

Simulation of Built-in PHP Features for Precise Static Code Analysis

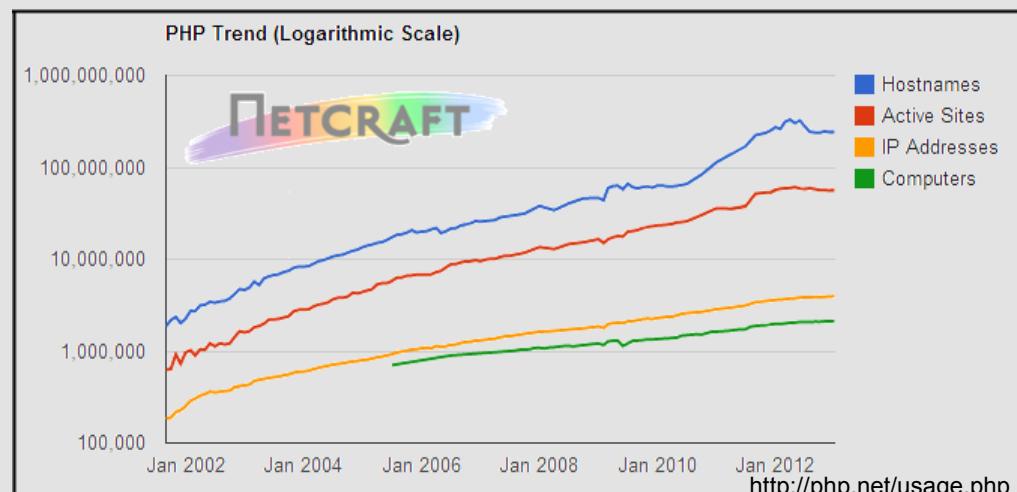
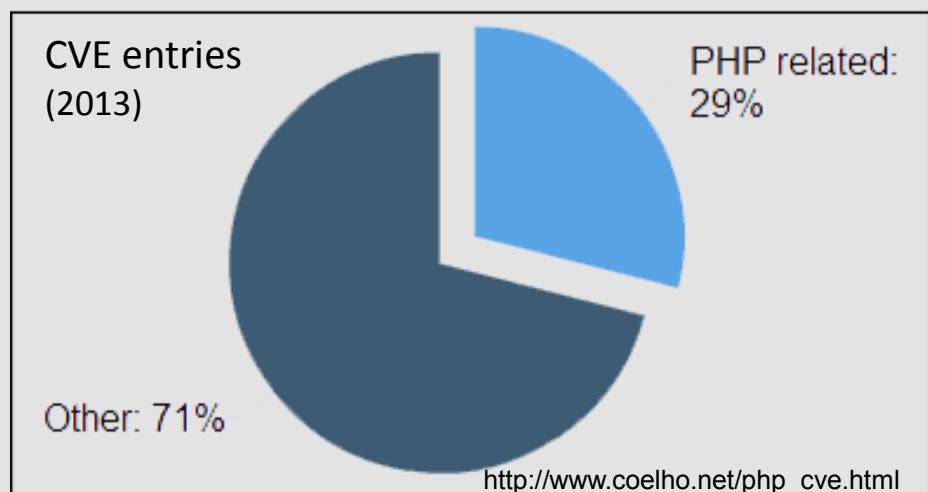
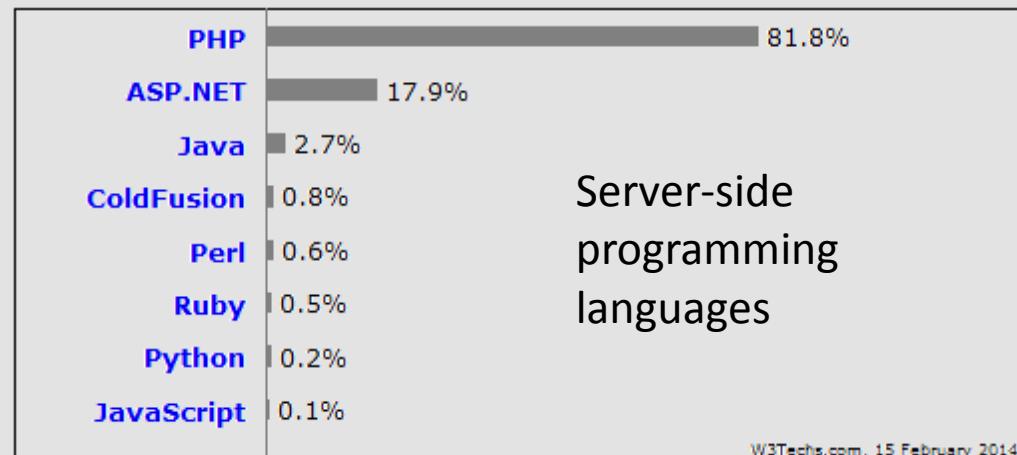
Johannes Dahse and Thorsten Holz
Ruhr-University Bochum

