Large Scale Generation of Complex and Faulty PHP Test Cases

Bertrand STIVALET

Elizabeth FONG

http://samate.nist.gov

ICST 2016
Chicago, IL, USA
April 15th, 2016

Authors

Bertrand STIVALET

National Institute of Standards and Technology

bertrand.stivalet@nist.gov

Elizabeth FONG

National Institute of Standards and Technology

efong@nist.gov

*"If **debugging** is the process of removing software bugs, then **programming** must be the process of putting them in"*

E. Dijkstra

NIST - SAMATE - SARD

- NIST - National Institute of Standards and Technology
  - Part of the US Department Of Commerce
  - Promote U.S. Innovation and Industrial Competitiveness

- SAMATE - Software Assurance Metrics And Tool Evaluation
  - Improve Software Assurance by:
    - developing materials, specifications, and methods
    - testing tools and techniques and measure their effectiveness

- SARD - Software Assurance Reference Dataset
  - Provide database of known security flaws
  - C/C++, JAVA, PHP, C#
  - 148,903 Test cases / 665,481 Files

Outline

1. Software Testing

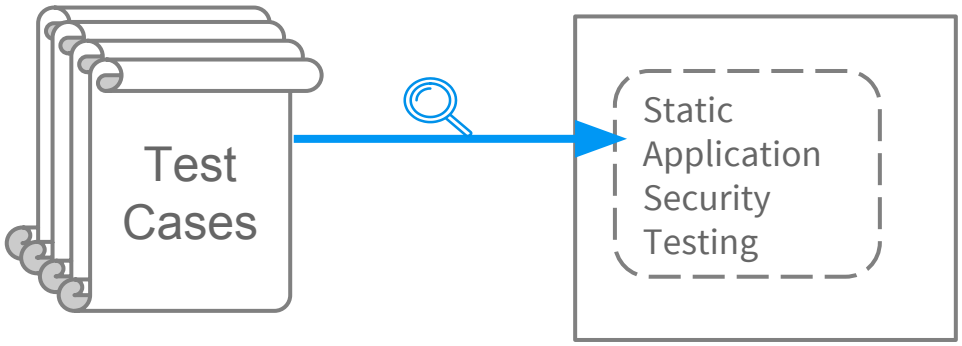
Outline

1. Software Testing

Outline

2. Design of Test Cases

1. Software Testing

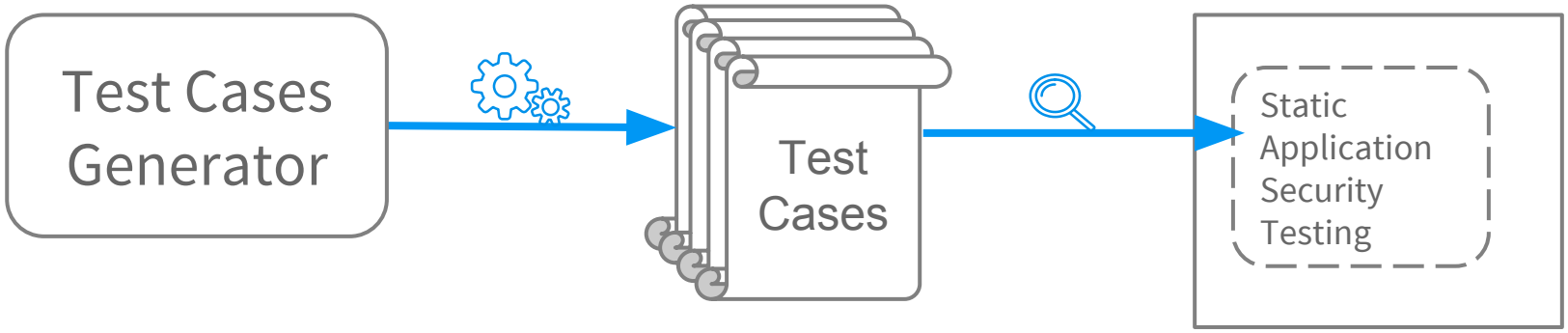


Outline

3. PHP Vulnerability Test Cases Generator

2. Design of Test Cases

1. Software Testing

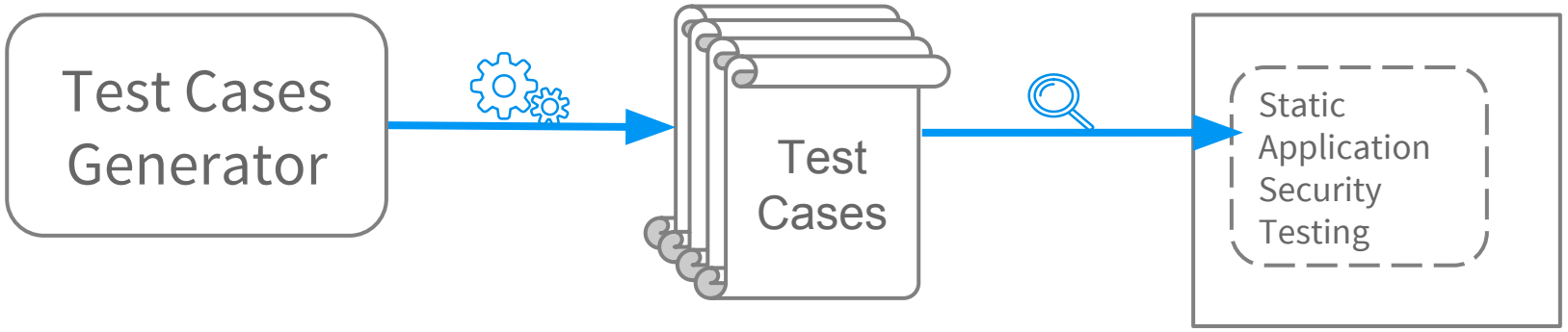


Outline

3. PHP Vulnerability Test Cases Generator

2. Design of Test Cases

1. Software Testing



4. Live Demo

1.
Software Testing

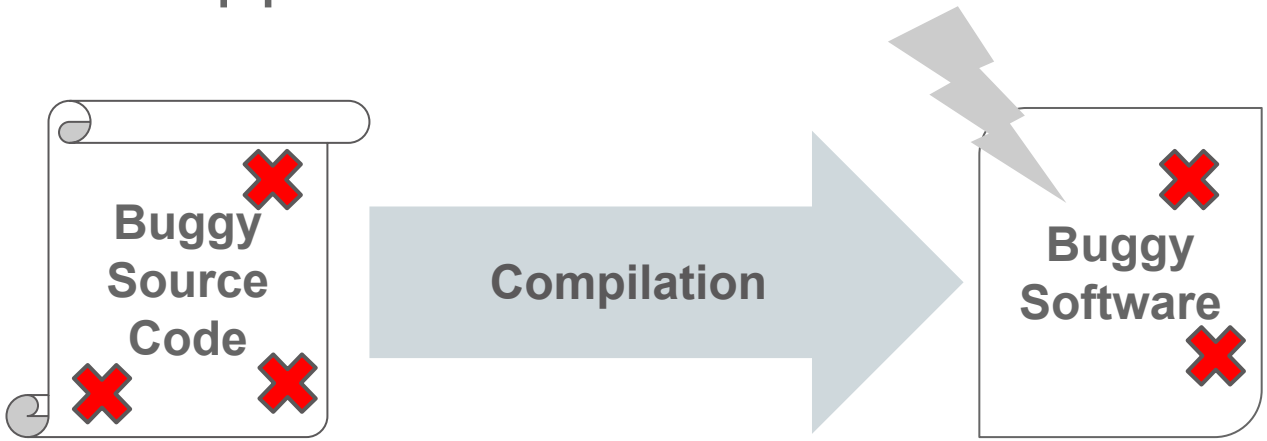
Introduction to Static Analysis

Static Analysis

- Automated analysis of large software
- Defect detection and remediation
- Use different approaches:
  - Syntax checking
  - Heuristics
  - Formal methods

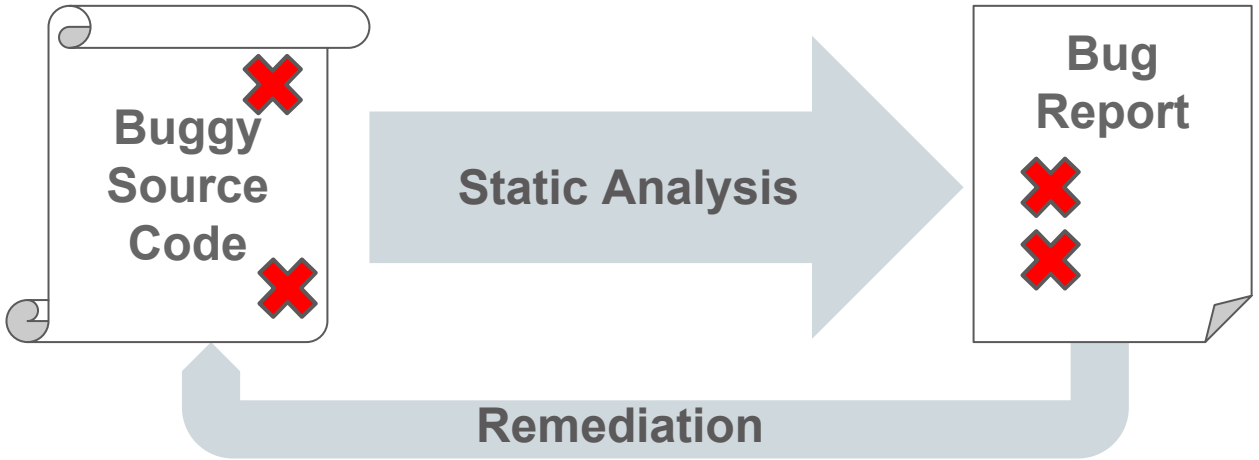
Static Analysis

- Automated analysis of large software
- Defect detection and remediation
- Use different approaches



