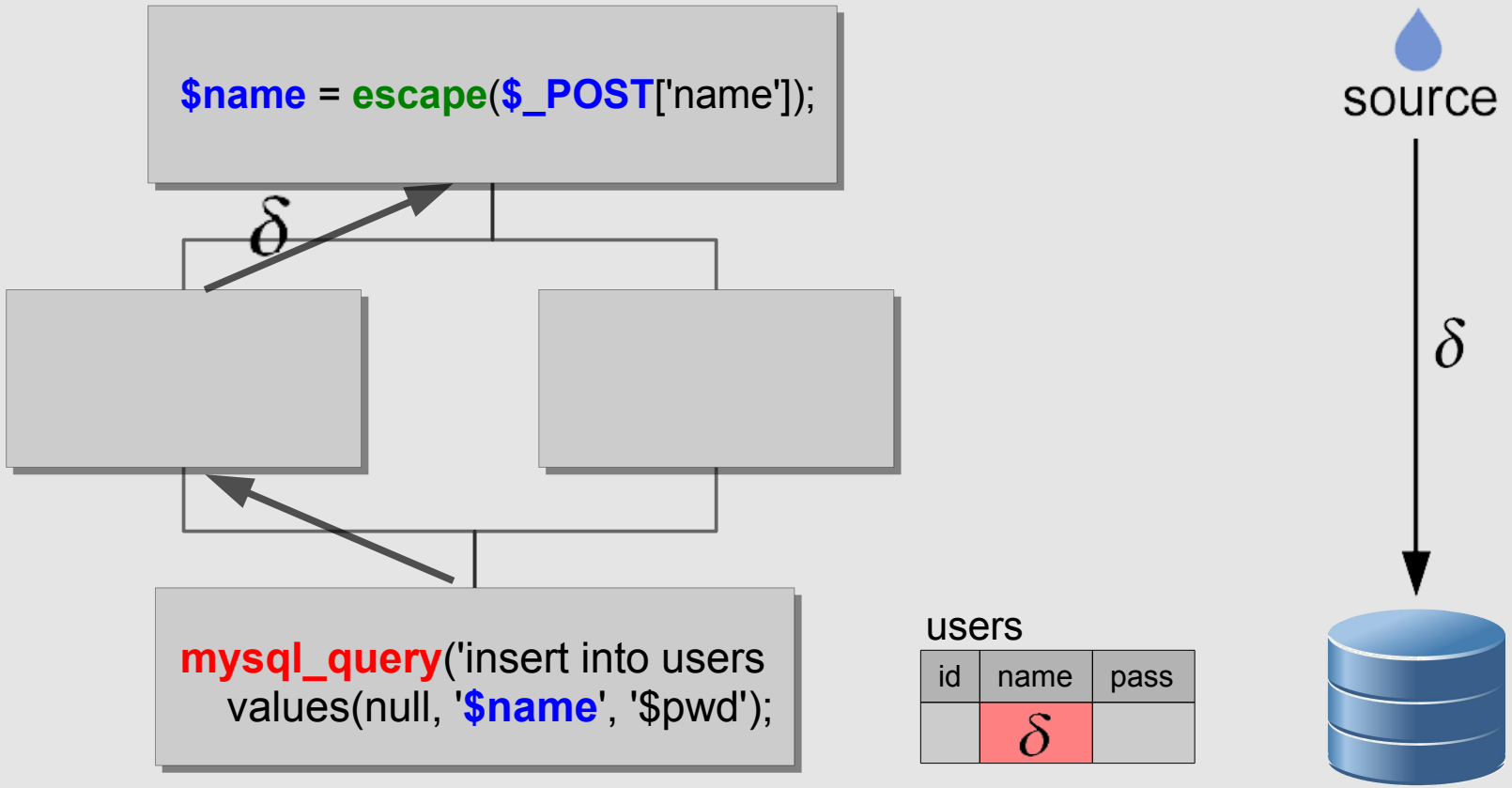
First-Order Taint Analysis



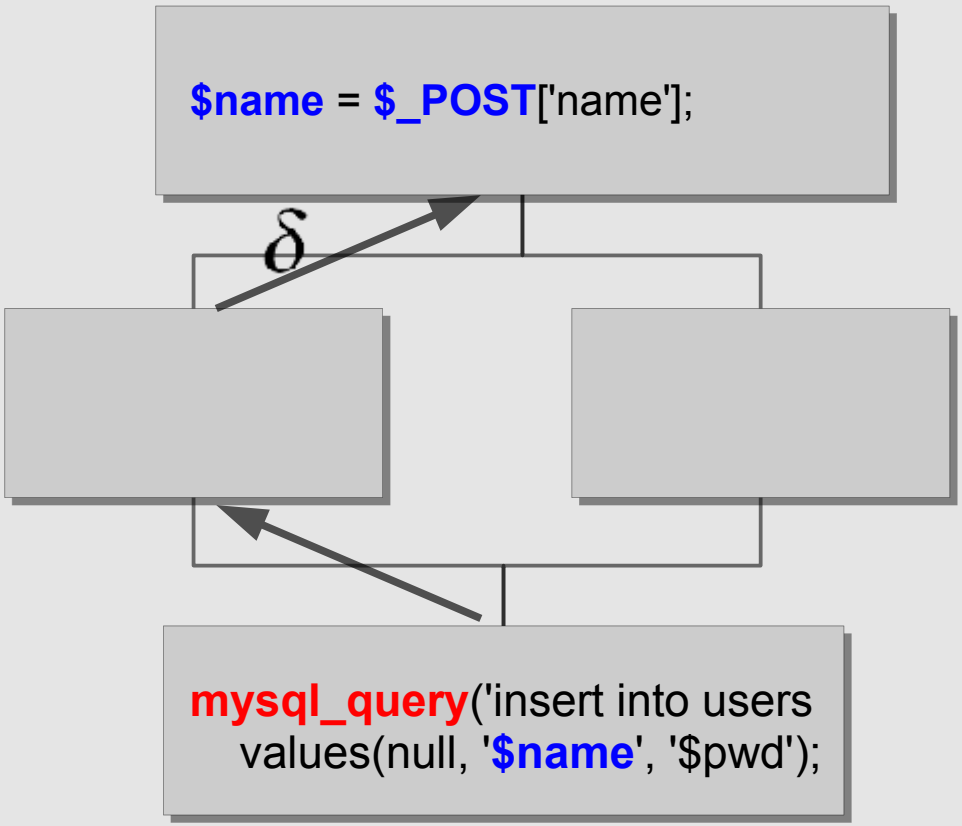
Vulnerability Report
POST[name]
SQLi



Second-Order Taint Analysis (write)



