72nd
IFIPWG 10.4
Meeting

Detection of Vulnerabilities broken by Circular Dependencies in Static Analysis

Ibéria Medeiros
LaSIGE, Faculty of Sciences, University of Lisboa

The security of web applications continues to be a challenging problem

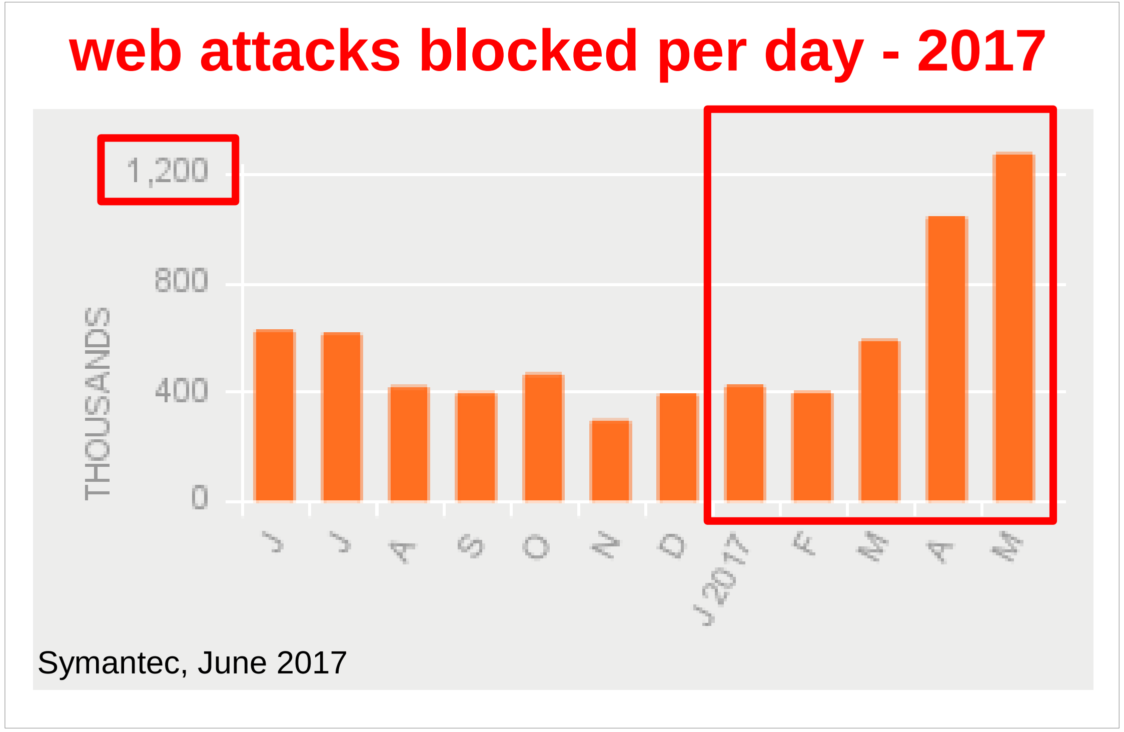
vulnerabilities

left in the source code by

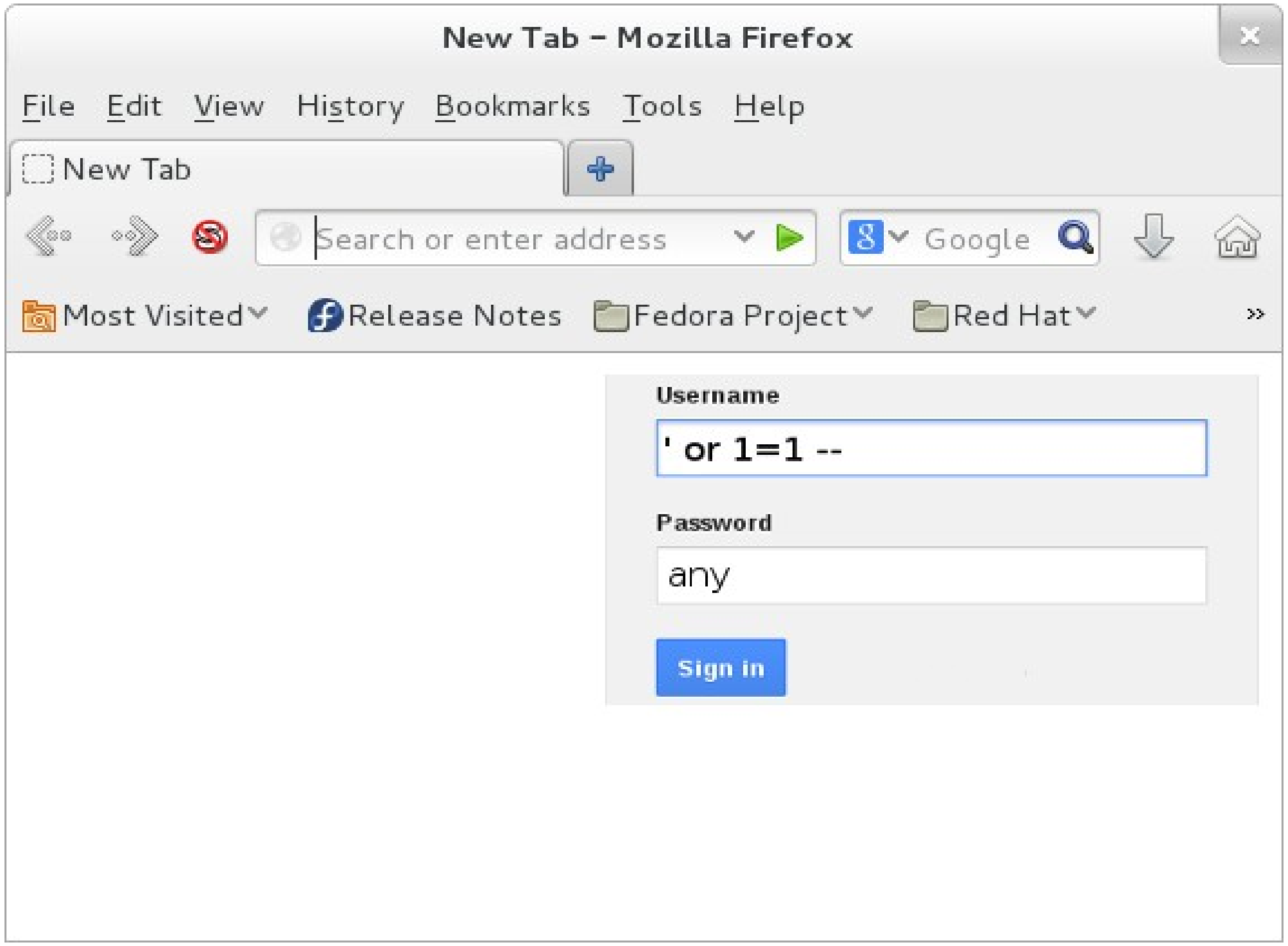
developers that make mistakes

attacks

SQL injection
XSS
RFI
...