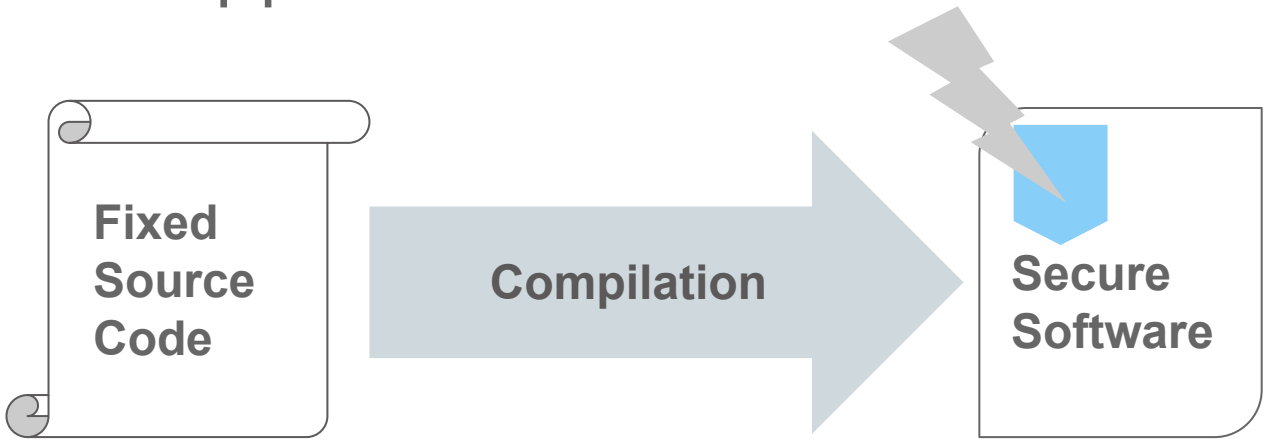
Static Analysis

- Automated analysis of large software
- Defect detection and remediation
- Use different approaches



Static Analysis

- Automated analysis of large software
- Defect detection and remediation
- Use different approaches

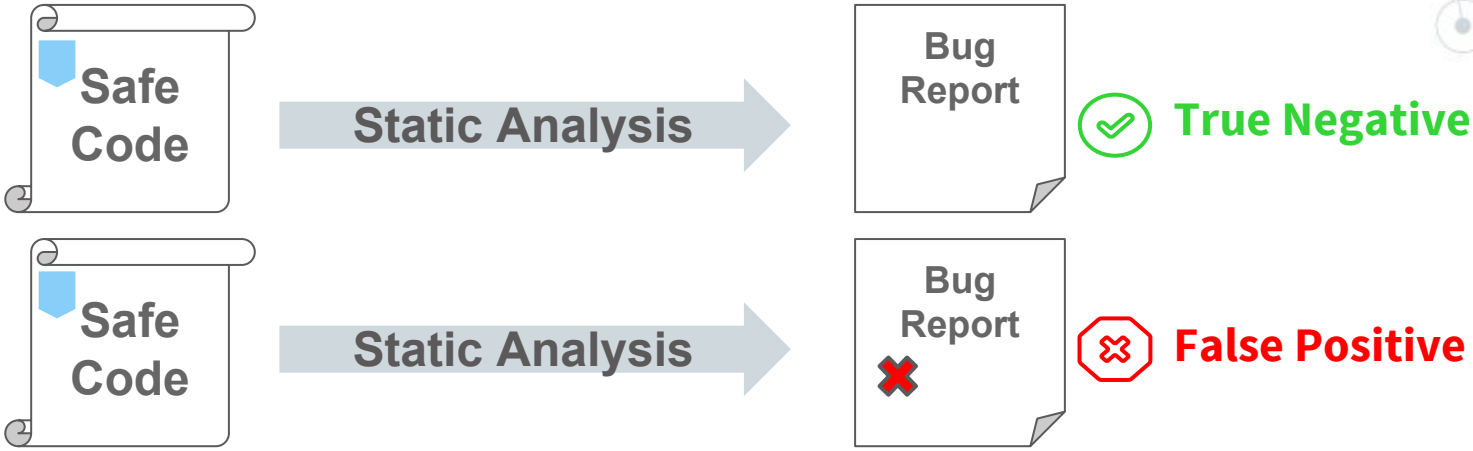


Static Analysis

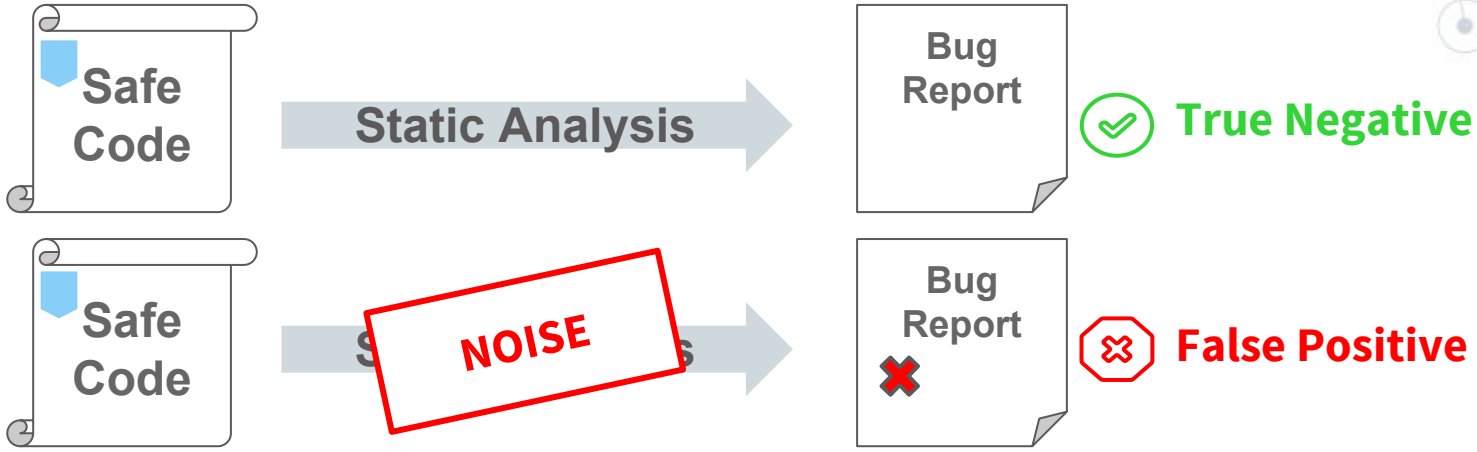
- Automated analysis of large software
- Defect detection and remediation
- Use different approaches

