



Testability Pattern-driven Web Application Security and Privacy Testing

Presented by:

Soheil Khodayari & Lukas Seidel

OBO. TESTABLE Consortium



testable.eu



@Testable_EU



pellegrino@cispa.de

Projects to Policy Seminar | 30.06.2022

Problem Statement



- Web vulnerabilities have critical consequences for the society
 - Over 4.8 billion websites online, 1.7 billion users ¹
 - Personal data leaks, financial loss, ...



Banking



Shopping



Education

- The complexity of web applications are rising



Problem: existing approaches fall short of capturing this ever-increasing complexity

- TESTABLE addresses the grand challenge of:

Building and maintaining modern web-based and AI-powered systems **secure** and **privacy-friendly**

¹ internetlivestats.com

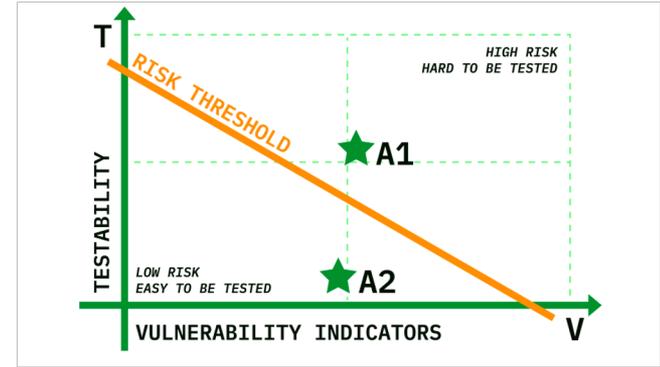
Our Vision



I. New Testability Metric

- Existing methodologies for measuring the probability that a web application contains risky behaviors rely on indicators:
 - Code size/complexity, presence of sensitive function calls, ...

Lowering indicators is challenging, costly, not always necessary, or even impossible!



- TESTABLE proposes a **precise risk score metric**:
 - New **bi-dimensional** metric with the novel notion of webapps' **testability** w.r.t. testing techniques
 - Compute testability via **measurable** and (possibly) **transformable testability patterns**

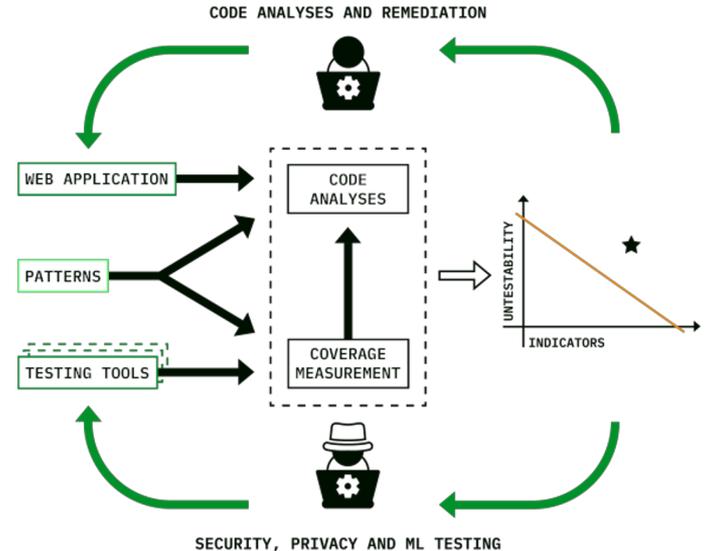
Our Vision (Cont'd)

I. New Testability Metric

II. New Decision and Action Space

The new testability metric provides a **natural way to improve the security and privacy** of webapps:

- **Optimize testing** by focusing on problematic components
- **Refractor design and code** to increase code testability
- **Defense in-depth layers** when no other actions are viable



Our Vision (Cont'd)

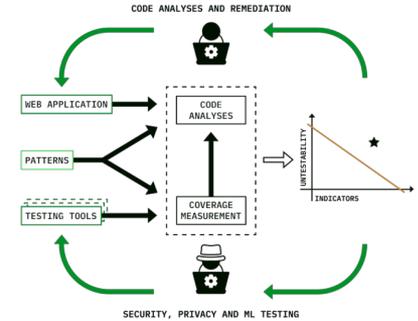


I. New Testability Metric

II. New Decision and Action Space

III. New Comprehensive and Effective Testing Techniques

- Existing techniques focus on security, privacy issues often overlooked
- Reduced effectiveness due to rapid integration of ML components in webapps



Testable Domain Areas:

- **Security:** improve **security testing techniques** (SAST, DAST, and HAST)
- **Privacy:** design novel techniques to test for **privacy-related problems**
- **ML:** develop new techniques to enable **security and privacy testing of AI/ML modules**

Target Users and Policy Benefits



Product Managers

More precise risk estimate quantifying the security and privacy risks of a program



Product Developers

Better and flexible tools to improve testability, reducing security and privacy **risk exposure**



Security Teams

More **accurate and effective** security, privacy, and AI/ML **testing techniques** and tools



Standardization Bodies

Collection of **testability patterns and best practices** for de-facto standards

TESTABLE Over SAST: A First Story

No bugs under the carpet? Testability for SAST: good?

```

CVE-2011-3357: FILE INJECTION BUG 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);//tarpit?
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
    
```

SAST TOOL



Pattern creation (1)

```

function f($var) { // CODE COMPANION
    return $var; // FOR THE TARPIT
}

Sa = $_GET["p1"]; // SOURCE

Sb = call_user_func_array("f", [$a]); // TARPIT

echo $b; // SINK
    
```

SAST measurement (2)

SAST	CORRECT
RIPS	NO
phpSAFE	NO
WAP	NO
Progpilot	YES
Comm1	NO
Comm2	YES

Pattern discovery and transformation (3)

```

// SAST tools detect File
// injection, by replacing
// the tarpit discovered
// in line 12 with:
12 $r = gpc_get($args);
    
```

Created Many Patterns/Tarpits

- PHP: ~120 pattern instances
- JS: ~150 pattern instances

SAST Tools Measurement: A lot to Improve

- PHP: <50% support rate
- JS: ~60% support rate

Our Tarpits are Highly Prevalent

- created tarpit discovery rules
- analyzed >3000 PHP apps
- in AVG: 21 tarpit per app, one tarpit every 20 LoC

Refactoring Tarpits Increase Application Testability

- two exps: manual and automated refactoring
- >400 new vulnerabilities emerged upon refactoring (~200 already confirmed)



Project Events & Dissemination

- Organisation of webinar series, invited talks and seminars
- Dissemination in large internal events of the industrial project partners (SAP, NortonLifeLock)
- Participation in the organization of scientific events (conferences, workshops)
- Participation in OWASP events, and large cybersecurity conferences (e.g., BlackHat, DEFCON, etc)
- Liaison with SPARTA and Concordia competence networks
- TESTABLE open source framework (hosted by OWASP)
- TESTABLE results integrated in industrial products and development processes (Minded Security)