

# 10th UCAAT

User Conference on  
Advanced Automated Testing

## Testability Tarpits: the Impact of Code Patterns on the Security Testing of Software Applications

Luca Compagna, SAP Security Research



14/11/2023



# About me and others that contributed...



Luca Compagna

**SAP** Security Research



Funded by EU project TESTABLE

**CISPA** HIGH LEVEL CENTER FOR INFORMATION SECURITY

**IMQ MINDED SECURITY**

Technische Universität Braunschweig

**NortonLifeLock**

**uc3m**

Pluribus One  
seeing one in many

**SAP**

**EURECOM** European Centre for Research in Advanced Technologies and Innovation

**ShiftLeft**

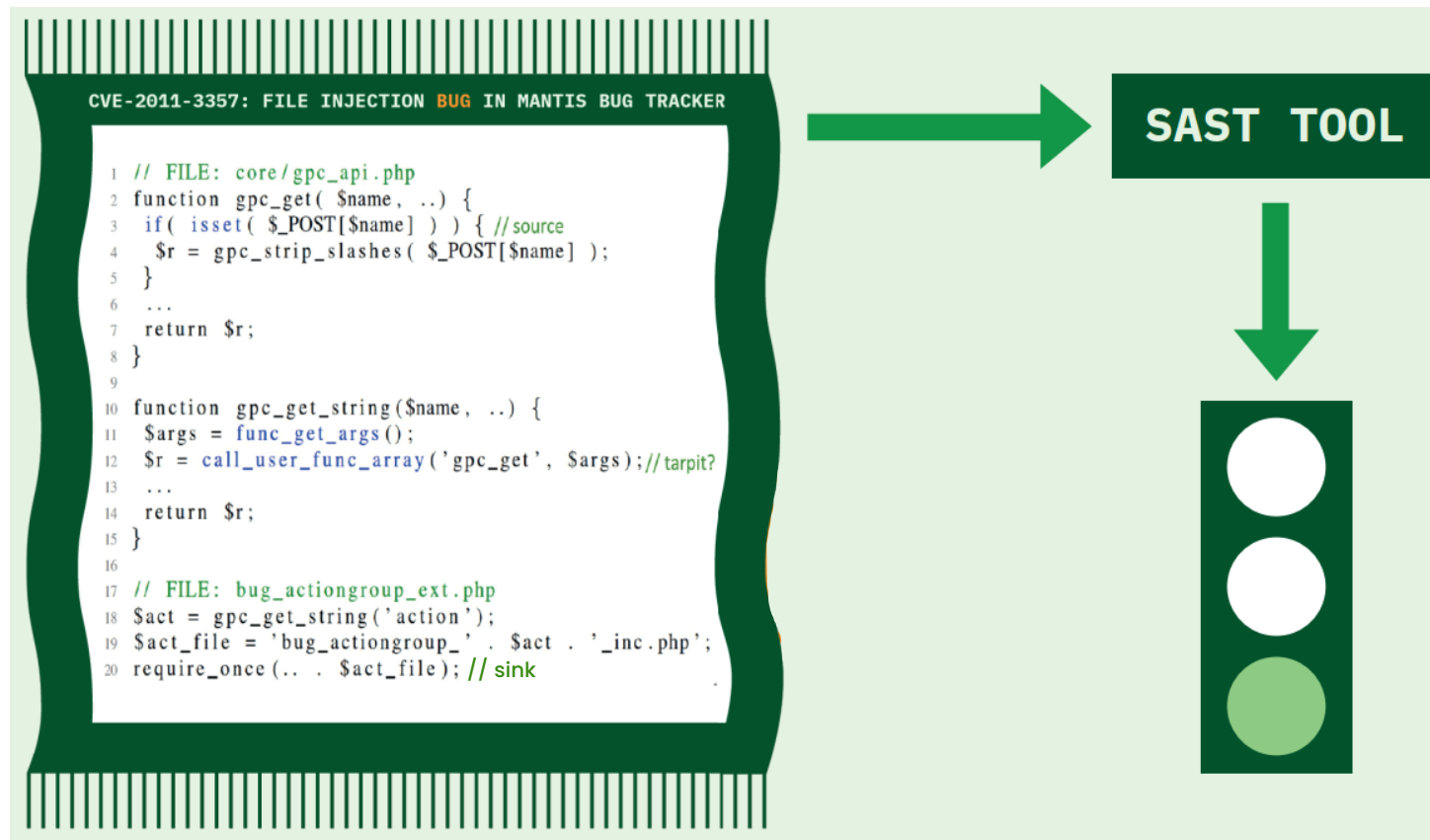
<https://www.testable.eu/>

*joint work with:*

Feras Al Kassar (SAP), Giulia Clerici (SAP), Fabian Yamaguchi (SHIFTLEFT), Davide Balzarotti (EURECOM)