Vulnerability SQL Injection example



Username

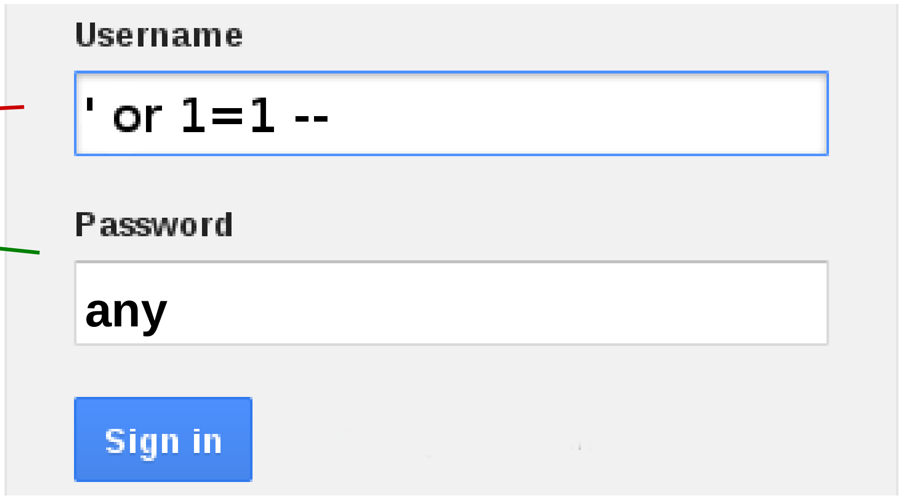
' or 1=1 --

Password

any

Motivation | Vulnerability SQL Injection example

```
$u = $_POST['user'];
$p = $_POST['password'];
$q = “SELECT * FROM users
        WHERE user='$u' AND pass='$p'”;
$r = mysql_query($q);
```



Username

' or 1=1 --

Password

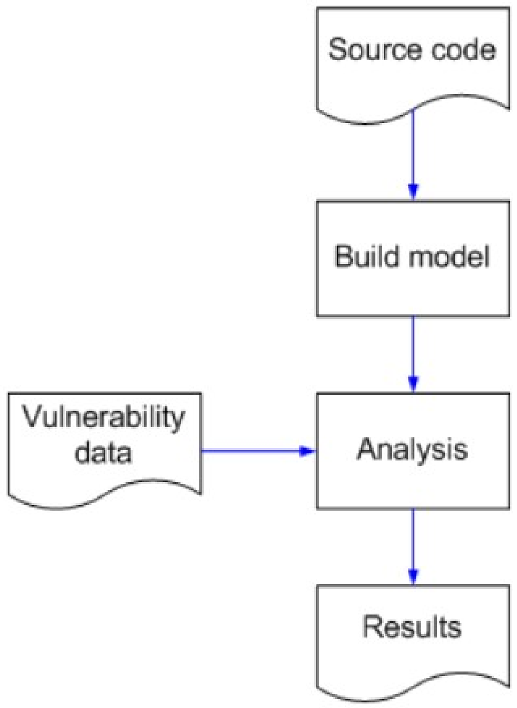
any

```
$u = ' or 1=1 -- ;
$p = any;
$q = “SELECT * FROM users WHERE user='' or 1=1;-- ' AND pass='any'”;
$r = mysql_query($q);
```

SQL injection vulnerability exploited !!