Static Analysis Testing

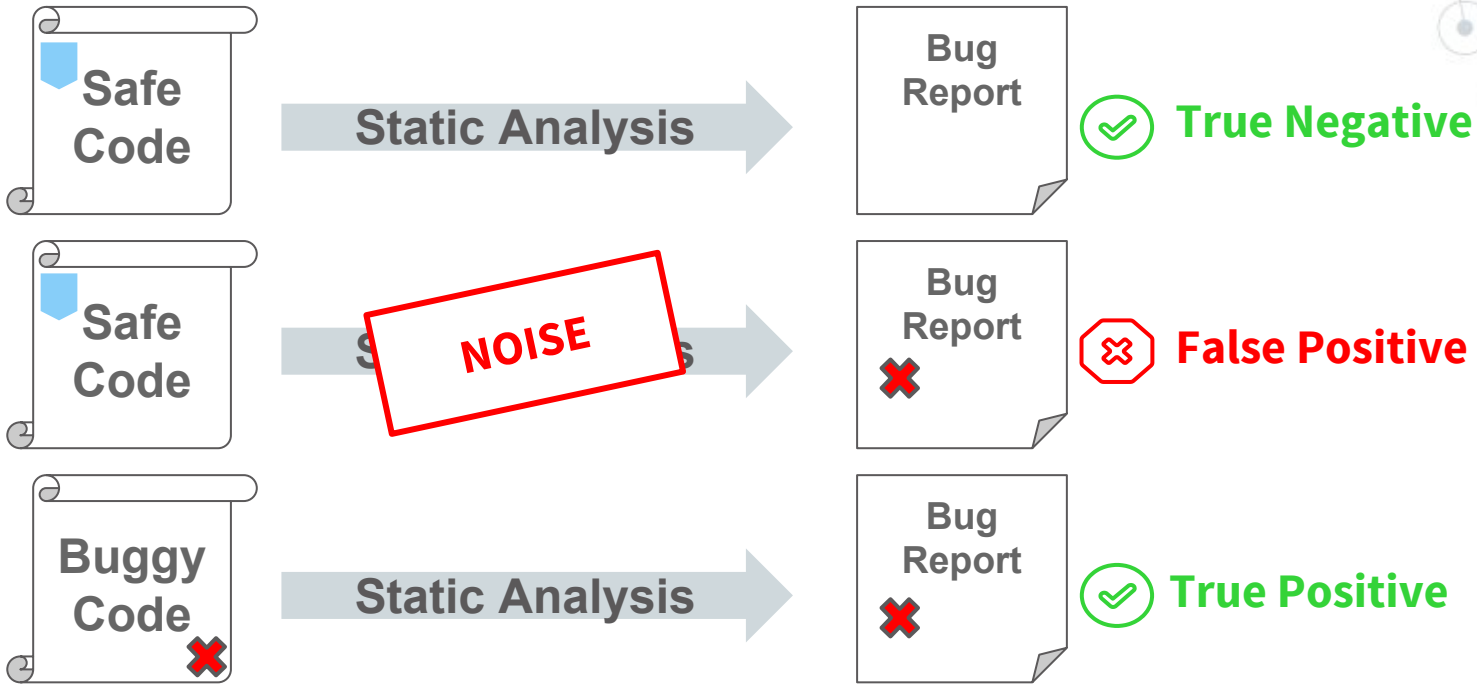
Static Analysis Testing



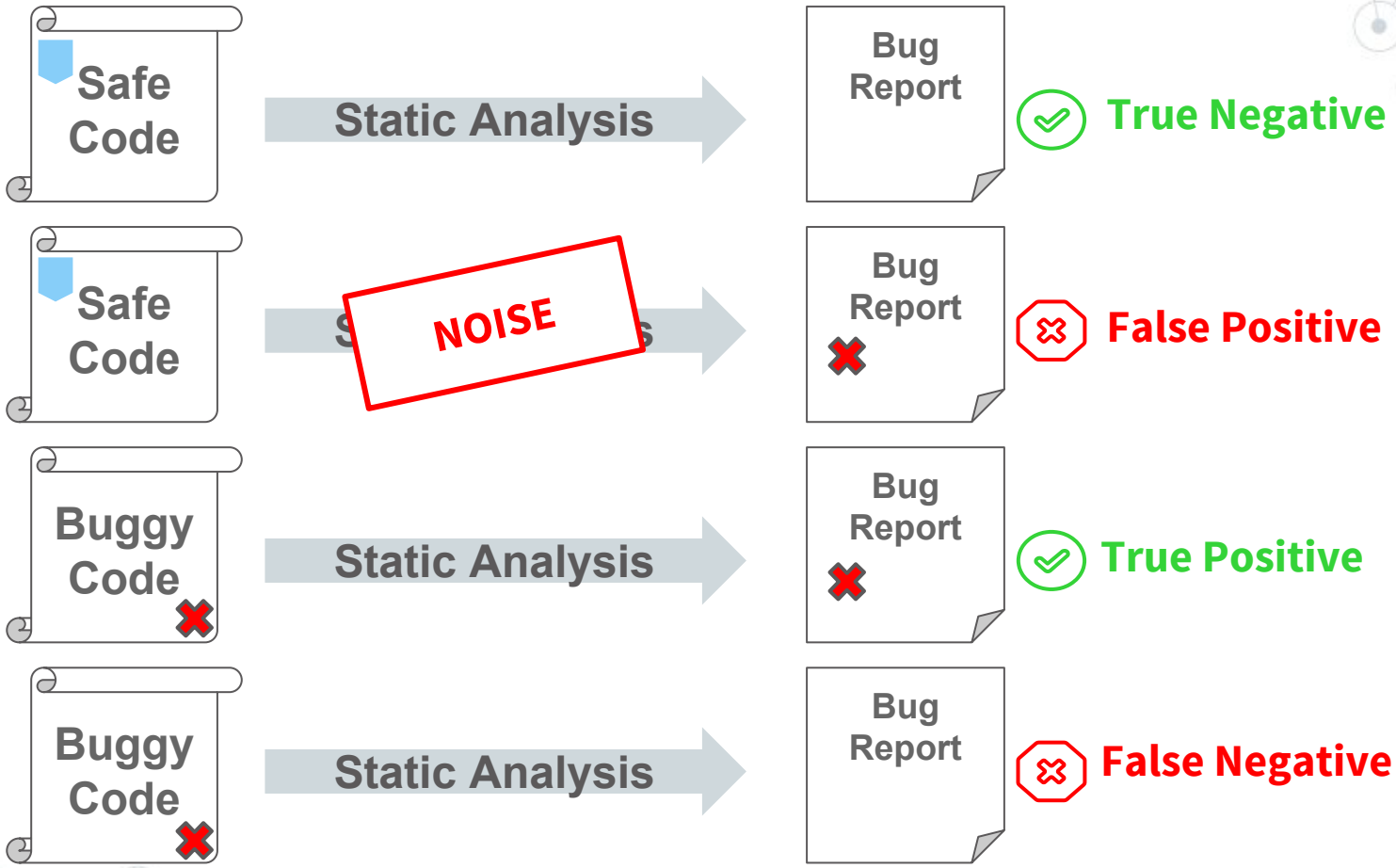
Static Analysis Testing



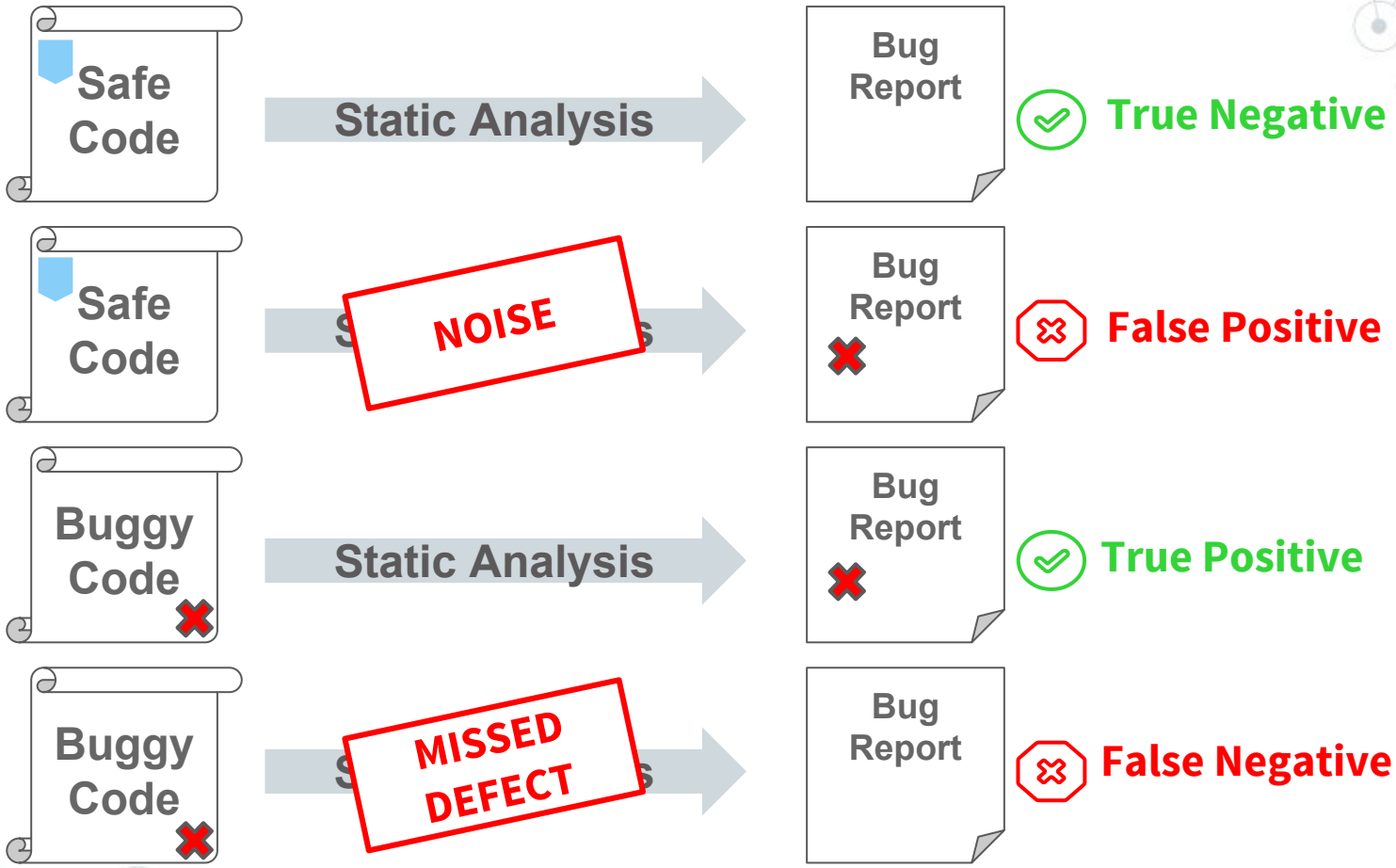
Static Analysis Testing



Static Analysis Testing



Static Analysis Testing



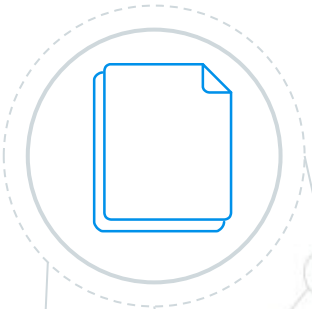
Pros and Cons

- Improves software assurance
- Saves time and money
- Takes customized rule sets

- False positive (noise)
- False negative (missed defects)
- Limited scope

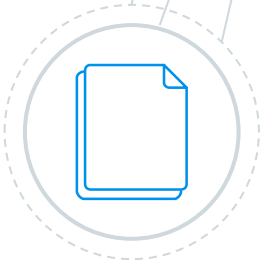
2.
Design of Test Cases

Test cases features



Test Cases Design

- Cover the most vulnerabilities possible
- Various complexities
- Statistically significant
- Ground truth
- Paired safe and flawed test cases
- Representative of production code



PHP Test Case Example

```php
<?php

    $input = $_POST['UserData'];


    $tainted = mysql_real_escape_string($input);


    $query = "SELECT * FROM student where id=". $tainted . "";

    $conn = mysql_connect('localhost', 'mysql_user', 'mysql_password');