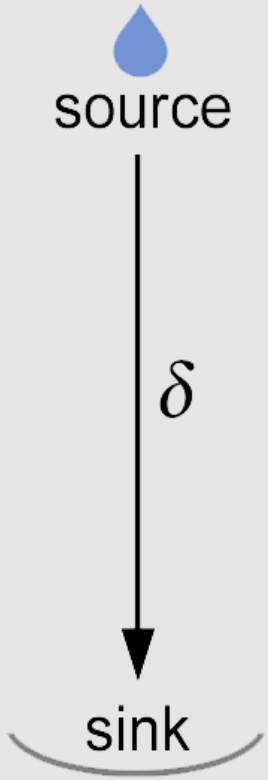
Multi-Step Taint Analysis (write)



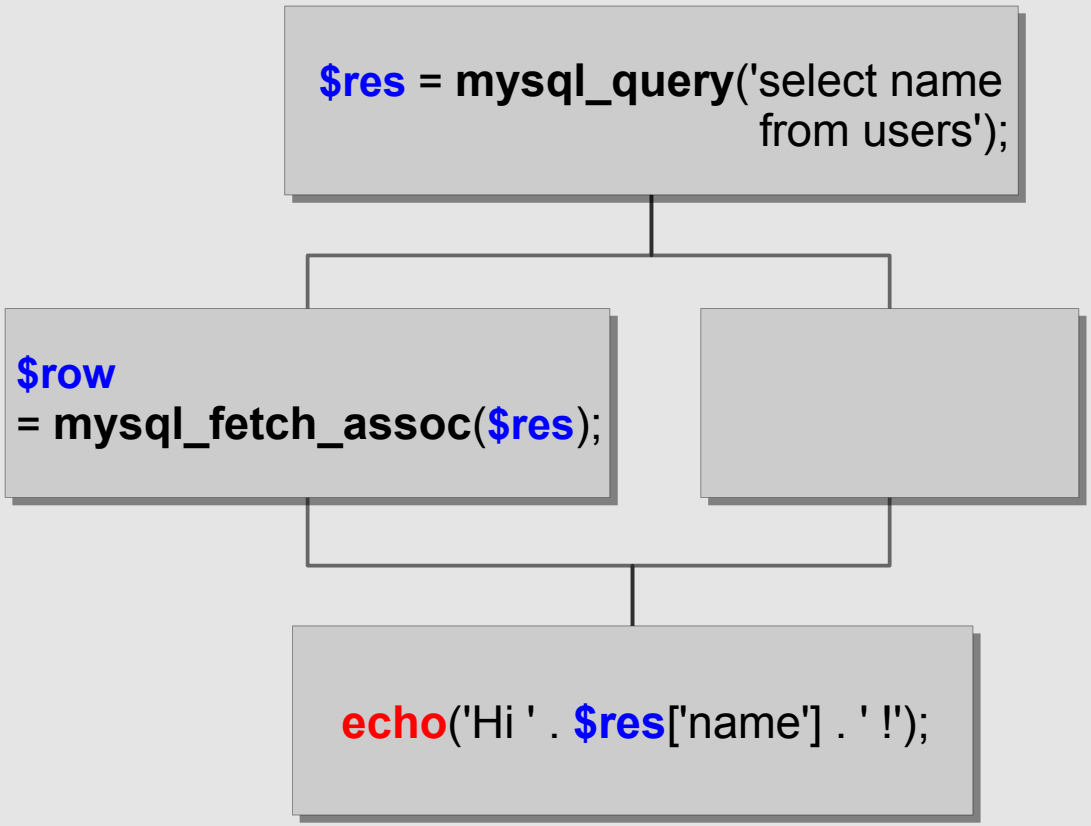
Vulnerability Report
POST[name]
SQLi

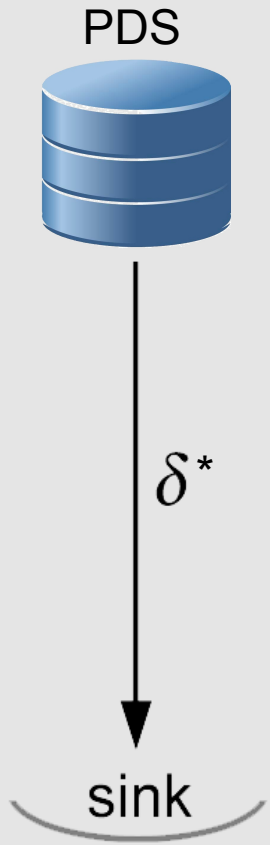
users

id	name	pass
	δ	δ

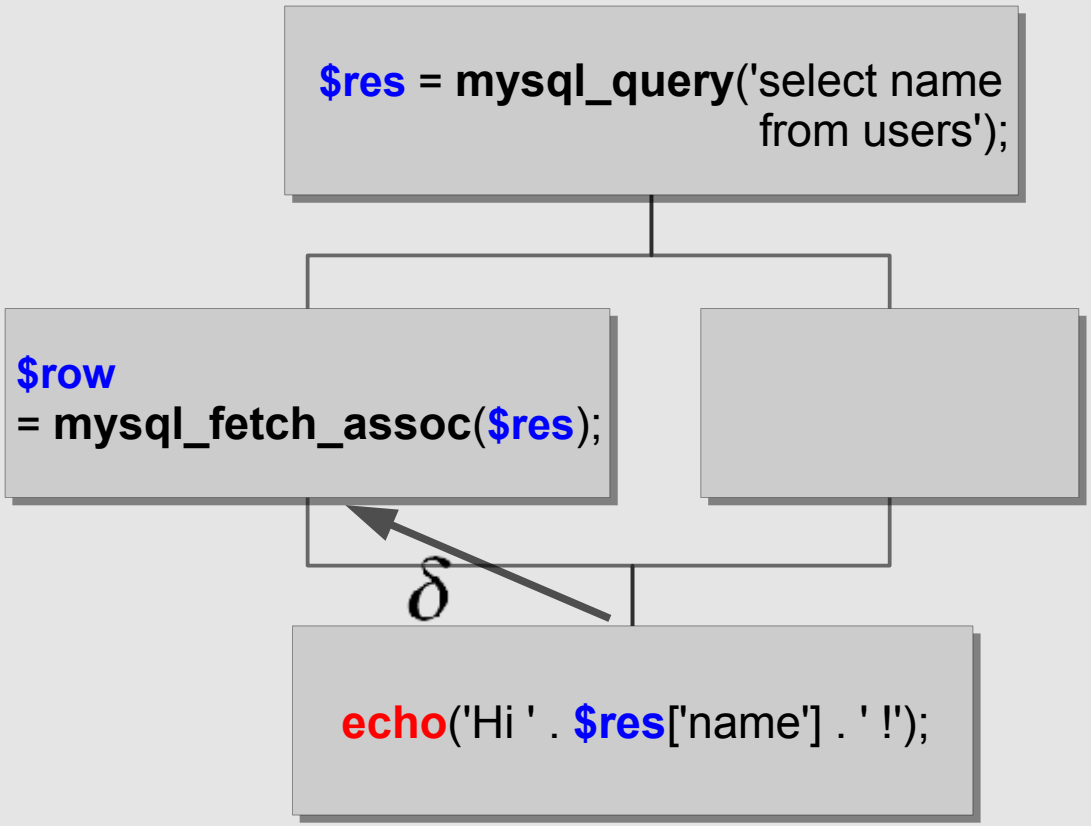


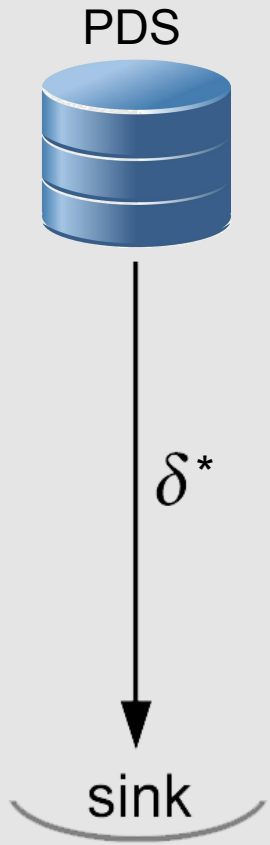
Second-Order Taint Analysis (read)



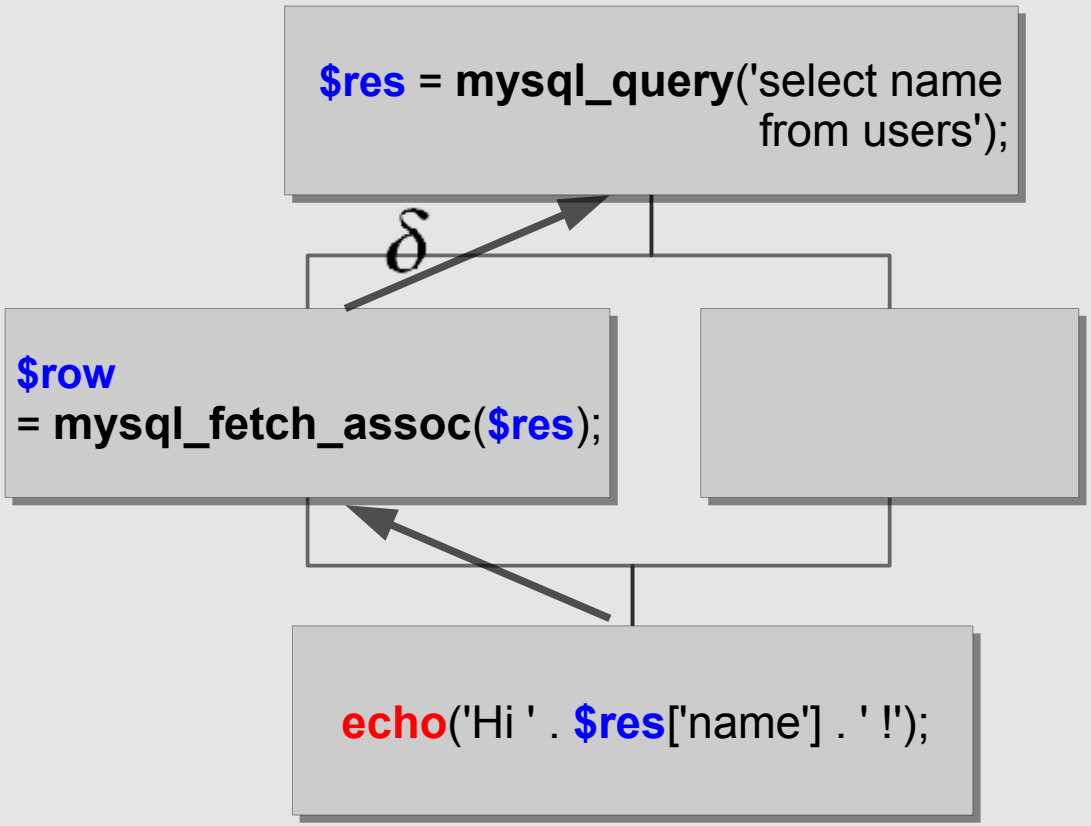


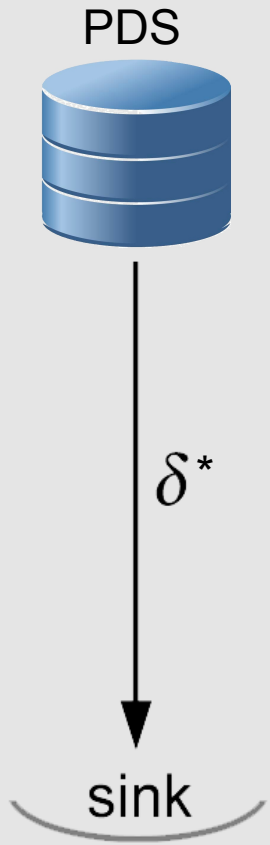
Second-Order Taint Analysis (read)