# Context: SAST and testability

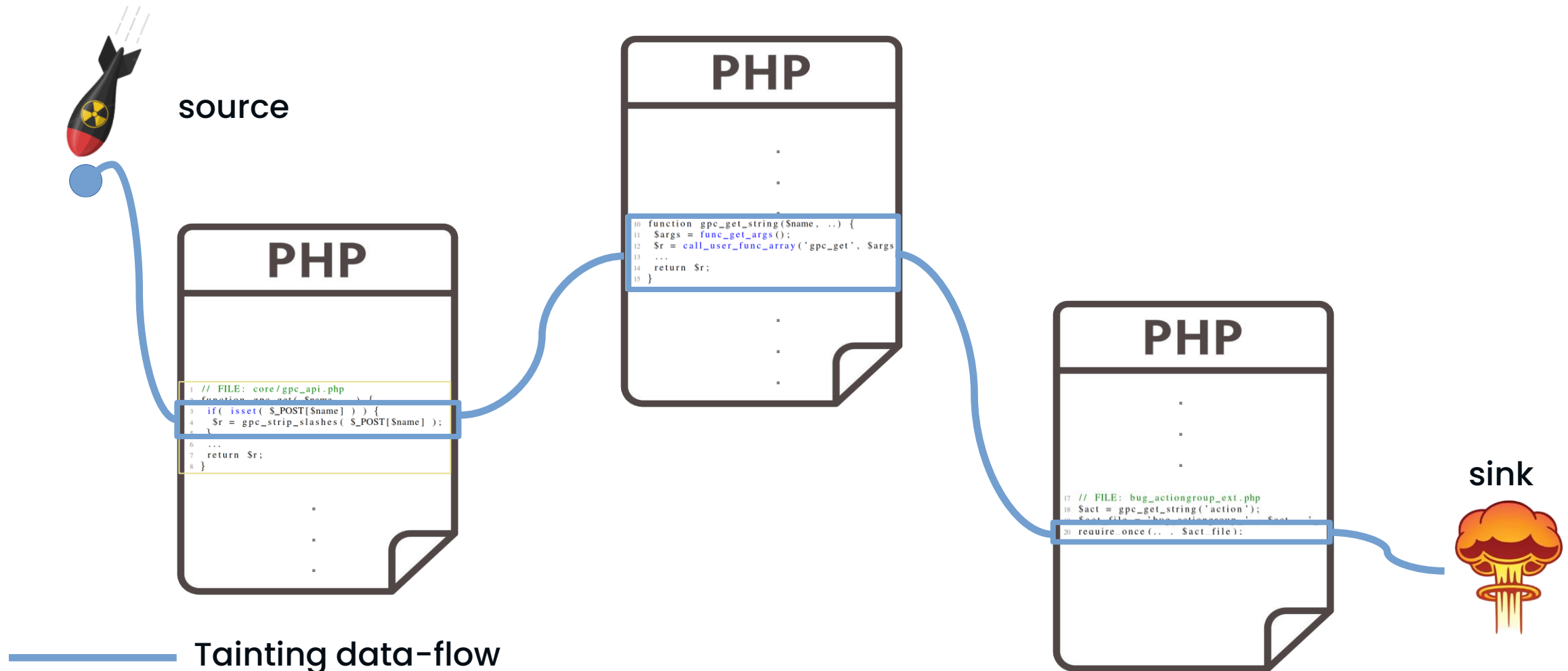
Static application security testing (SAST) is widely used in industry to detect vulnerabilities [1]