    mysql_select_db('dbname') ;
    echo "query : ". $query ."<br /><br />" ;

    $res = mysql_query($query); //flaw

    while($data = mysql_fetch_array($res)){
        print_r($data) ;
        echo "<br />" ;
    }

    mysql_close($conn);
?>
```

PHP Test Case Example

```php
<?php

    $input = $_POST['UserData'];                                    INPUT


    $tainted = mysql_real_escape_string($input);


    $query = "SELECT * FROM student where id=". $tainted . "";

    $conn = mysql_connect('localhost', 'mysql_user', 'mysql_password');
    mysql_select_db('dbname') ;
    echo "query : ". $query ."<br /><br />" ;

    $res = mysql_query($query); //flaw

    while($data = mysql_fetch_array($res)){
        print_r($data) ;
        echo "<br />" ;
    }

    mysql_close($conn);
?>
```

PHP Test Case Example

```php
<?php

    $input = $_POST['UserData'];                    INPUT

    $tainted = mysql_real_escape_string($input);    FILTERING

    $query = "SELECT * FROM student where id=". $tainted . "";

    $conn = mysql_connect('localhost', 'mysql_user', 'mysql_password');
    mysql_select_db('dbname') ;
    echo "query : ". $query ."<br /><br />" ;

    $res = mysql_query($query); //flaw

    while($data = mysql_fetch_array($res)){
        print_r($data) ;
        echo "<br />" ;
    }

    mysql_close($conn);
?>
```

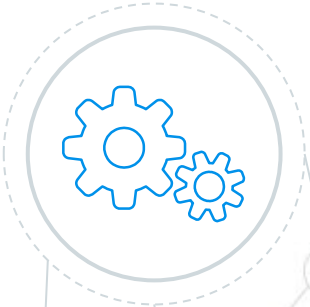
PHP Test Case Example

```php
<?php

    $input = $_POST['UserData'];                                    INPUT

    $tainted = mysql_real_escape_string($input);                    FILTERING

    $query = "SELECT * FROM student where id=". $tainted . "";

    $conn = mysql_connect('localhost', 'mysql_user', 'mysql_password');
    mysql_select_db('dbname') ;
    echo "query : ". $query ."<br /><br />" ;

    $res = mysql_query($query); //flaw                              SINK

    while($data = mysql_fetch_array($res)){
        print_r($data) ;
        echo "<br />" ;
    }

    mysql_close($conn);
?>
```