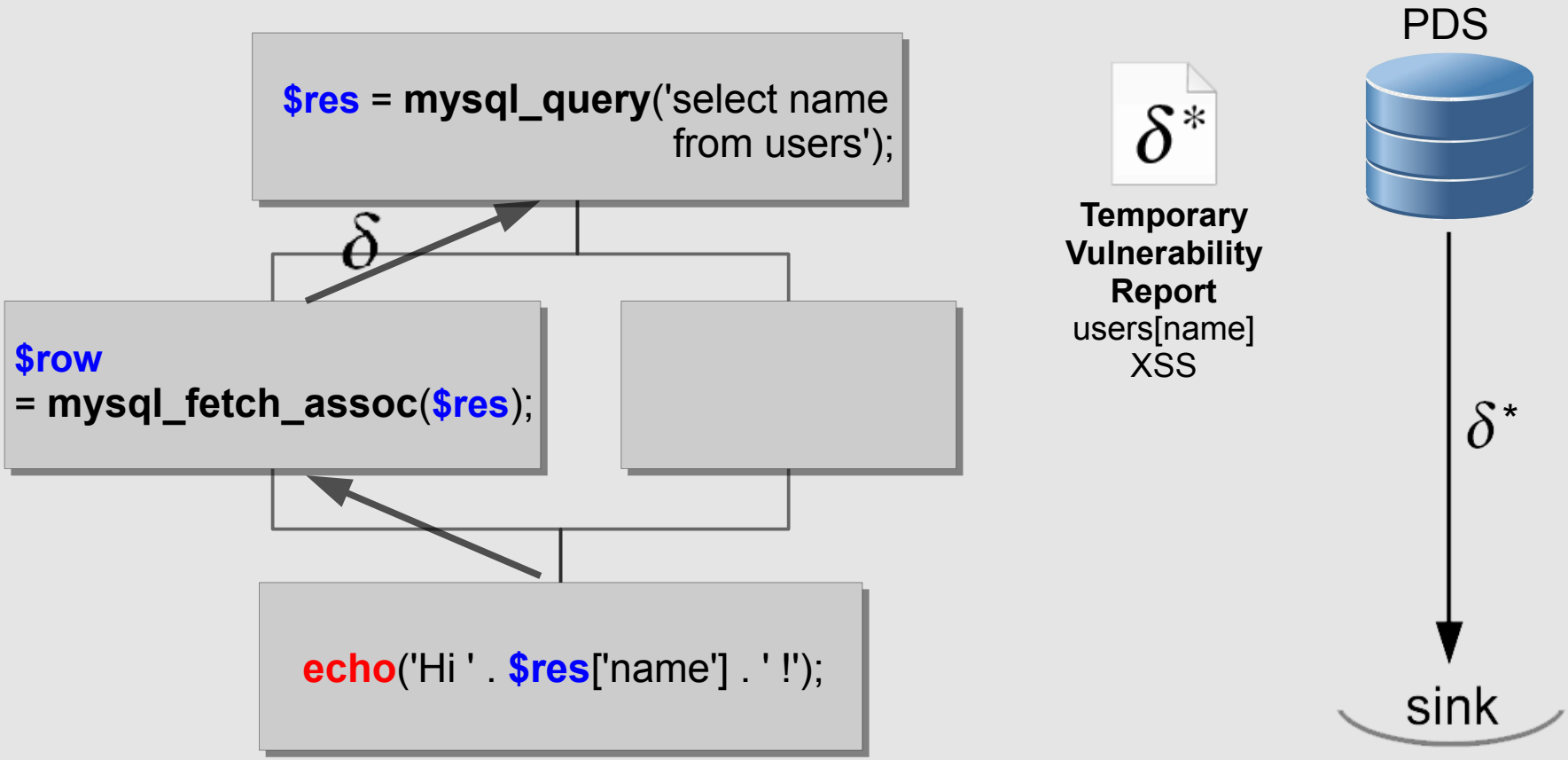


Second-Order Taint Analysis (read)