NDSS '14, 23-26 February 2014, San Diego, CA, USA

Simulation of Built-in PHP Features for Precise Static Code Analysis

1. Introduction
2. Implementation
3. Evaluation
4. Conclusion

php is everywhere.



Simulation of Built-in PHP Features for Precise Static Code Analysis

1. Introduction
2. Implementation
3. Evaluation
4. Conclusion

The screenshot shows a web browser window with the title "Sign in - NDSS 2014". The URL bar displays the address <https://ndss2014.ece.cmu.edu/crp/ndss2014/index.php>. A red circle highlights the URL. The main content area shows the "NDSS 2014 Sign in" page. It includes fields for "Email" and "Password", and radio buttons for "Sign me in", "I forgot my password, email it to me", and "I'm a new user and want to create an account using this email address". A "Sign in" button is present. To the right, a sidebar titled "Conference information" lists "Deadlines", "Program committee", and "Conference site", along with a note that "55 papers were accepted out of 293 submitted". At the bottom, a message states "Submissions: The deadline for registering new papers has passed." A footer at the bottom right reads "HotCRP Conference Management Software".

Sign in - NDSS 2014

https://ndss2014.ece.cmu.edu/crp/ndss2014/index.php

NDSS 2014 Sign in

Welcome to the NDSS 2014 submissions site. Sign in to submit or review papers. For general information about NDSS 2014, see the [conference site](#).

Email

Password

Sign me in
 I forgot my password, email it to me
 I'm a new user and want to create an account using this email address

Sign in

Submissions: The deadline for registering new papers has passed.

Conference information

[Deadlines](#)
[Program committee](#)
[Conference site](#)

55 papers were accepted out of 293 submitted.

HotCRP Conference Management Software

Simulation of Built-in PHP Features for Precise Static Code Analysis

1. Introduction
2. Implementation
3. Evaluation
4. Conclusion

Target: Taint-style Vulnerabilities

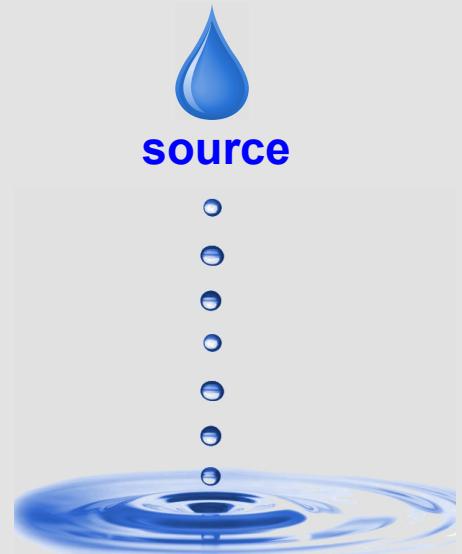
- SQL injection

```
<?php
    $id = $_GET['id'];
    $sql = "SELECT data FROM users WHERE id = '$id' ";
    mysql_query($sql);
?>
```



- Cross-Site Scripting

```
<?php
    $name = $_GET['name'];
    $html = "<h1>Hello $name</h1>";
    print($html);
?>
```