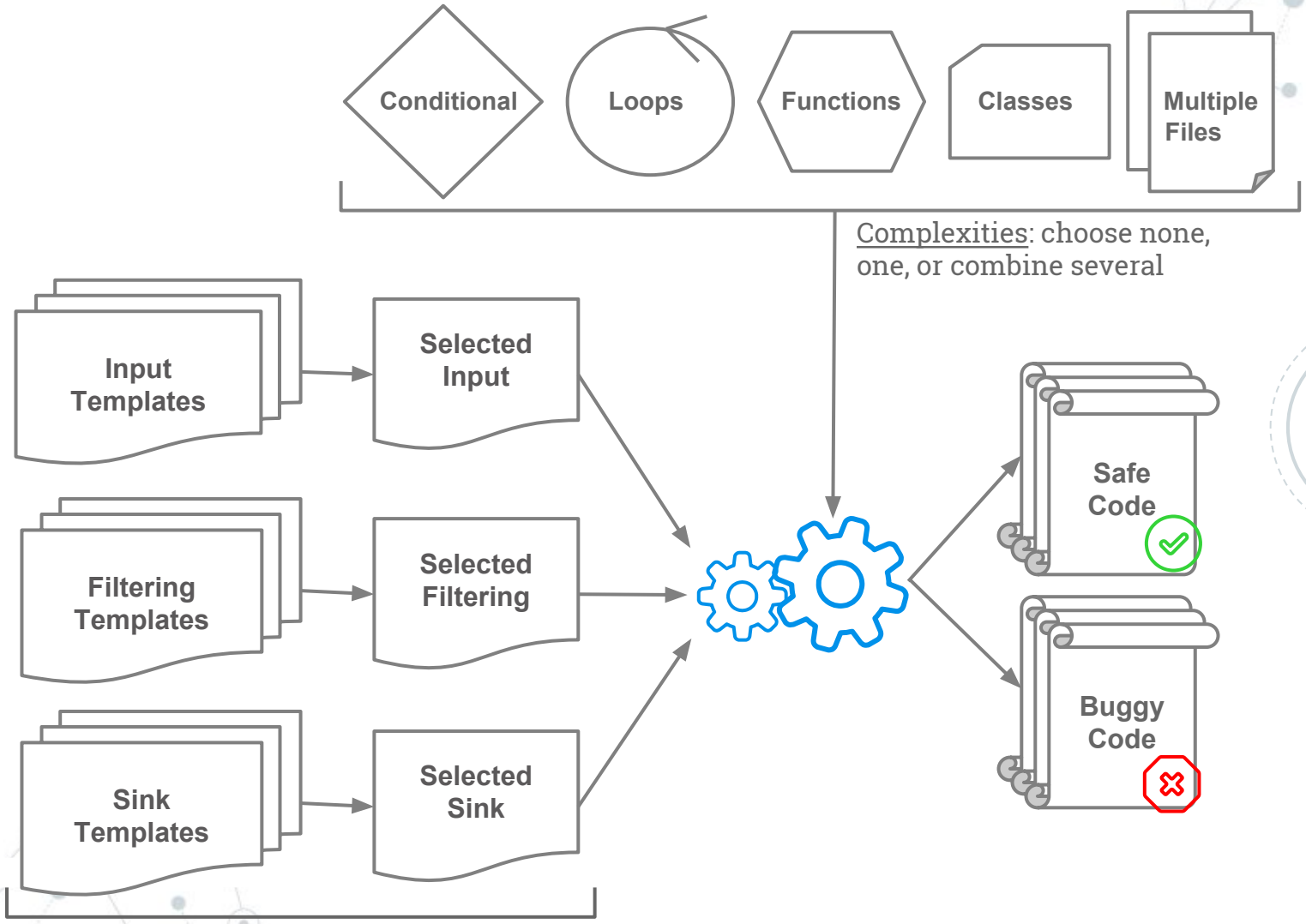
3.

PHP Vulnerability Test Cases Generator

Overview of the Test Cases generator

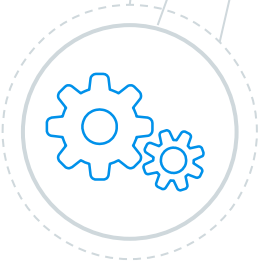


Test Cases Generator



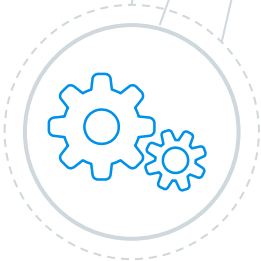
Test Cases Design

- Various complexities
- Statistically significant
- Ground truth
- Paired safe and flawed test cases
- Cover the more vulnerabilities possible
- Representative of production code



Vulnerabilities covered

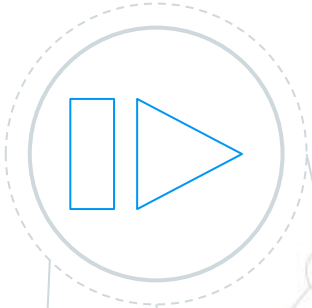
- Vulnerabilities based on OWASP Top 10 2013
  [#safe / #unsafe]
  - Injection [20912 / 5920]
  - Broken Authentication and Session Management
  - Cross Site Scripting (XSS) [5728 / 4352]
  - Insecure Direct Object References [400 / 80]
  - Security Misconfiguration [5 / 3]
  - Sensitive Data Exposure [5 / 7]
  - Missing Function Level Access Control
  - Cross-Site Request Forgery (CSRF)
  - Using Known Vulnerable Component
  - Unvalidated Redirects and Forwards [2208 / 2592]



4.