[1] [OWASP Code Review Guide v2.0](#), cf. Figure 1 and Figure 2

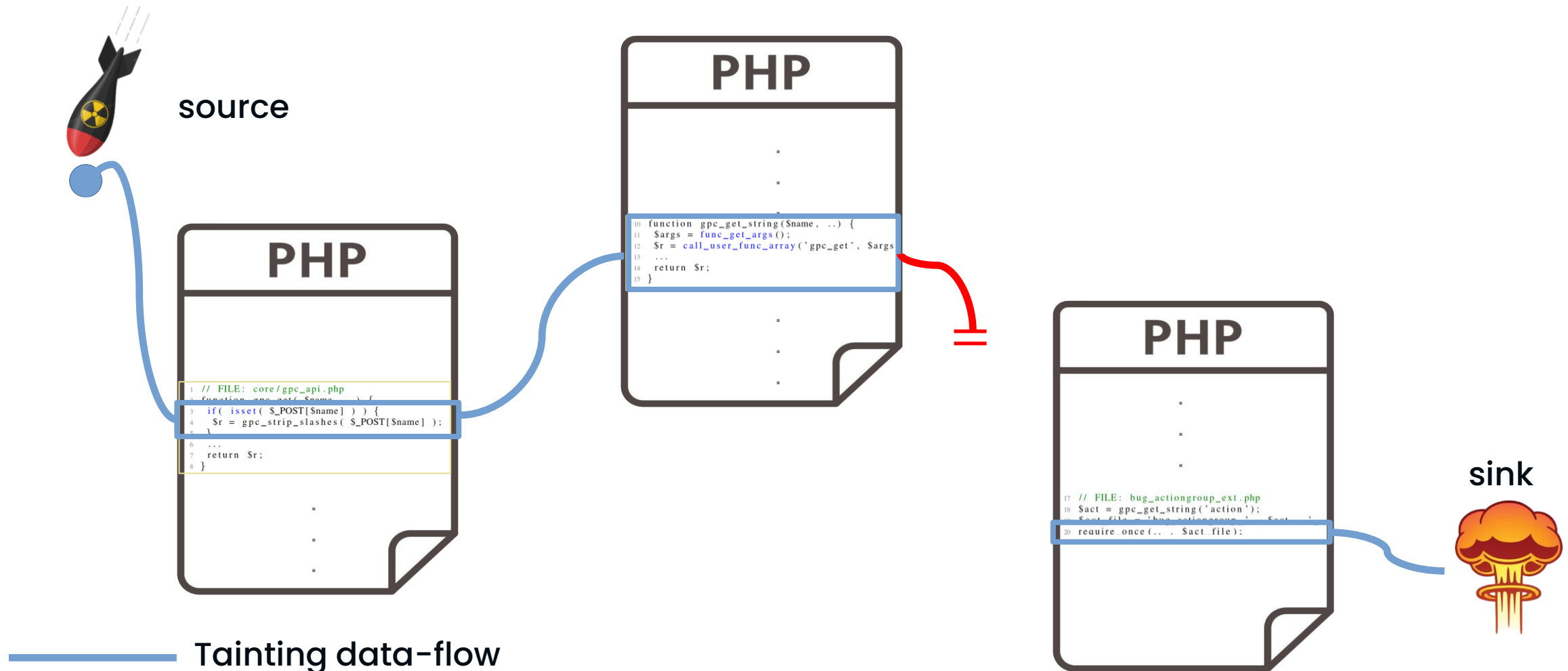
# Context: Injection vulnerabilities

Any attacker-controlled data (source) flowing in a dangerous operation (sink) without sanitization?



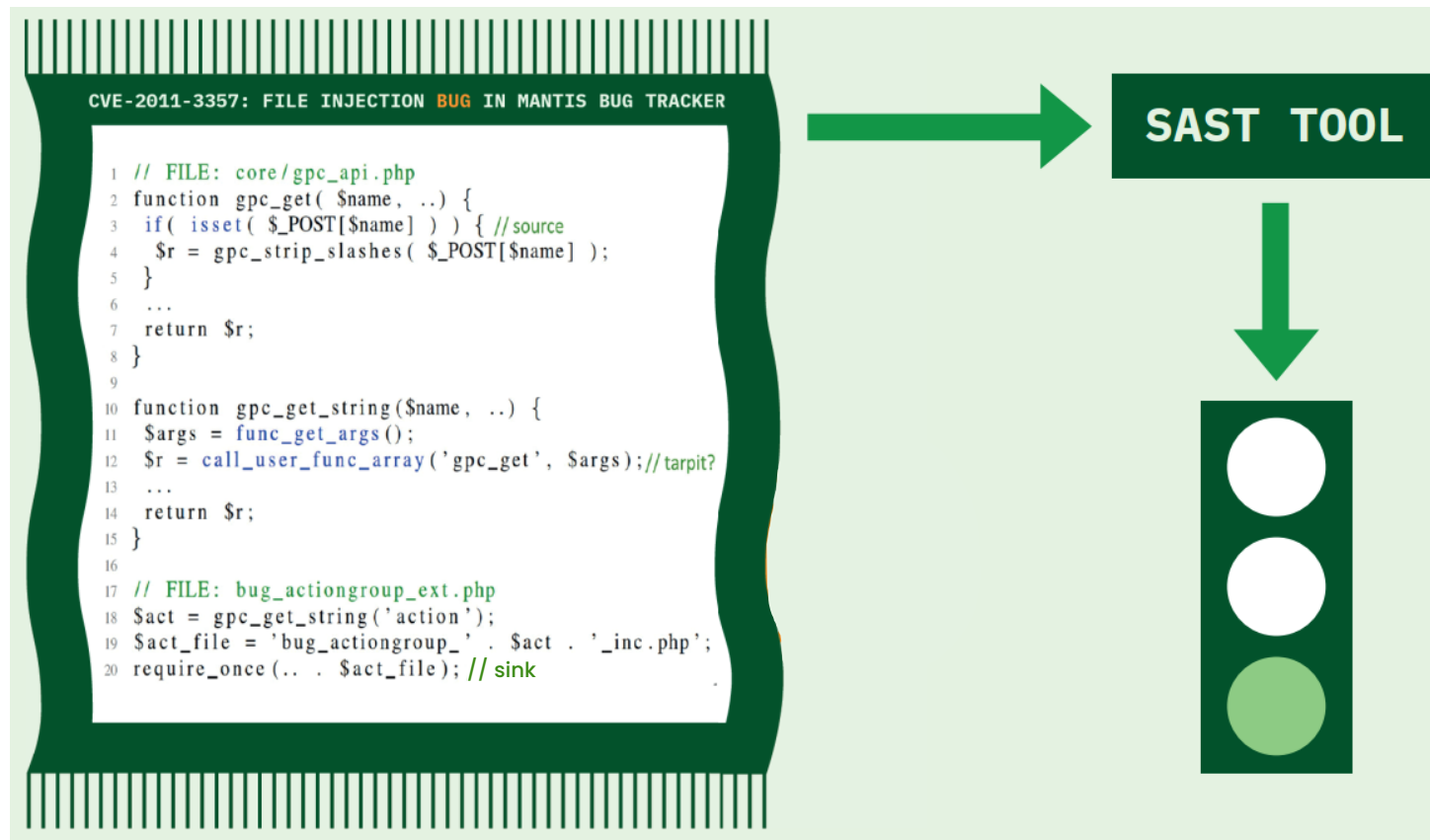
# Context: Injection vulnerabilities

Any attacker-controlled data (source) flowing in a dangerous operation (sink) without sanitization?