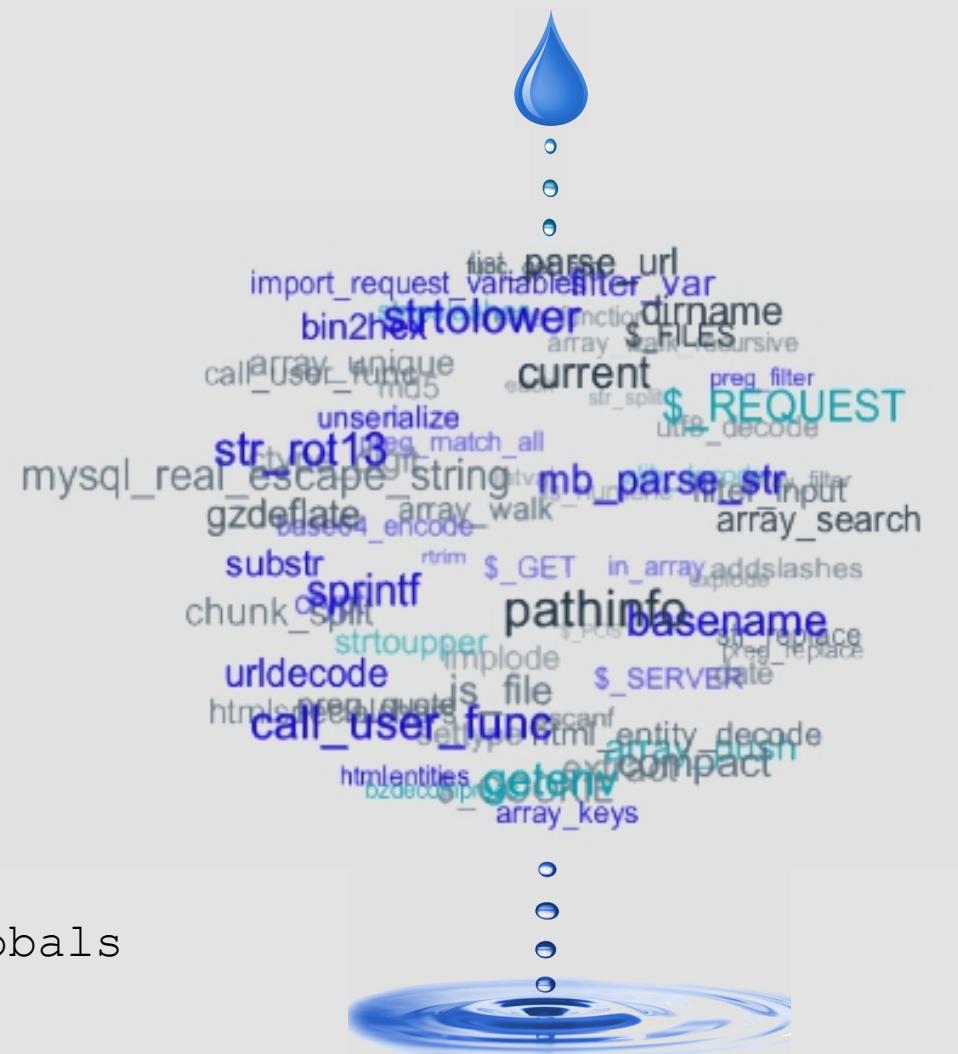


sensitive sink

Simulation of Built-in PHP Features for Precise Static Code Analysis

1. Introduction
 2. Implementation
 3. Evaluation
 4. Conclusion

PHP Built-in Features



- 228+ Extensions
 - 5700+ built-in functions
 - Sinks, sanitization, data flow
 - 10+ *superglobal* variables
 - \$GLOBALS, \$_FILES, \$_SERVER ...
 - Settings
 - magic quotes gpc, register globals

Simulation of Built-in PHP Features for Precise Static Code Analysis

1. Introduction
2. Implementation
3. Evaluation
4. Conclusion

Our Approach

- Static Code Analysis for PHP applications
- Precise simulation of built-in features is the key
 - to detect taint-style vulnerabilities
 - *to accept your paper on your own*



Simulation of Built-in PHP Features for Precise Static Code Analysis

1. Introduction
2. Implementation
3. Evaluation
4. Conclusion



2. Implementation



Simulation of Built-in PHP Features for Precise Static Code Analysis

1. Introduction
2. Implementation
3. Evaluation
4. Conclusion

Precise Simulation

<http://rub.de/index.php/payload>

```
<?php
    $uri = trim($_SERVER['PHP_SELF']);
    $uri = urldecode($uri);
    $url = 'http://rub.de/' . htmlentities($uri);
    $html = "<a href='$url' >back</a>";
    print($html);
?>
```

Simulation of Built-in PHP Features for Precise Static Code Analysis

1. Introduction
2. Implementation
3. Evaluation
4. Conclusion

Precise Simulation

1. taintable sources

- `$_FILES[]['name']`
- `$_FILES[]['tmp_name']`
- `$_SERVER['PHP_SELF']`
- `$_SERVER['REMOTE_ADDR']`

```
<?php
    $uri = trim($_SERVER['PHP_SELF']);
    $uri = urldecode($uri);
    $url = 'http://rub.de/' . htmlentities($uri);
    $html = "<a href='$url' >back</a>";
    print($html);
?>
```