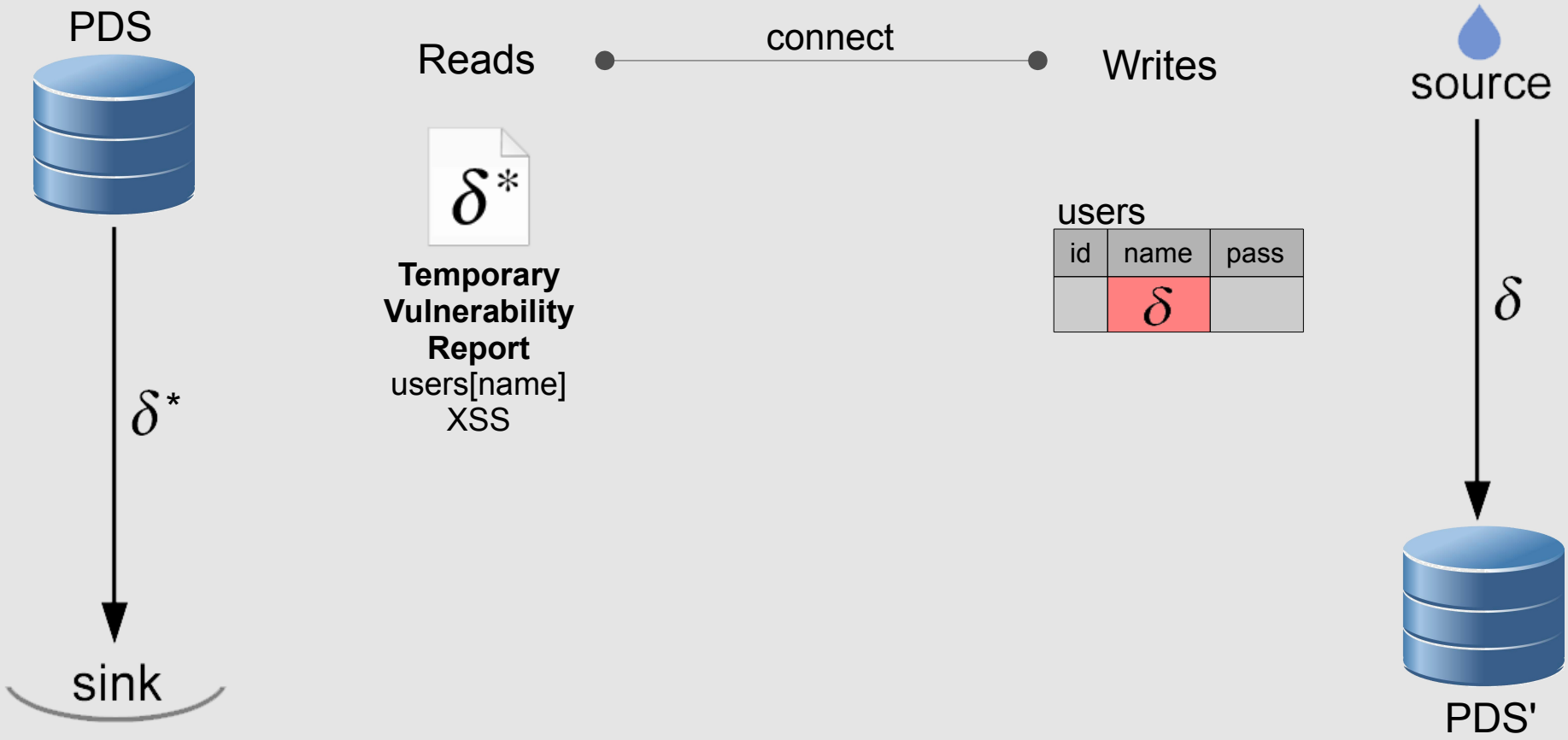




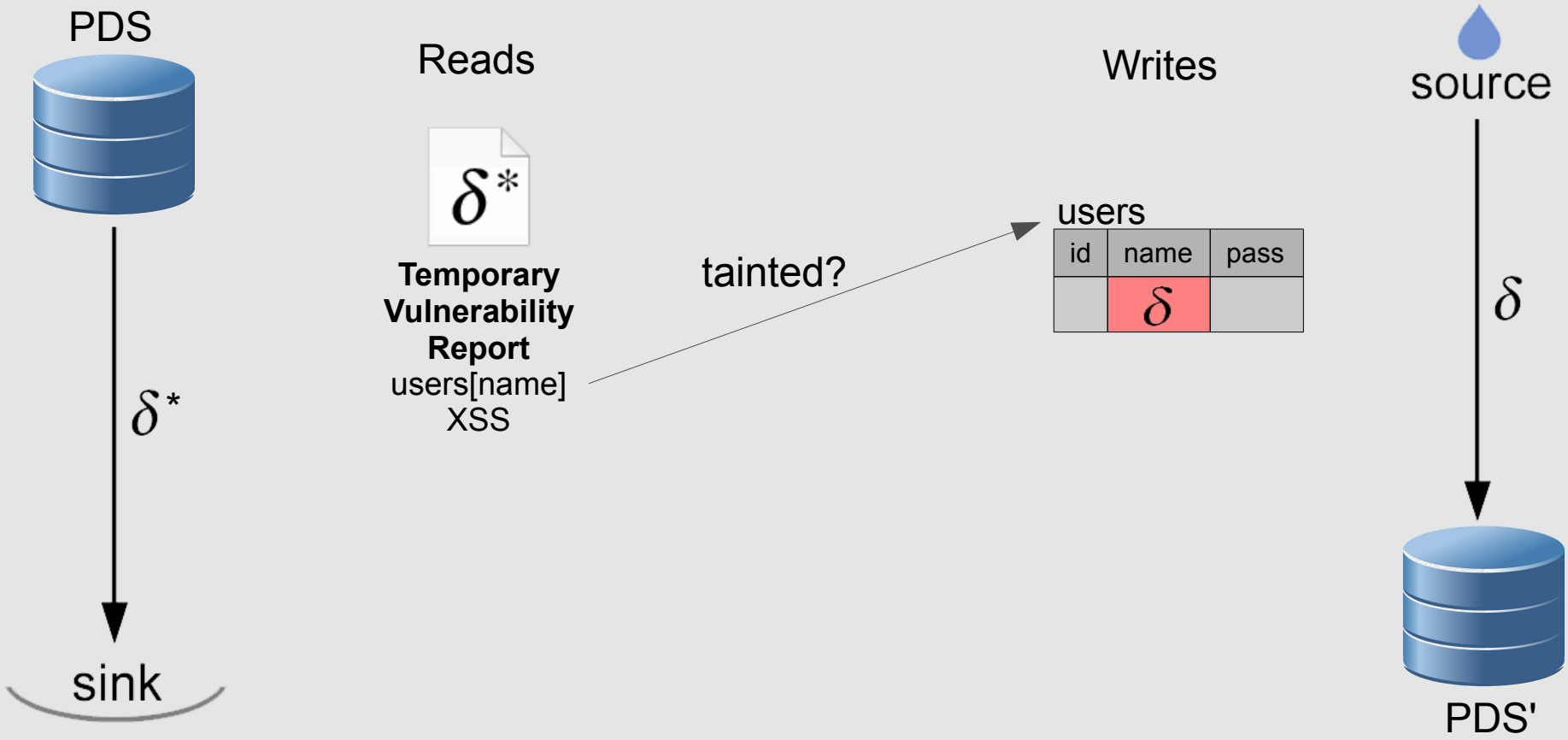
Second-Order Taint Analysis (read)



