Leveraging Flawed PHP Tutorials for Seeding Large-Scale Web Vulnerability Discovery

Tommi Unruh, Bhargava Shastry, Malte Skoruppa, Federico Maggi, Konrad Rieck, Jean-Pierre Seifert, and Fabian Yamaguchi
Origins

Oakland’14: Modeling and Discovering Vulnerabilities with Code Property Graphs

Joern

Euro S&P’17: Efficient and Flexible Discovery of PHP Application Vulnerabilities

Joern for PHP
Hypothesis: Vulnerabilities in popular tutorials propagate to production code

Our proposal

Use **pattern mining** to:

- Examine hypothesis
- **Scale up** vulnerability search
Key Results

- **64,415** repos scanned, **117** vulnerabilities
  - Hypothesis validated!

- **8 SQLi** vulnerabilities traced to a **single** tutorial!
- Used a standard **PC** and broadband **DSL**
  - Low barrier to entry!
Motivation

Manual audit of popular PHP tutorials betrayed lack of security understanding

If developers borrow code, they borrow vulns
Design

Vulnerable Tutorial ➔ Template ➔ Gremlin_{G=(V,E)} ➔ GitHub ➔ Analogue

\[ a = \_p[a] \] 
\[ \text{mysql}_q(a) \]

\[ b = \_p[x] \] 
\[ \text{mysql}_q(b) \]
Example

```php
$title=$_POST["title"];
$result=mysql_query("SELECT * FROM wp_posts where; post_title like '%$title%' and post_status='publish'"),
$found=mysql_num_rows($result):
```

**Diagram:**

```
AST
  ASSIGN
    $title
  ASSIGN
    $result
  CALL
    mysql...
  ASSIGN
    $found
  CALL
    mysql...
  $result

AST
  ASSIGN
    $_POST
  CALL
    mysql...
  CALL
    mysql...
```

**Example Text:**

```
$title=$_POST["title"];
$result=mysql_query("SELECT * FROM wp_posts where;
    post_title like '%$title%' and post_status='publish'"),
$found=mysql_num_rows($result):
```
Implementation

- Template generation ⇒ Lightweight PHP parser
- Traversals ⇒ Gremlin
- Python GitHub Crawler
- Code serialization ⇒ Joern for PHP

https://github.com/tommiu/ccdetection
## Results

<table>
<thead>
<tr>
<th>Data set</th>
<th>Size</th>
<th>Num. Analogues</th>
<th>Num. Vulnerabilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not popular</td>
<td>42,064</td>
<td>269</td>
<td>80 (29.74%)</td>
</tr>
<tr>
<td>Popular</td>
<td>16,037</td>
<td>528</td>
<td>35 (6.63%)</td>
</tr>
<tr>
<td>Very popular</td>
<td>6,314</td>
<td>23</td>
<td>2 (8.7%)</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>64,415</strong></td>
<td><strong>820</strong></td>
<td><strong>117 (14.27%)</strong></td>
</tr>
</tbody>
</table>
Insights

- Traversals efficient for scaling up analysis
- Structural analysis (AST) robust
- Run time for top 10 PHP projects low
- Standard desktop PC ⇒ $19s < t < 53\text{ m}$
Summary

- Developers consult informal documentation
- Poorly written tutorials may put software at risk
- Finding vulnerabilities from tutorials is possible
Future Work

- Language agnostic analogue detection
- Plug-in for IDEs such as Eclipse
Code

Joern

https://github.com/octopus-platform/joern

PHP Joern

https://github.com/malteskoruppa/phpjoern

GitHub Spider

https://github.com/tommiu/GithubSpider
Questions?
Related Work

- **Code clone finders**
  - Code borrowed from tutorials likely lexically different
  - Lexical matching $\Rightarrow$ **False negatives**

- **Vulnerability scanners**
  - **Not** yet another PHP vuln scanner
  - Intended to shed light on unsafe coding practices