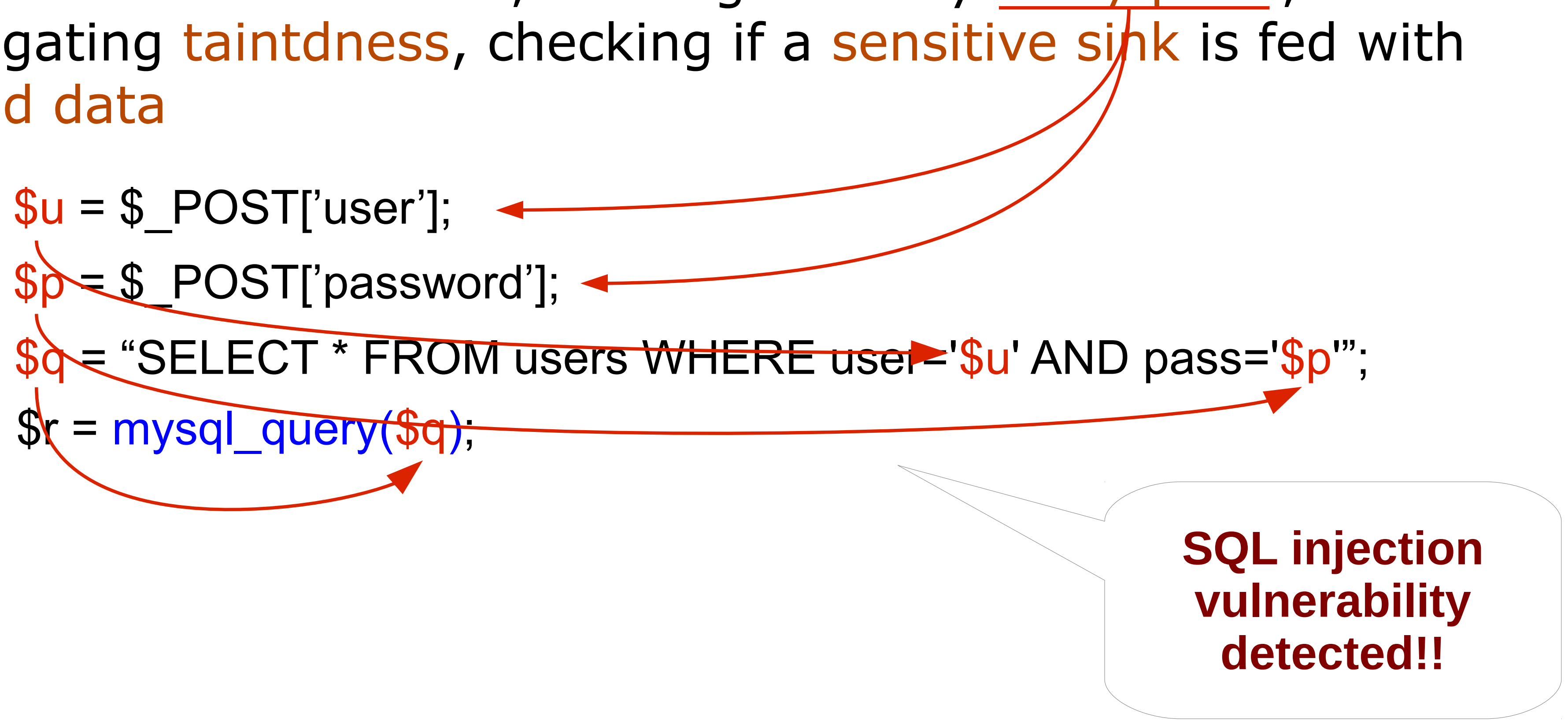
Source Code Static Analysis

- Objective: to find vulnerabilities in the applications' (source) code automatically
  - Similar to compiler's error checking but for vulnerabilities
  - Similar to manual code reviewing but automatically
- Static: because the code is not executed



Taint Analysis

Analyses the source code, starting at every entry point, propagating taintdness, checking if a sensitive sink is fed with tainted data



```
$u = $_POST['user'];
$p = $_POST['password'];
$q = "SELECT * FROM users WHERE user='$u' AND pass='$p'";
$r = mysql_query($q);
```

SQL injection vulnerability detected!!

Web Application Protection

Taint Analysis

Analyses the source code, starting at every entry point, propagating taintdness, checking if a sensitive sink is fed with tainted data

```
$u = $_POST['user'];
$p = $_POST['password'];
$q = "SELECT * FROM users WHERE user='$u' AN
$r = mysql_query($q);
```

SQL injection vulnerability detected!!

```
$u = $_POST['user'];
$p = $_POST['password'];
$uu = mysql_real_escape_string($u);
$pp = mysql_real_escape_string($p);
$q = "SELECT * FROM users WHERE user='$uu' AND pass='$pp'";
$r = mysql_query($q);
```