# Context: SAST and testability

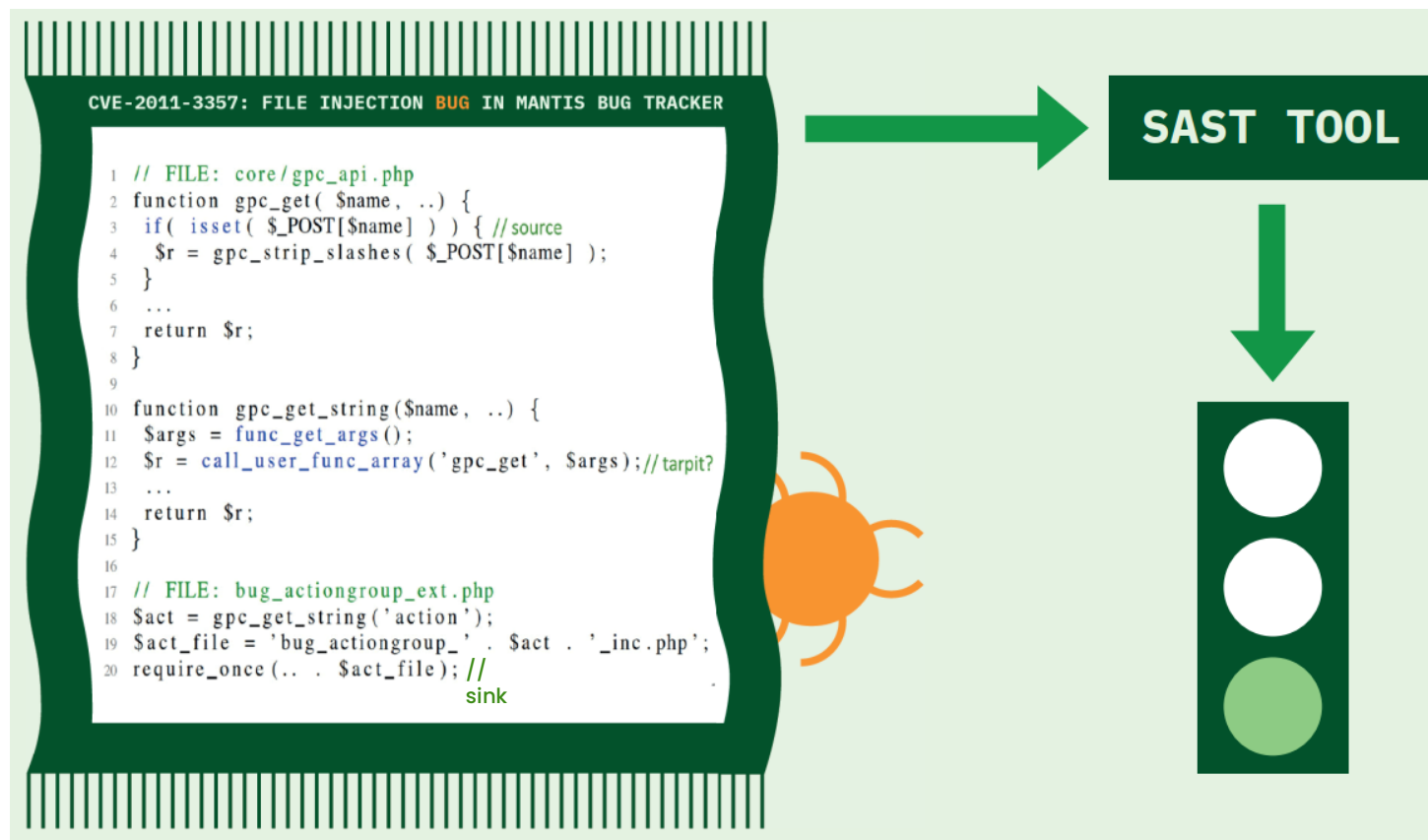
Static application security testing (SAST) is widely used in industry to detect vulnerabilities [1]



[1] [OWASP Code Review Guide v2.0](#), cf. Figure 1 and Figure 2

# Context: SAST and testability

Static application security testing (SAST) is widely used in industry to detect vulnerabilities [1]



Was all the code analyzed?  
No bugs under the carpet?

## Research questions

Code obstacles impacting SAST?  
Can we measure these obstacles?  
Can we discover/remediate them?

[1] [OWASP Code Review Guide v2.0](#), cf. Figure 1 and Figure 2

# CVE-2011-3357: File inclusion in mantis bug tracker

```
20 require_once (.. . $act_file); // sink
```



# CVE-2011-3357: File inclusion in mantis bug tracker

```
17 // FILE: bug_actiongroup_ext.php
18 $act = gpc_get_string('action');
19 $act_file = 'bug_actiongroup_' . $act . '_inc.php';
20 require_once(.. . $act_file); // sink
```

# CVE-2011-3357: File inclusion in mantis bug tracker

```
10 function gpc_get_string($name, ..) {
11     $args = func_get_args();
12     $r = call_user_func_array ('gpc_get', $args);
13     ...
14     return $r;
15 }
16
17 // FILE: bug_actiongroup_ext.php
18 $act = gpc_get_string('action');
19 $act_file = 'bug_actiongroup_' . $act . '_inc.php';
20 require_once(.. . $act_file); // sink
```

# CVE-2011-3357: File inclusion in mantis bug tracker

```
1 // FILE: core/gpc_api.php
2 function gpc_get( $name, ..) {
3     if( isset( $_POST[$name] ) ) {
4         $r = gpc_strip_slashes( $_POST[$name] );
5     }
6     ...
7     return $r;
8 }
9
10 function gpc_get_string( $name, ..) {
11     $args = func_get_args();
12     $r = call_user_func_array( 'gpc_get', $args );
13     ...
14     return $r;
15 }
16
17 // FILE: bug_actiongroup_ext.php
18 $act = gpc_get_string( 'action' );
19 $act_file = 'bug_actiongroup_' . $act . '_inc.php';
20 require_once( .. . $act_file ); // sink
```