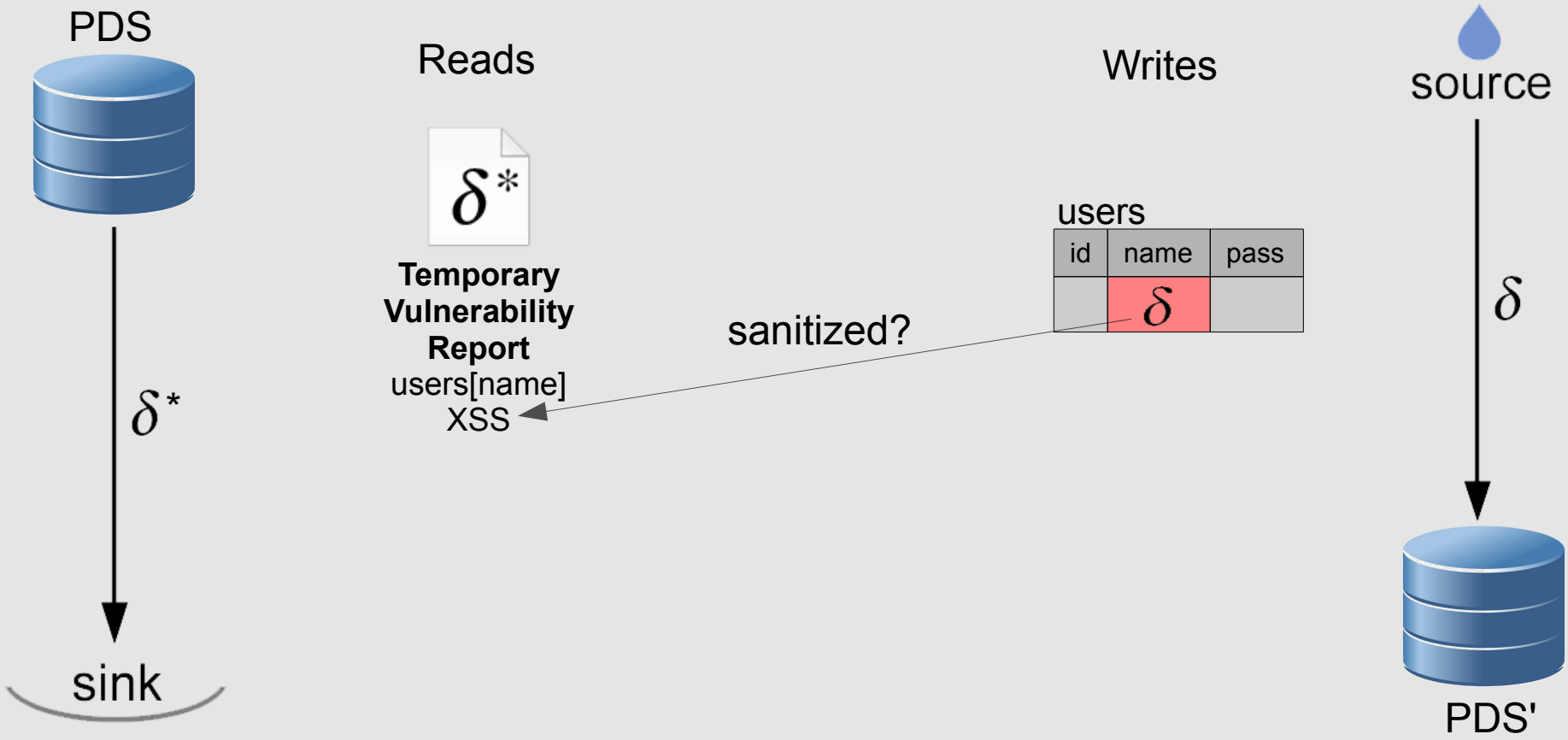
Second-Order Taint Decision



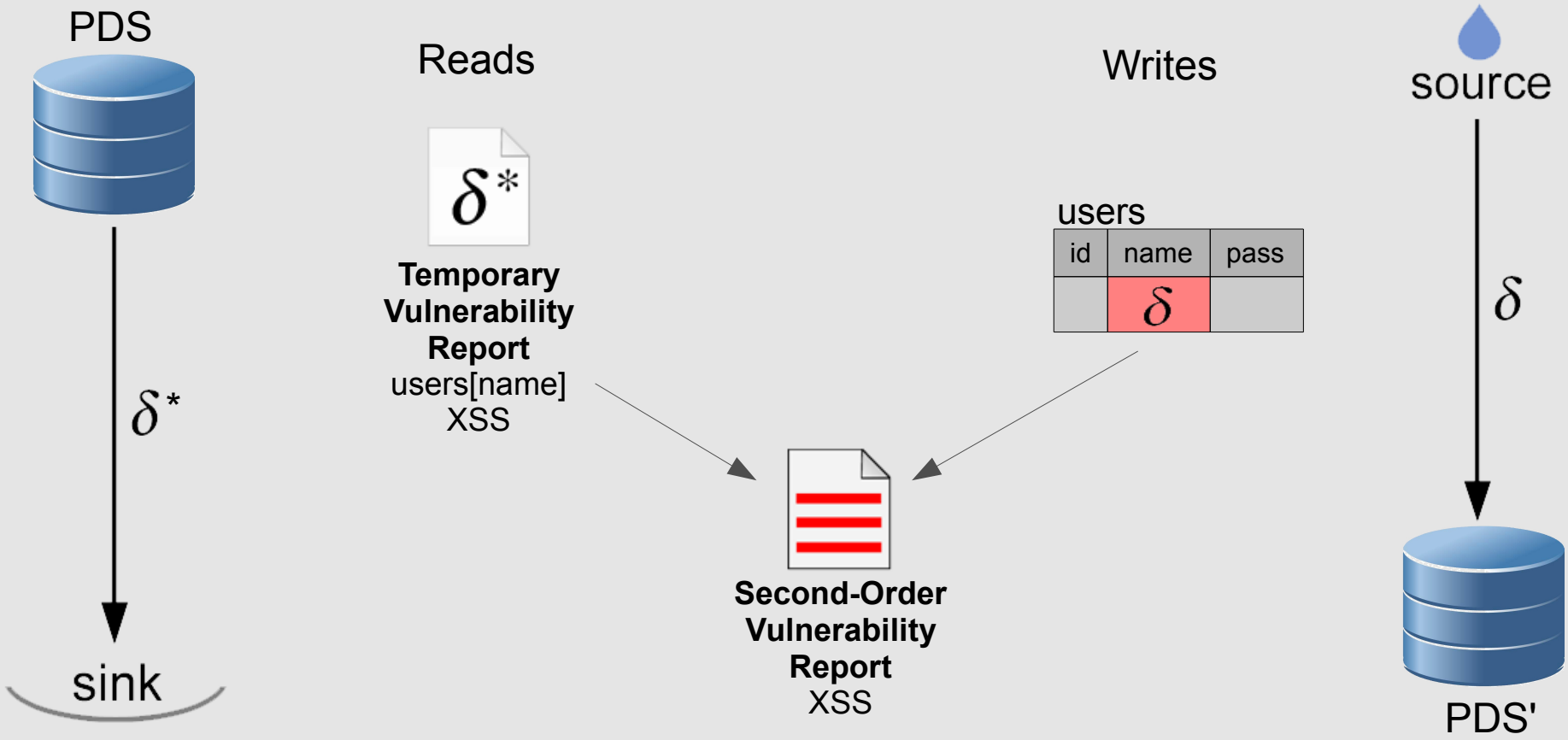
Second-Order Taint Decision

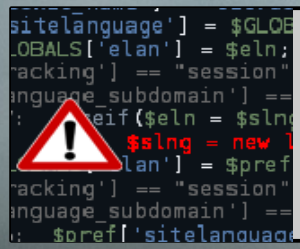


Second-Order Taint Decision



Second-Order Taint Decision





3. Evaluation

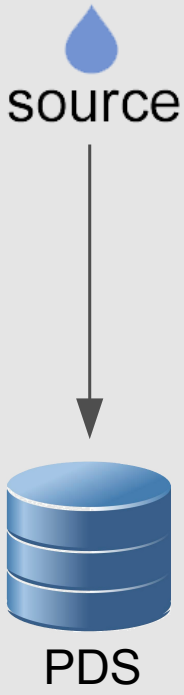
Source: http://rewalls.com

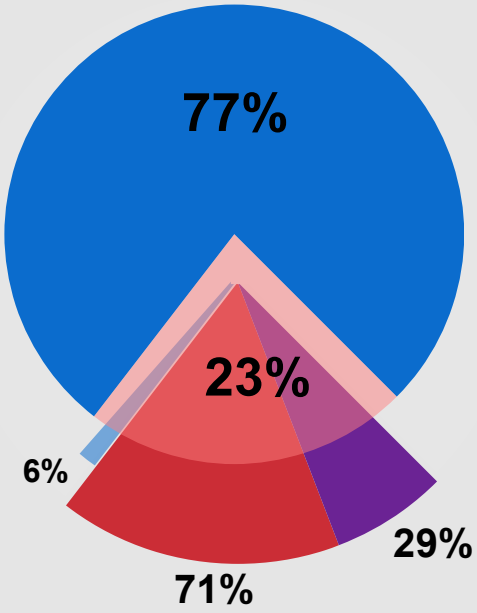
Selected Software

- osCommerce 2.3.3.4
- HotCRP 2.61
- OpenConf 5.30

- MyBloggie 2.1.4
- NewsPro 1.1.5
- Scarf 2007-02-27