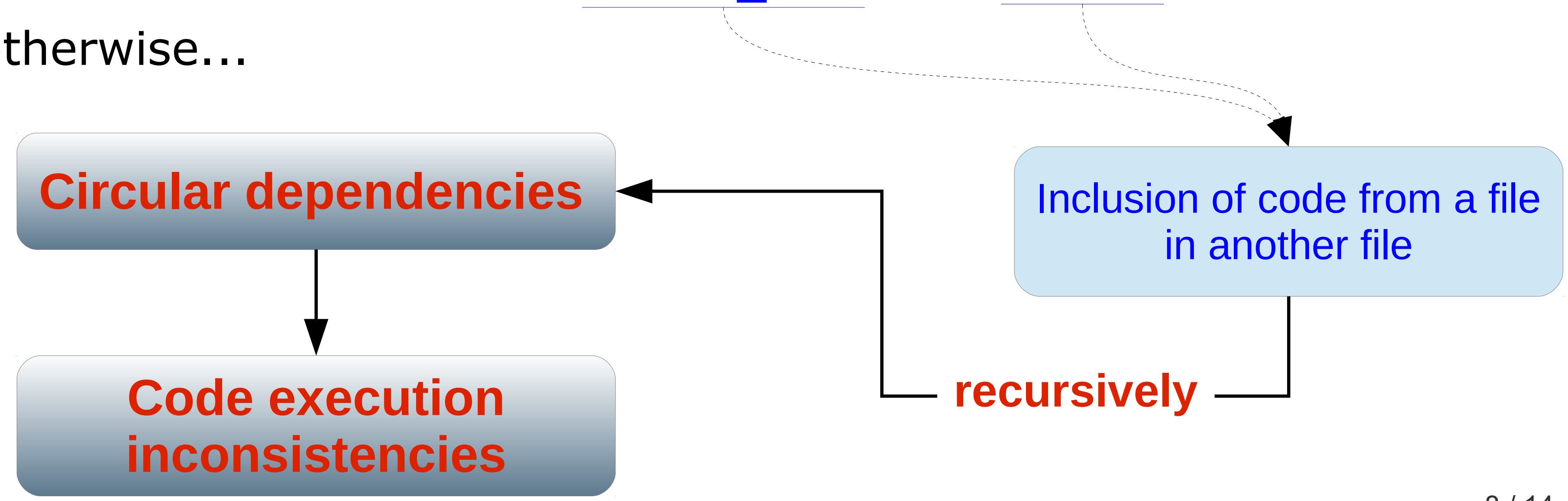
some functions sanitize, so "untaint", the data flow

SECURE APPLICATTIONS

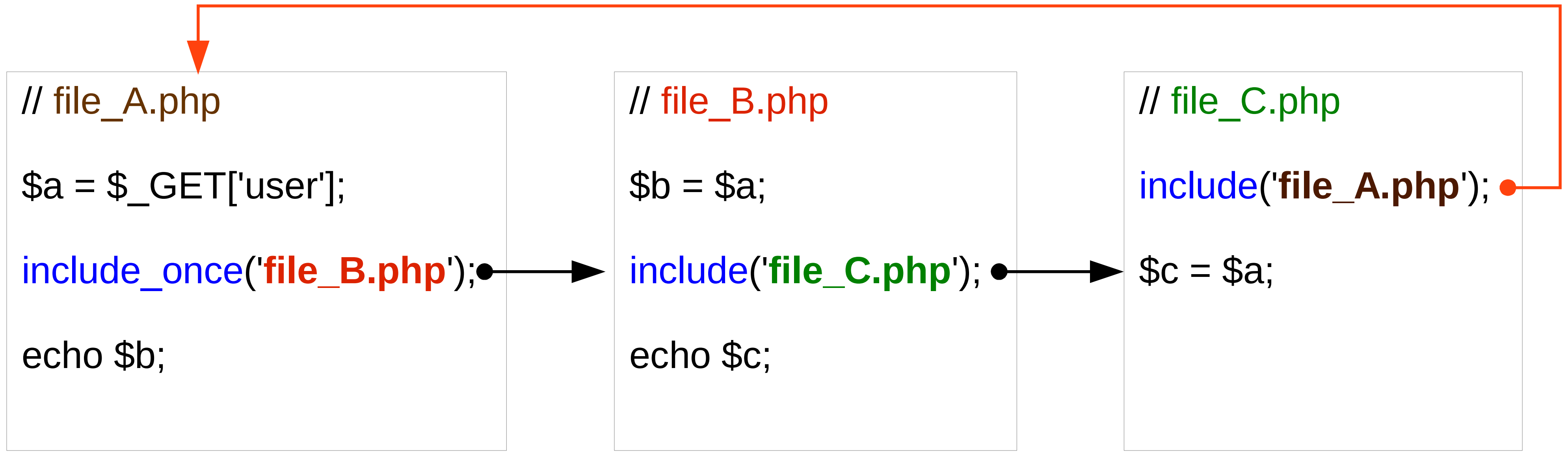
- Create secure applications is an important factor
- Knowledge about how to build secure code is required
  - sanitize and/or validate entry points
  - otherwise, vulnerabilities are left in the code