# CVE-2011-3357: File inclusion in mantis bug tracker

```
1 // FILE: core/gpc_api.php
2 function gpc_get( $name, ..) {
3   if( isset( $_POST[$name] ) ) { // source
4     $r = gpc_strip_slashes( $_POST[$name] );
5   }
6   ...
7   return $r;
8 }
9
10 function gpc_get_string( $name, ..) {
11   $args = func_get_args();
12   $r = call_user_func_array( 'gpc_get', $args );
13   ...
14   return $r;
15 }
16
17 // FILE: bug_actiongroup_ext.php
18 $act = gpc_get_string( 'action' );
19 $act_file = 'bug_actiongroup_' . $act . '_inc.php';
20 require_once( .. . $act_file ); // sink
```

POST

<https://mantisb.com/service>

action=<ATTACKER\_PAYLOAD>

# Toward testability patterns for SAST

Many **SAST tools** (including commercial ones) do **not find** that **File inclusion**

```
1 // FILE: core/gpc_api.php
2 function gpc_get( $name, ..) {
3   if( isset( $_POST[$name] ) ) { //source
4     $r = gpc_strip_slashes( $_POST[$name] );
5   }
6   ...
7   return $r;
8 }
9
10 function gpc_get_string($name, ..) {
11   $args = func_get_args();
12   $r = call_user_func_array('gpc_get', $args); //obstacle?
13   ...
14   return $r;
15 }
16
17 // FILE: bug_actiongroup_ext.php
18 $act = gpc_get_string('action');
19 $act_file = 'bug_actiongroup_' . $act . '_inc.php';
20 require_once( .. . $act_file); //sink
```

vulnerable app

pattern  
creation (1)



```
// replace with
// code companion for the obstacle
//
$a = $_GET["p1"]; // source
$b = $a // replace with obstacle!
echo $b; // sink
```

testability pattern skeleton (baseline XSS)

# Toward testability patterns for SAST

Many **SAST tools** (including commercial ones) do **not find** that **File inclusion**

```

1 // FILE: core/gpc_api.php
2 function gpc_get( $name, ..) {
3   if( isset( $_POST[$name] ) ) { //source
4     $r = gpc_strip_slashes( $_POST[$name] );
5   }
6   ...
7   return $r;
8 }
9
10 function gpc_get_string($name, ..) {
11   $args = func_get_args();
12   $r = call_user_func_array('gpc_get', $args); //obstacle?
13   ...
14   return $r;
15 }
16
17 // FILE: bug_actiongroup_ext.php
18 $act = gpc_get_string('action');
19 $act_file = 'bug_actiongroup_' . $act . '_inc.php';
20 require_once( .. . $act_file); //sink

```

vulnerable app

pattern creation (1)



```

function F($var) { // code companion
  return $var; // for the obstacle
} //
$a = $_GET["p1"]; // source
$b = call_user_func_array("F", [$a]); // obstacle
echo $b; // sink

```

testability pattern instance

pattern transformation (3)



```

12 $r = gpc_get(...$args); // no obstacle anymore

```

SAST measurement (2)



SAST	Correct
RIPS	NO
phpSAFE	NO
WAP	NO
Progpilot	YES
Comm_2	NO
Comm_1	YES

After that **transformation**, commercial tool **Comm\_2 finds** the **File inclusion!**

# Many variants of that pattern can be created...

```
function F($var) { // code companion
    return $var; // for the obstacle
} //
$a = $_GET["p1"]; // source
$b = call_user_func_array("F", [$a]); // obstacle
echo $b; // sink
```

instance 1



```
function F($var) { // code companion
    return $var; // for the obstacle
} //
$a = $_GET["p1"]; // source
$f = "F"; // obstacle
$b = call_user_func_array($f, [$a]); // obstacle
echo $b; // sink
```

instance 2

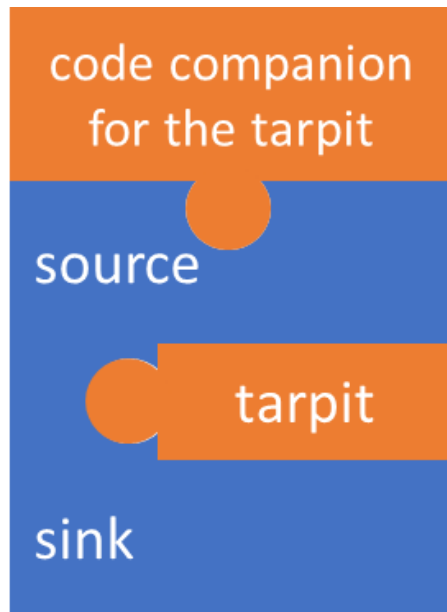


```
function F_1($var) { // code companion
    return $var; // for the obstacle
} //
function F_2($var) { //
    return "foo"; //
} //
$a = $_GET["p1"]; // source
$f = $_GET["Func_id"]; // obstacle
$b = call_user_func_array("F_".$f, [$a]); // obstacle
echo $b; // sink
```

instance 3



...