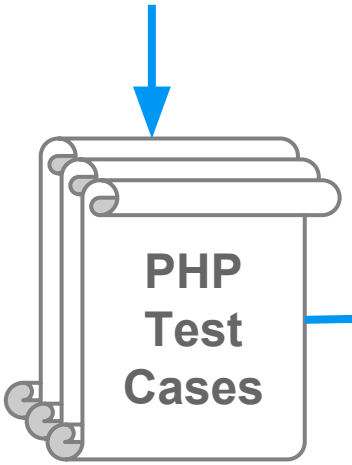
Live Demo

Generating Test Cases to Testing



Live Demo

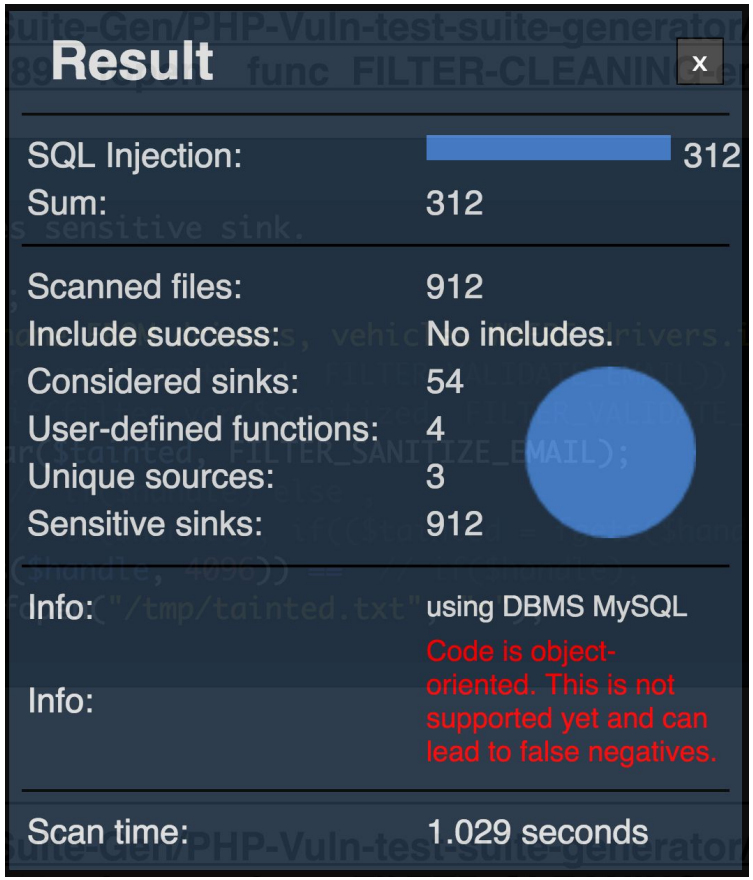
<?PHP Vulnerability Test Case Generator?>



RIPS - Metrics

Missed defects
- present : 912
- found : 312*

Recall = 312 / 912
= 34.2%



* considering all findings are True positives

RIPS - True Positive

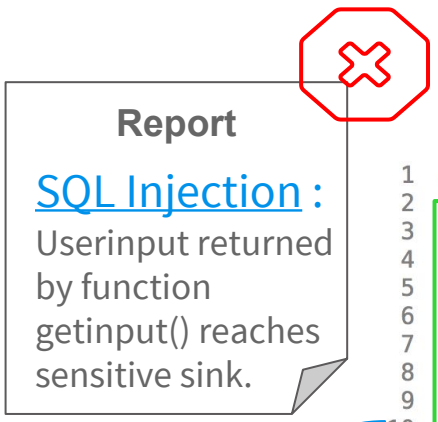
Report

SQL Injection :
Userinput reaches sensitive sink.

```
1  <?php
2
3  $tainted = $_GET['UserData'];                                    INPUT
4
5  //no_sanitizing                                                  FILTERING
6
7  $query = "SELECT * FROM COURSE c WHERE c.id IN (SELECT idcourse FROM
       REGISTRATION WHERE idstudent= $tainted )";
8
9  $conn = mysql_connect('localhost', 'mysql_user', 'mysql_password');
10 mysql_select_db('dbname') ;
11 echo "query : ". $query ."<br /><br />" ;
12
13 $res = mysql_query($query); //execution                           SINK
14
15 while($data =mysql_fetch_array($res)){
16 print_r($data) ;
17 echo "<br />" ;
18 }
19 mysql_close($conn);
20
21 ?>
```

CWE_89__GET__no_sanitizing__multiple_select-interpretation.php

RIPS - False Positive




```php
<?php

class Input{
  public function getInput(){
    return $_GET['UserData'] ;
  }
}

$temp = new Input();
$tainted =  $temp->getInput();

if(settype($tainted, "float"))
  $tainted = $tainted ;
else
  $tainted = 0.0 ;

$query = sprintf("SELECT Trim(a.FirstName) & ' ' & Trim(a.LastName) AS employee_name, a.city, a.
    street & (' ' +a.housenum) AS address FROM Employees AS a  WHERE a.supervisor=%u", $tainted);

$conn = mysql_connect('localhost', 'mysql_user', 'mysql_password');
mysql_select_db('dbname') ;
echo "query : ". $query ."<br /><br />" ;

$res = mysql_query($query); //execution

while($data =mysql_fetch_array($res)){
print_r($data) ;
echo "<br />" ;
}
mysql_close($conn);

?>
```

INPUT

FILTERING

SINK

CWE_89__object-directGet__CAST-func_settype_float__multiple_AS-sprintf_%u.php

Conclusion

- Tools need evaluation!
- Test cases need improvement
- PHP Vulnerability Test Suite Generator:
  - Automated generation
  - Modular and expandable
  - Customizable with options
  - 42 000 PHP test cases generated

Conclusion

- Tool is available on Github:

  https://github.com/stivalet/PHP-Vuln-test-suite-generator

- Test cases are hosted in the SARD:

  https://samate.nist.gov/SARD/view.php?tsID=103

- Project is already used by researchers:
  - M. K. Gupta, et al, “Security Vulnerabilities in Web Applications", JCSSE 2015
  - M. K. Gupta, et al, "XSSDM: Towards Detection and Mitigation of Cross-Site Scripting Vulnerabilities in Web Applications", ICACCI 2015
  - SATE VI - Static Analysis Tool Exposition, NIST 2016

Thanks!

Any questions?

Find us at:

http://samate.nist.gov

stivalet@nist.gov

Twitter: @B_Stivalet