On-going work

Correct use of the functionalities of the programming language
  - differentiate when to use include_once and include functions
  - otherwise...


Circular dependencies
Code execution inconsistencies
Inclusion of code from a file in another file
recursively

Circular Dependencies



```
// file_A.php

$a = $_GET['user'];

include_once('file_B.php');

echo $b;
```

```
// file_B.php

$b = $a;

include('file_C.php');

echo $c;
```

```
// file_C.php

include('file_A.php');

$c = $a;
```

CIRCULAR DEPENDENCIES IN STATIC ANALYSIS

- Static analysis analyzes the code of include files for each time a include or include_once instruction appears
- If there are circular dependencies in the source code then they will be notice in static analysis
- Circular dependencies break static analysis process

- **A infinite loop is created**
- **Code execution inconsistency is generated**

Vulnerability detection is broken

Resolving Circular Dependencies using Static Analysis

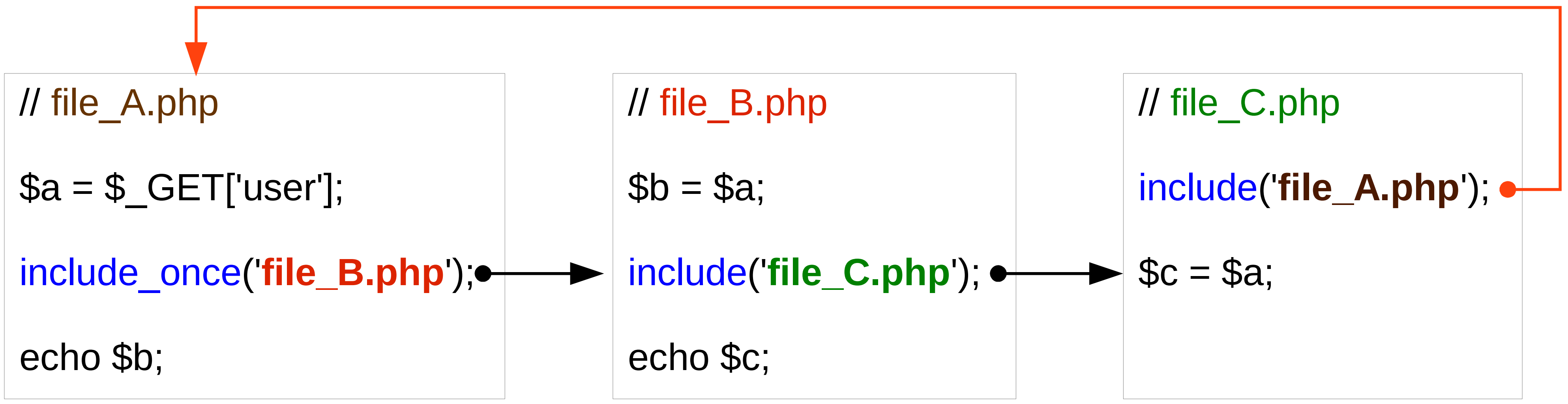
Build execution file paths

- Get for each file its include files
- Create trees to represent the execution file paths, identifying the parents and children for each tree node
- Identify the circular dependency points using the parents and children information