# Testability Patterns for SAST

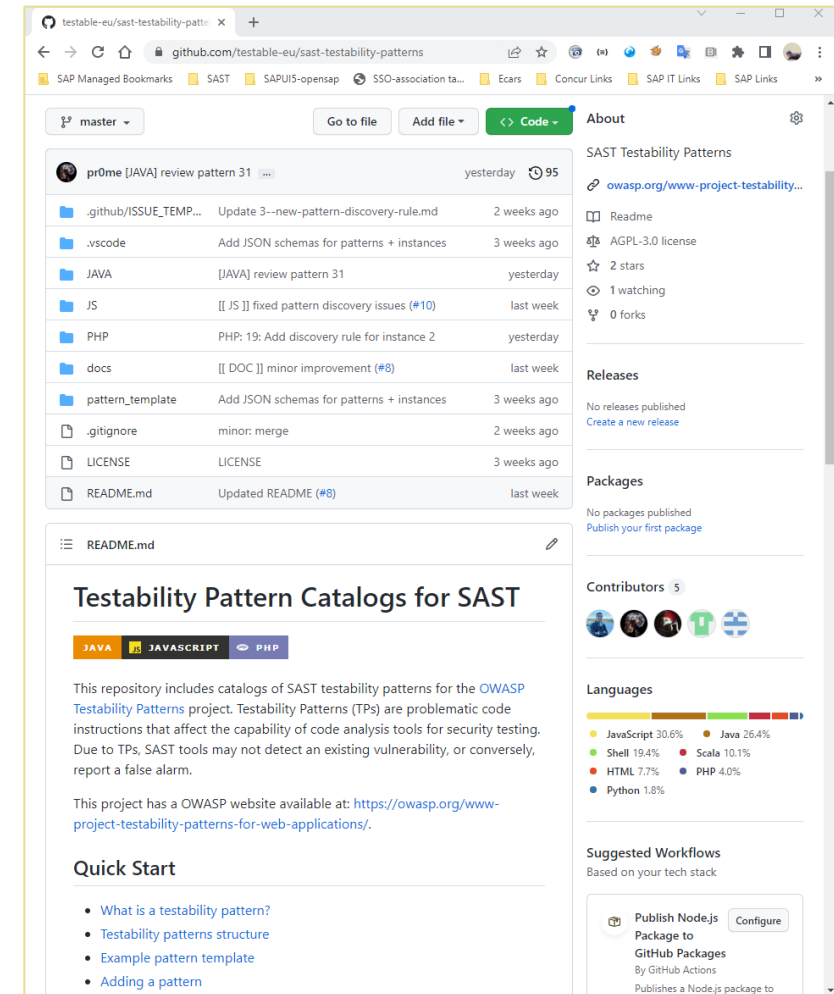
Targeted **3** popular **languages**

- **PHP**: ~120 pattern instances
- **JS**: ~150 pattern instances
- **Java**: ~200 pattern instances

We inspected the entire language manual

What do we want to do with them?

- SAST tools **measurement**
- **Discover** patterns into applications
- Remediate patterns to increase testability



<https://github.com/testable-eu/sast-testability-patterns>



# Testability Patterns Framework for SAST

Framework provides operations for

- SAST tools **measurement** (e.g., codeql integrated)
- **Discover** patterns into apps (via Joern and Scala queries)

```
tpframework measure -l JS -p 1 2 --tools codeql:2.9.2
```

```
tpframework discovery -t MYAPP/ -l PHP -a --tools T1:V1 T2:V2
```

## Results spoiler

1. Measurement: many SAST tools struggle on our patterns
2. Discovery: many apps in Github use these patterns
3. Testability for SAST can be improved!

The screenshot shows the GitHub repository page for `testable-eu/sast-tp-framework`. The commit history table lists recent changes, including updates to measurement JSON files, SAST tool integration, and documentation. The README section provides an overview of the framework's purpose and supported languages.

Commit Message	Author	Time
Fixed content of measurement json files	edsbca	20 hours ago
add vsconfig for json schema		3 weeks ago
SAST measurement: vuln type matching corrected		yesterday
fixes #7		2 months ago
[[ docs ]] fix misplaced content in templates for discovery rules		3 days ago
Fixed content of measurement json files		20 hours ago
update testability patterns submodule		4 days ago
Fixed content of measurement json files		20 hours ago
Testability_pattern submodule set to public url		3 months ago
add vsconfig for json schema		3 weeks ago
Testability_pattern submodule set to public url		3 months ago
joern installation fixes		last week
[[ docs ]] reformatted the structure		3 weeks ago
Update README.md		3 weeks ago
Added console logging + improved logging for check_discov...		last week
Fixes #30		last week
Testability_pattern submodule set to public url		3 months ago
minor typo in docker-compose		3 months ago
[Docker] added example.env + updated documentation (#3)		3 months ago
Fixes #30: correcting pytest.ini		4 days ago
Fixes #30		4 days ago
Testability_pattern submodule set to public url		3 months ago
Testability_pattern submodule set to public url		3 months ago

**TP-Framework: Testability Pattern Framework for SAST**

python 3.10 | dockerized yes | License: GPLv3

TP-Framework relies on `testability patterns` to reduce false positive/negative rate in SAST analysis over supported programming languages. Testability patterns are code patterns that make difficult for SAST tools to detect a vulnerability.