PDS Usage and Coverage (first-order)





Manually counted PDS (841)

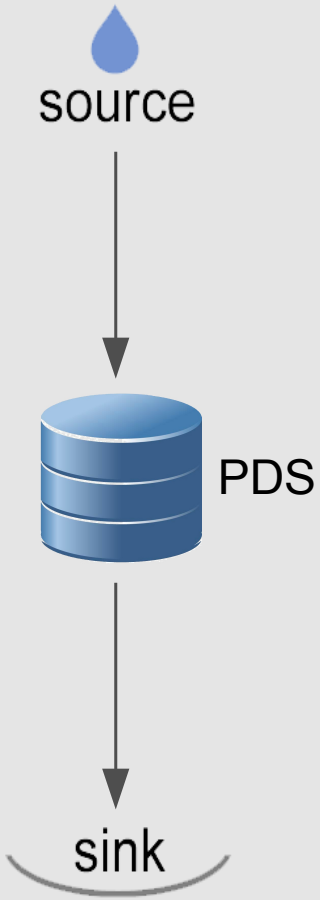
- Non-Taintable
- Taintable '"\<>

Detected Taintable PDS

- False Positive
- True Positive
- False Negative

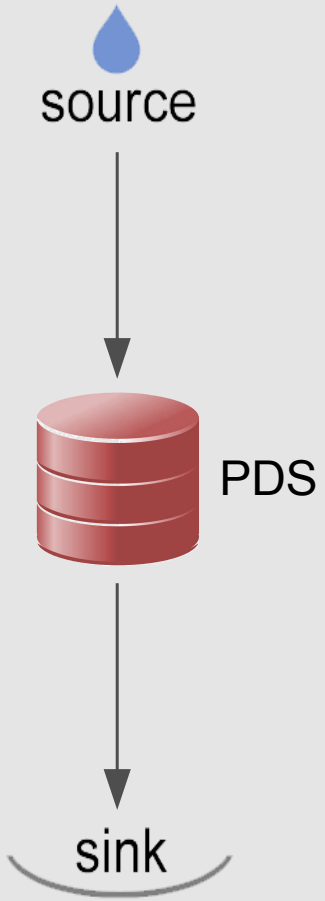
Second-Order Vulnerabilities

- 159 True Positives (79%)
  - 97% persistent XSS (database)
  - Missed by previous work
- 43 False Positives (21%)
  - Root cause: Path-sensitive sanitization
  - E.g., store only valid email
  - Failures in 1st step propagate to 2nd step