Realize taint analysis

- Perform taint analysis in each execution file path
- For circular dependencies points
  - include_once, the analysis stops there
  - include, the remaining code is analyzed

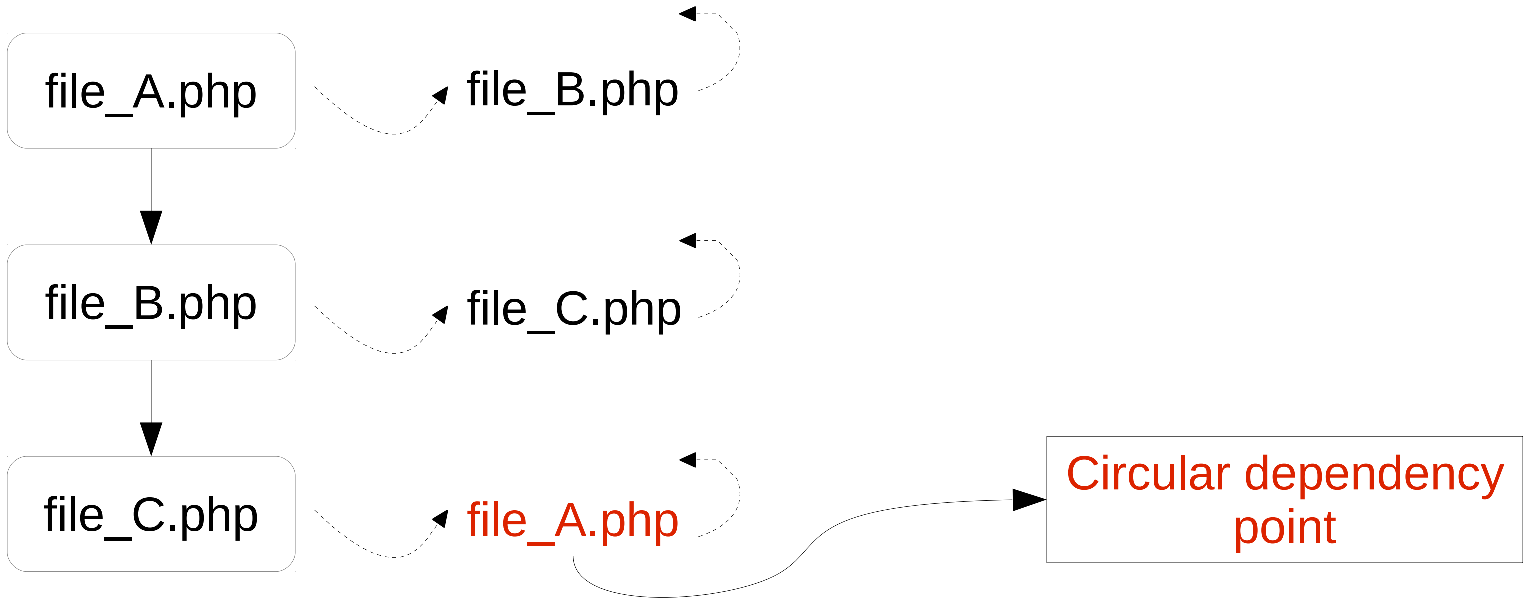
Execution File Path



```
// file_A.php

$a = $_GET['user'];

include_once('file_B.php');

echo $b;
```

```
// file_B.php

$b = $a;

include('file_C.php');

echo $c;
```

```
// file_C.php

include('file_A.php');

$c = $a;
```



Some Results

- We evaluate 4 static analysis tools with...
  - include and include_once instructions
  - include files with code and user functions

Results with circular dependencies points

- Some tools stop analysis for include_once
- Some tools stop analysis for include, resulting false negatives
- Some tools crash with include

Circular dependencies is an effective problem in static analysis tools

72nd
IFIPWG 10.4
Meeting

Detection of Vulnerabilities broken by Circular Dependencies in Static Analysis

Ibéria Medeiros
LaSIGE, Faculty of Sciences, University of Lisboa

Thank you!