TP-Framework enables operations such as:

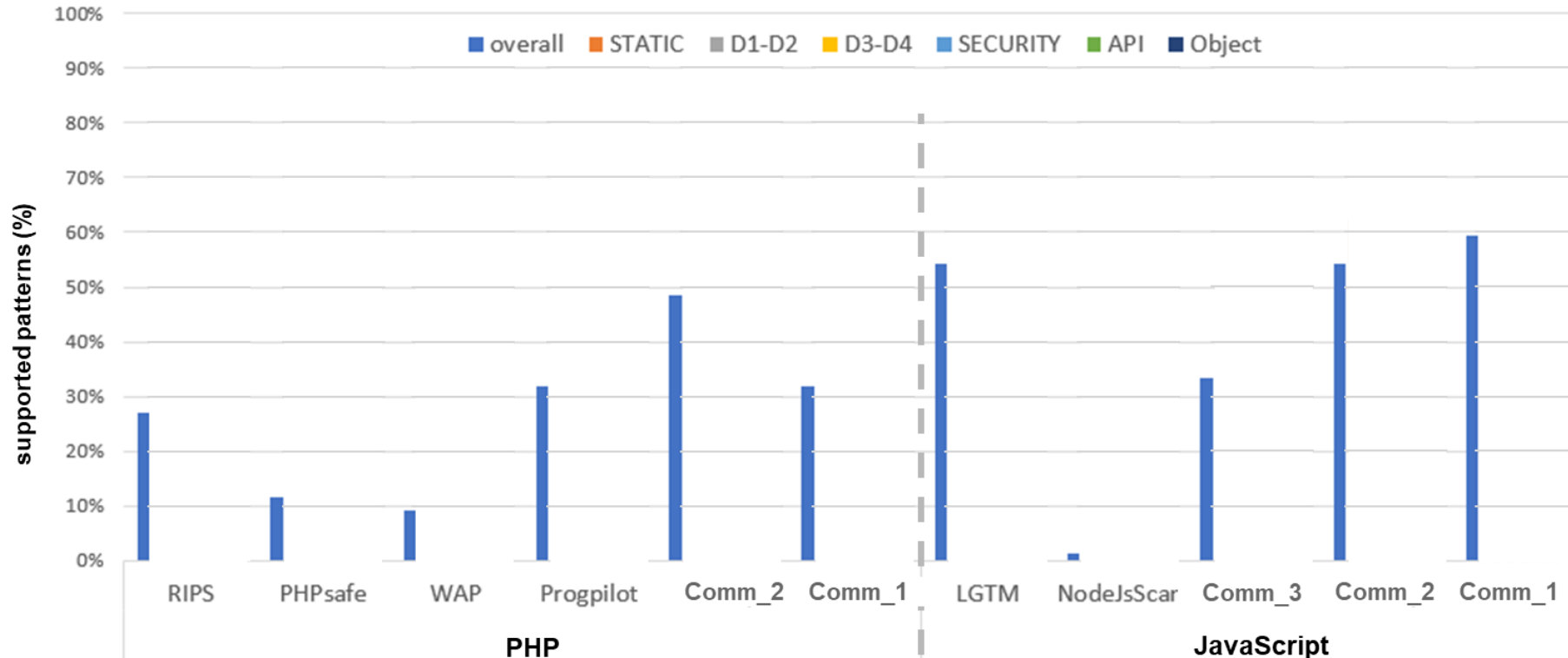
- measurement of SAST tools against a catalog of testability patterns, and
- discovery of testability patterns within application source code

<https://github.com/testable-eu/sast-tp-framework>

# 1. Measurement: many SAST tools struggle on our patterns

## Measured our pattern instances against SAST

- Overall: <50% support for PHP and <60% for JS
- Tools Combination: 66% PHP, 82% JS

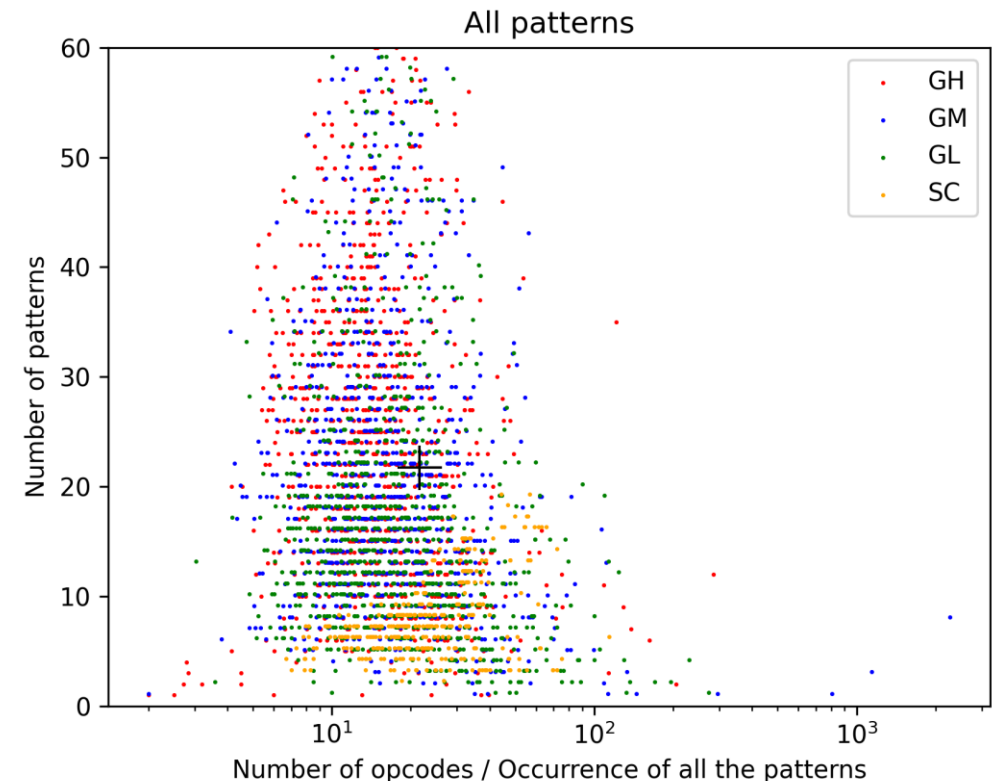


## 2. Discovery: many apps in Github use these patterns

**PHP**: created **discovery rules** for our patterns and run them over **>3000** open-source PHP **apps** (from Github and Sourcecodester)