Simulation of Built-in PHP Features for Precise Static Code Analysis

1. Introduction
2. Implementation
3. Evaluation
4. Conclusion

Precise Simulation

1. tainted sources

- `$_FILES[]['name']`
- `$_FILES[]['tmp_name']`
- `$_SERVER['PHP_SELF']`
- `$_SERVER['REMOTE_ADDR']`

```
<?php
    $uri = trim($_SERVER['PHP_SELF']);
    $uri = urldecode($uri);
    $url = 'http://rub.de/' . htmlentities($uri);
    $html = "<a href='$url'>back</a>";
    print($html);
?>
```

S1	<code><u>\$_SERVER</u></code>
	<code>PHP_SELF</code>
	Path ../ Traversal

`http://rub.de/index.php/../../../../..`

Simulation of Built-in PHP Features for Precise Static Code Analysis

1. Introduction
2. Implementation
3. Evaluation
4. Conclusion

Precise Simulation

2. data flow

- Format string
- Regular expressions

1. tainted sources

- `$_FILES[]['name']`
- ~~• `$_FILES[]['tmp_name']`~~
- `$_SERVER['PHP_SELF']`
- ~~• `$_SERVER['REMOTE_ADDR']`~~

```
<?php
    $uri = trim($_SERVER['PHP_SELF']);
    $uri = urldecode($uri);
    $url = 'http://rub.de/' . htmlentities($uri);
    $html = "<a href='$url' >back</a>";
    print($html);
?>
```

S1	<code>\$_SERVER</code>
	<code>PHP_SELF</code>
	Path ../ Traversal

Simulation of Built-in PHP Features for Precise Static Code Analysis

1. Introduction
2. Implementation
3. Evaluation
4. Conclusion

Precise Simulation

3. encoding

- Encoding stack
- Interaction with sanitization

2. data flow

- Format string
- Regular expressions

1. taintable sources

- `$_FILES[]['name']`
- `$_FILES[]['tmp_name']`
- `$_SERVER['PHP_SELF']`
- `$_SERVER['REMOTE_ADDR']`

```
<?php
$uri = trim($_SERVER['PHP_SELF']);
$uri = urldecode($uri);
$url = 'http://rub.de/' . htmlentities($uri);
$html = "<a href='$url' >back</a>";
print($html);
?>
```

S1	<code>\$_SERVER</code>
	<code>PHP_SELF</code>
	Path ../ Traversal

Simulation of Built-in PHP Features for Precise Static Code Analysis

1. Introduction
2. Implementation
3. Evaluation
4. Conclusion

Precise Simulation

3. encoding

- Encoding stack
- Interaction with sanitization

2. data flow

- Format string
- Regular expressions

1. taintable sources

- `$_FILES[]['name']`
- `$_FILES[]['tmp_name']`
- `$_SERVER['PHP_SELF']`
- `$_SERVER['REMOTE_ADDR']`

```
<?php
$uri = trim($_SERVER['PHP_SELF']);
$uri = urldecode($uri);
$url = 'http://rub.de/' . htmlentities($uri);
$html = "<a href='$url' >back</a>";
print($html);
?>
```

S1

`$_SERVER`

`PHP_SELF`

Simulation of Built-in PHP Features for Precise Static Code Analysis

1. Introduction
2. Implementation
3. Evaluation
4. Conclusion

Precise Simulation

3. encoding

- Encoding stack
- Interaction with sanitization

2. data flow

- Format string
- Regular expressions

4. sanitization

- Sanitization tags
- Context-sensitive

1. taintable sources

- `$_FILES[]['name']`
- `$_FILES[]['tmp_name']`
- `$_SERVER['PHP_SELF']`
- `$_SERVER['REMOTE_ADDR']`

```
<?php
$uri = trim($_SERVER['PHP_SELF']);
$uri = urldecode($uri);
$url = 'http://rub.de/' . htmlentities($uri);
$html = "<a href='$url' >back</a>";
print($html);
?>
```