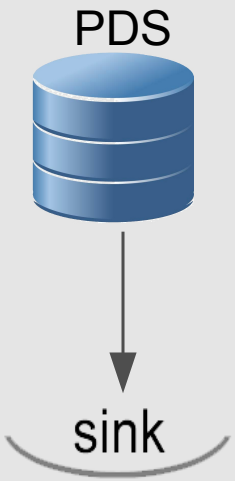
Multi-Step Exploits

- 14 True Positives (93%)
  - 2 based on file upload
  - 12 based on SQLi
  - Missed by previous work
- 1 False Positives (7%)
  - False positive SQLi



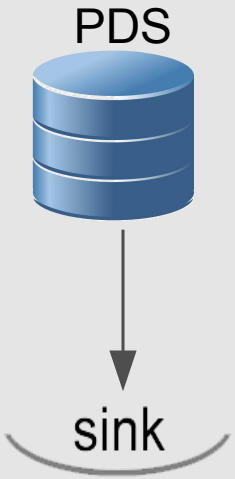
Second-Order LFI in OpenConf

```
$r = mysql_query("select setting, value from " . OCC_TABLE_CONFIG);
while ($l = mysql_fetch_assoc($r)) {
    $config[$l['setting']] = $l['value'];
}

function printHeader($what, $function="0") {
    require $GLOBALS['pfx'] . $GLOBALS['config']['OC_headerFile'];
}
```



Second-Order LFI in OpenConf

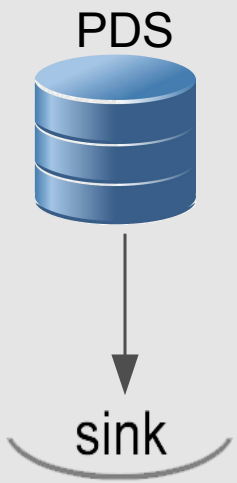
```
$r = mysql_query("select setting, value from " . OCC_TABLE_CONFIG);
while ($l = mysql_fetch_assoc($r)) {
      $config[$l['setting']] = $l['value'];
}

function printHeader($what, $function="0") {
      require $GLOBALS['pfx'] . $GLOBALS['config']['OC_headerFile'];
}
```

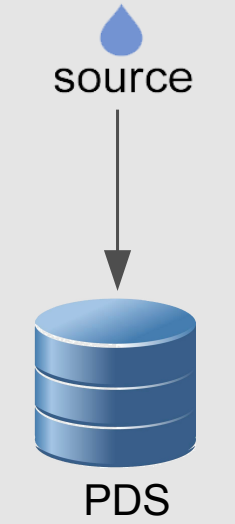


Second-Order LFI in OpenConf

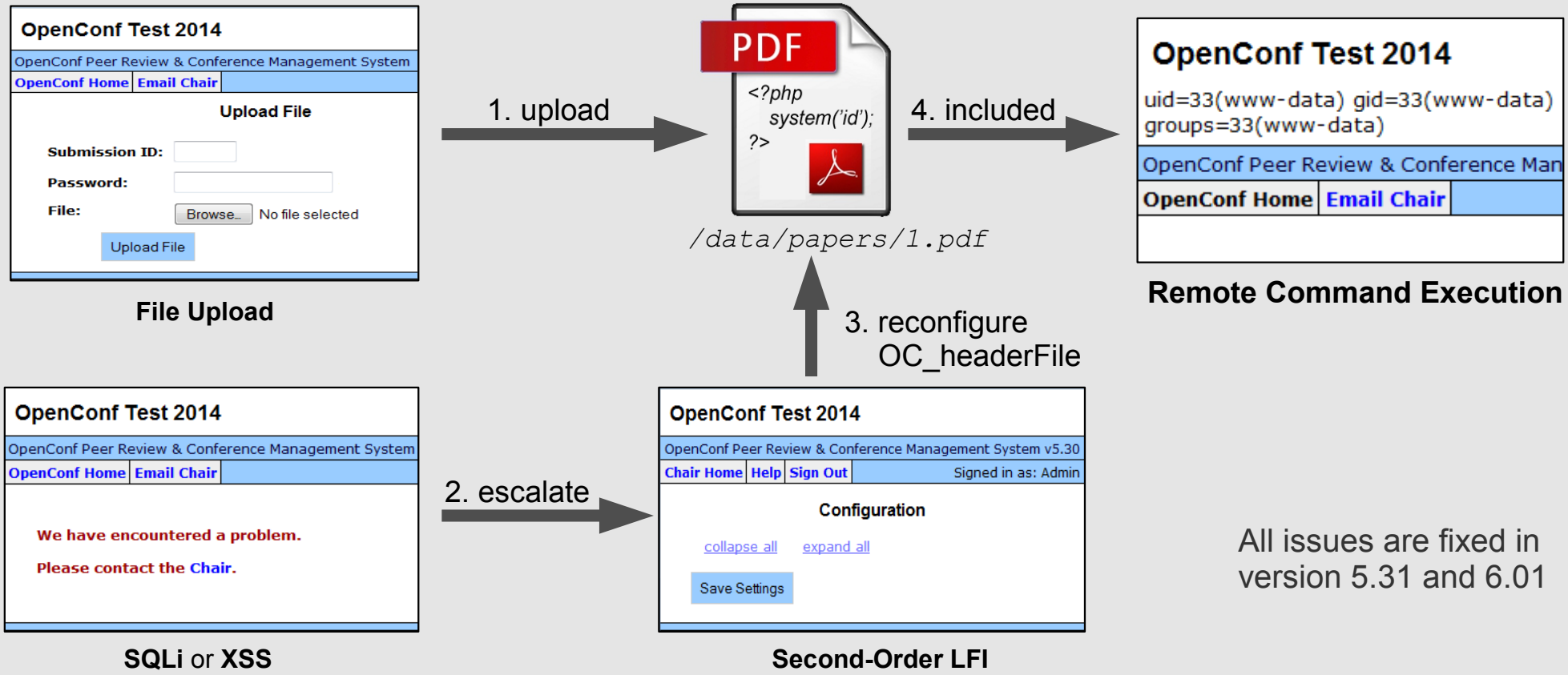
```
$r = mysql_query("select setting, value from " . OCC_TABLE_CONFIG);
while ($l = mysql_fetch_assoc($r)) {
    $config[$l['setting']] = $l['value'];
}

function printHeader($what, $function="0") {
    require $GLOBALS['pfx'] . $GLOBALS['config']['OC_headerFile'];
}
```

```
function updateConfigSetting($setting, $value) {
    ocsql_query("UPDATE `" . OCC_TABLE_CONFIG . "`
                 SET `value`=' " . safeSQLstr(trim($value)) . " '
                 WHERE `setting`='" . safeSQLstr($setting) . " ' ");
}

foreach (array_keys($_POST) as $p) {
    if (preg_match("/^OC_[\w-]+$/", $p)) {
        updateConfigSetting($p, $_POST[$p]);
    }
}
```