Our patterns are very **prevalent** in the real world

	Unique obstacles per Project	obstacles per LoC
AVG	21	20

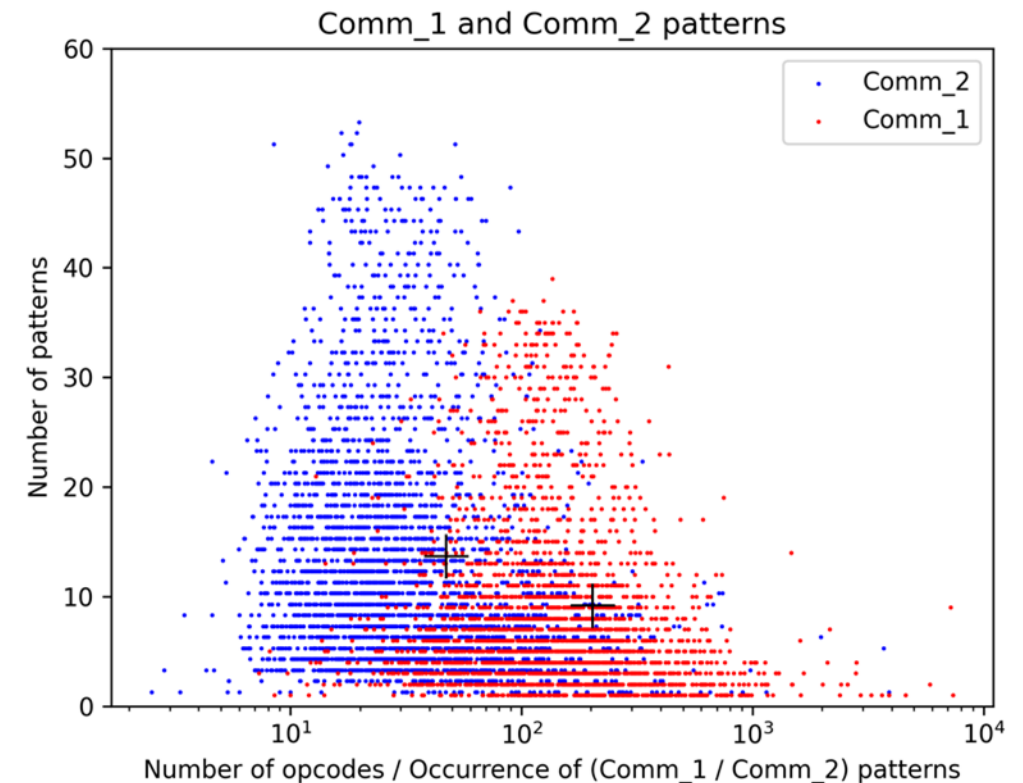


## 2. Discovery: many apps in Github use these patterns

**PHP**: created **discovery rules** for our patterns and run them over **>3000** open-source PHP **apps** (from Github and Sourcecodester)

Our patterns are very **prevalent** in the real world

	Unique obstacles per Project	obstacles per LoC
AVG	21	20
Comm_1	8	203
Comm_2	13	47



### 3. Testability for SAST can be improved!

**Remediation 1:** Two **transformation experiments** targeting PHP and JS

- transformations intended as code rewriting for obstacles
- **>9000 new alerts: 370 true positives** in 48 apps (over ~2700 alerts inspected)
- **182 true positives already confirmed** from 31 projects: 38 impacting popular Github projects

**Remediation 2:** **improve SAST tools** (e.g., our paper at USENIX 2023)

**Remediation 3:** provide **custom rules** to make SAST tools overcoming obstacles (*on-going work*)

# A new OWASP project: journey started

Targeting the **Testability** dimension

First concrete result: **Testability Patterns for SAST**

- <https://github.com/testable-eu/sast-testability-patterns>
- <https://github.com/testable-eu/sast-tp-framework>

Interested to contribute with your valuable expertise?

<https://owasp.org/www-project-testability-patterns-for-web-applications/>

Let us devise OWASP top 10 testability patterns for SAST!

Can we do the same for DAST, Privacy, ML?

The screenshot shows the GitHub repository page for 'OWASP Testability Patterns for Web Applications'. The page includes a navigation bar with 'OWASP.' and a search icon. Below the title, there are statistics for 'Watch' (2) and 'Star' (4). A 'Main' tab is selected, with other tabs for 'JAVA', 'JAVASCRIPT', and 'PHP'. The main content area features the project title, a description of the project's goal to improve security through testability, and a list of target audiences: Web and AI/ML Developers, Managers, and Security Teams. An 'Objective' section describes the project's aim to deliver tools and methodologies. A sidebar on the right provides information about the OWASP Foundation and lists related projects like Incubator Project, Breaker, Builder, and Tool, along with the current version (0.0.1) and social links.



**Any further questions?**