S1

`$_SERVER`

`PHP_SELF`

Simulation of Built-in PHP Features for Precise Static Code Analysis

1. Introduction
2. Implementation
3. Evaluation
4. Conclusion

Precise Simulation

3. encoding

- Encoding stack
- Interaction with sanitization

2. data flow

- Format string
- Regular expressions

4. sanitization

- Sanitization tags
- Context-sensitive

1. taintable sources

- `$_FILES[]['name']`
- `$_FILES[]['tmp_name']`
- `$_SERVER['PHP_SELF']`
- `$_SERVER['REMOTE_ADDR']`

```
<?php
$uri = trim($_SERVER['PHP_SELF']);
$uri = urldecode($uri);
$url = 'http://rub.de/' . htmlentities($uri);
$html = "<a href='$url' >back</a>";
print($html);
?>
```

S1	<code>\$_SERVER</code>
	<code>PHP_SELF</code>
XSS <> Element	XSS DQ" Attribute

Simulation of Built-in PHP Features for Precise Static Code Analysis

1. Introduction
2. Implementation
3. Evaluation
4. Conclusion

Precise Simulation

3. encoding

- Encoding stack
- Interaction with sanitization

2. data flow

- Format string
- Regular expressions

4. sanitization

- Sanitization tags
- Context-sensitive

1. taintable sources

- `$_FILES[]['name']`
- `$_FILES[]['tmp_name']`
- `$_SERVER['PHP_SELF']`
- `$_SERVER['REMOTE_ADDR']`

```
<?php
$uri = trim($_SERVER['PHP_SELF']);
$uri = urldecode($uri);
$url = 'http://rub.de/' . htmlentities($uri);
$html = "<a href='$url' >back</a>";
print($html);
?>
```

S1	<code>\$_SERVER</code>
	<code>PHP_SELF</code>
XSS <-> Element	XSS DQ" Attribute

5. sinks

- Parameter
- Vulnerability type

Simulation of Built-in PHP Features for Precise Static Code Analysis

1. Introduction
2. Implementation
3. Evaluation
4. Conclusion

Precise Simulation

3. encoding

- Encoding stack
- Interaction with sanitization

2. data flow

- Format string
- Regular expressions

4. sanitization

- Sanitization tags
- Context-sensitive

1. taintable sources

- `$_FILES[]['name']`
- `$_FILES[]['tmp_name']`
- `$_SERVER['PHP_SELF']`
- `$_SERVER['REMOTE_ADDR']`

```
<?php
$uri = trim($_SERVER['PHP_SELF']);
$uri = urldecode($uri);
$url = 'http://rub.de/' . htmlentities($uri);
$html = "<a href='$url' >back</a>";
print($html);
```

S1	<code>\$_SERVER</code>
	<code>PHP_SELF</code>
XSS <-> Element	XSS DQ" Attribute

6. markup context

```
<a href='http://rub.de/S1' >back</a>
→ XSS Single-Quoted ' Attribute
```

5. sinks

- Parameter
- Vulnerability type

Simulation of Built-in PHP Features for Precise Static Code Analysis

1. Introduction
2. Implementation
3. Evaluation
4. Conclusion

Precise Simulation

3. encoding

- Encoding stack
- Interaction with sanitization

2. data flow

- Format string
- Regular expressions

4. sanitization

- Sanitization tags
- Context-sensitive

1. taintable sources

- `$_FILES[]['name']`
- `$_FILES[]['tmp_name']`
- `$_SERVER['PHP_SELF']`
- `$_SERVER['REMOTE_ADDR']`

```
<?php
$uri = trim($_SERVER['PHP_SELF']);
$uri = urldecode($uri);
$url = 'http://rub.de/' . htmlentities($uri);
$html = "<a href='$url' >back</a>";
print($html);
```

S1	<code>\$_SERVER</code>
	<code>PHP_SELF</code>
XSS	<>
Element	XSS DQ"
	Attribute

6. markup context

```
<a href='http://rub.de/S1' >back</a>
→ ' onclick='alert(document.cookie)
```

5. sinks

- Parameter
- Vulnerability type

Simulation of Built-in PHP Features for Precise Static Code Analysis

1. Introduction
2. Implementation
3. Evaluation
4. Conclusion

Precise Simulation

3. encoding

- Encoding stack
- Interaction with sanitization

2. data flow

- Format string
- Regular expressions

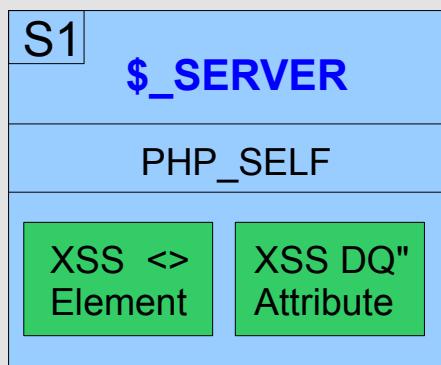
4. sanitization

- Sanitization tags
- Context-sensitive

1. taintable sources

- `$_FILES[]['name']`
- `$_FILES[]['tmp_name']`
- `$_SERVER['PHP_SELF']`
- `$_SERVER['REMOTE_ADDR']`