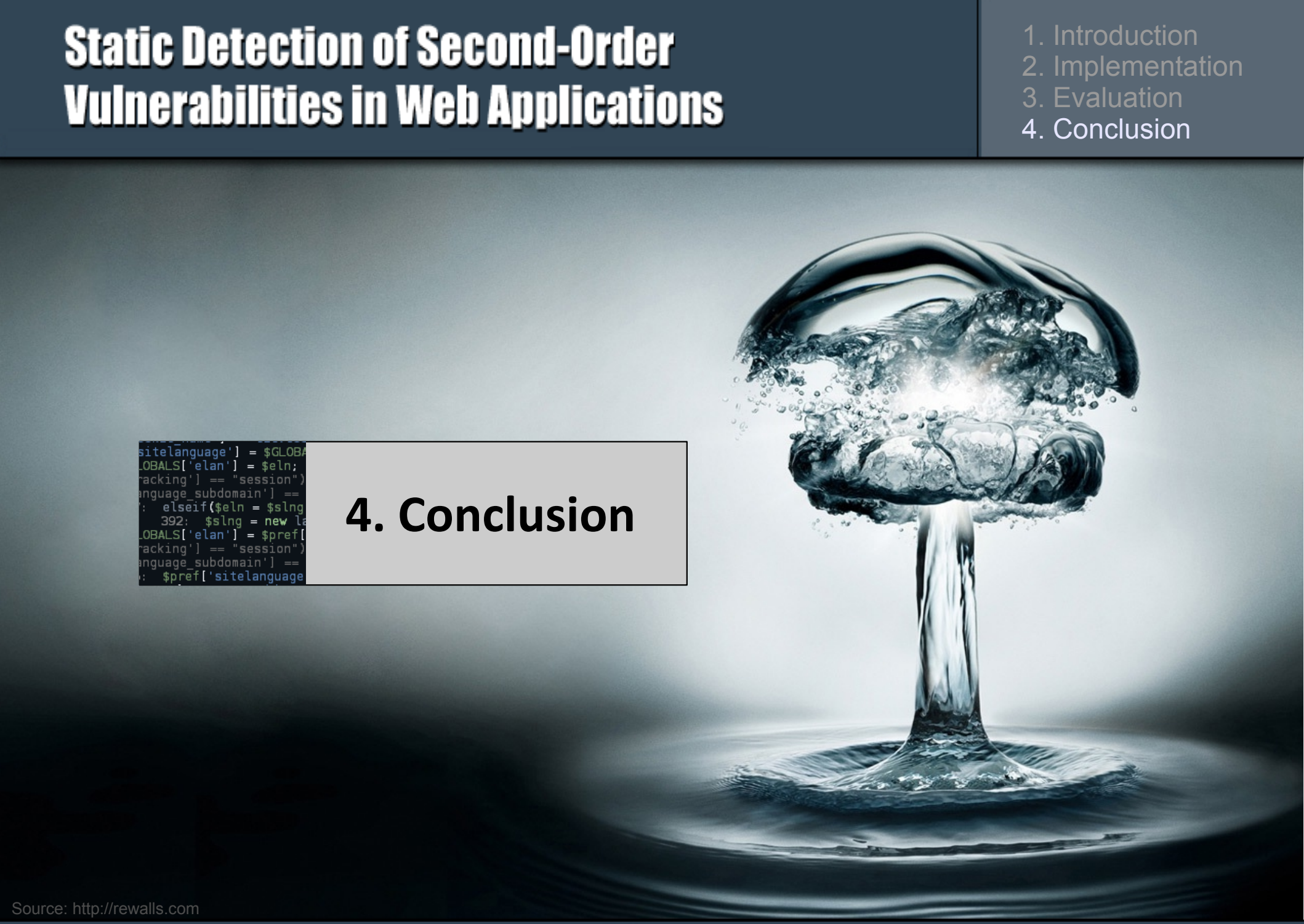
Multi-Step Exploitation in OpenConf



All issues are fixed in version 5.31 and 6.01

```
sitelanguage'] = $GLOBA
LOBALS['elan'] = $eln;
racking'] == "session")
anguage_subdomain'] ==
  elseif($eln = $slng
  392:  $slng = new la
LOBALS['elan'] = $pref[
racking'] == "session")
anguage_subdomain'] ==
  $pref['sitelanguage
```

4. Conclusion



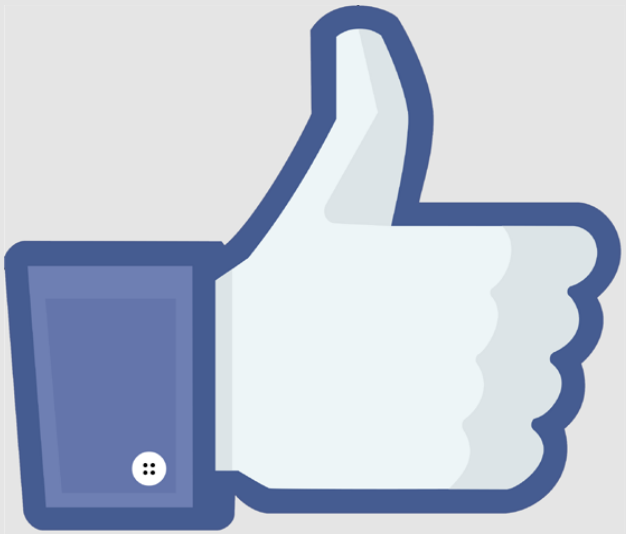
Source: http://rewalls.com

Conclusion

- Static detection of second-order vulnerabilities is possible
  - Analyze and collect reads/writes to PDS (database, file names, session data)
  - Determine sensitive data flow at the end of analysis
- > 150 new vulnerabilities
  - Leading to RCE in NewsPro, Scarf, OpenConf, osCommerce
  - Overlooked problem in practice, missed in previous work
- Future work
  - Support prepared statements
  - Improve SQL parser

Thank you Facebook

for the generous award



Questions?

johannes.dahse@rub.de

Thank you!
Enjoy the conference.