- 952 built-in functions
- 20 vulnerability types
- 45 markup contexts



6. markup context

```
<a href='http://rub.de/S1'>back</a>  
→ ' onclick='alert(document.cookie)
```

5. sinks

- Parameter
- Vulnerability type

Simulation of Built-in PHP Features for Precise Static Code Analysis

1. Introduction
2. Implementation
3. Evaluation
4. Conclusion

3. Evaluation

```
sitelanguage'] = $GLOBE
GLOBALS['elan'] = $eln;
tracking'] == "session"
language_subdomain'] ==
': !if($eln == $slng
    $slng = new l
    lan'] = $pref
tracking'] == "session"
language_subdomain'] ==
: $pref['sitelanguag
```

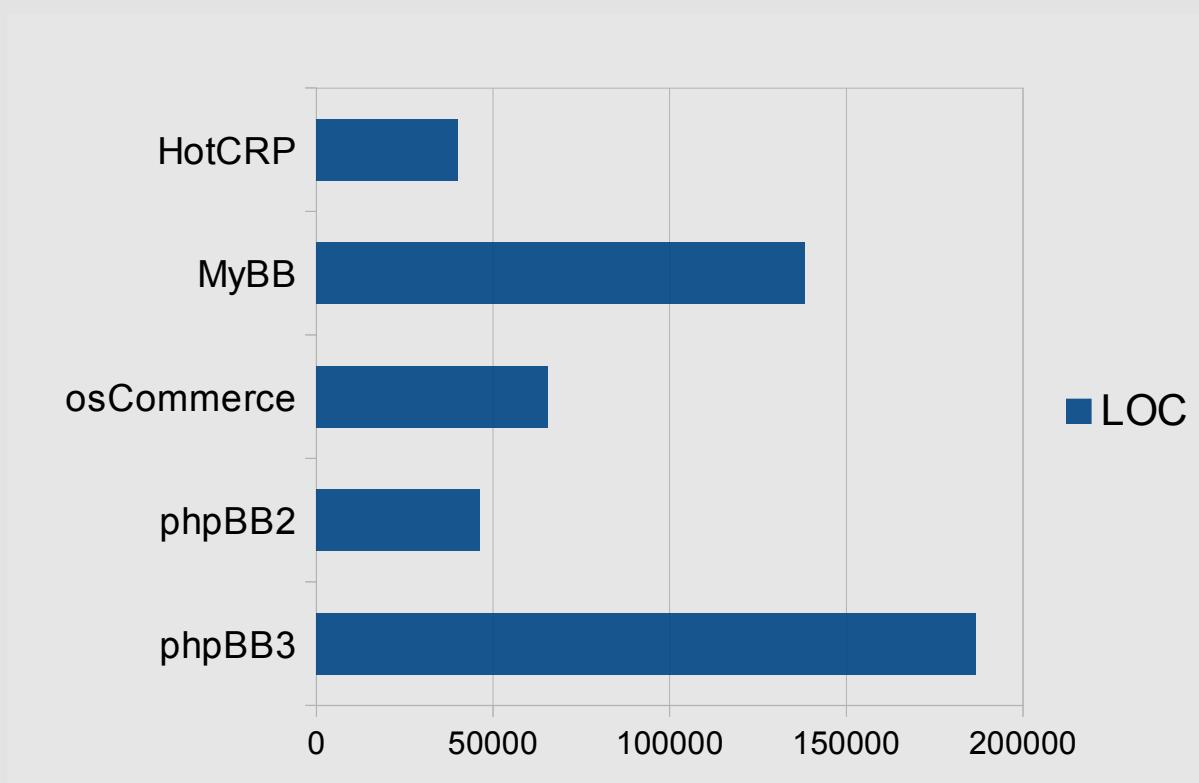


Simulation of Built-in PHP Features for Precise Static Code Analysis

1. Introduction
2. Implementation
3. Evaluation
4. Conclusion

Software

- HotCRP 2.60
- MyBB 1.6.10
- OsCommerce 2.3.3
- phpBB2 2.0.23
- phpBB3 3.0.11

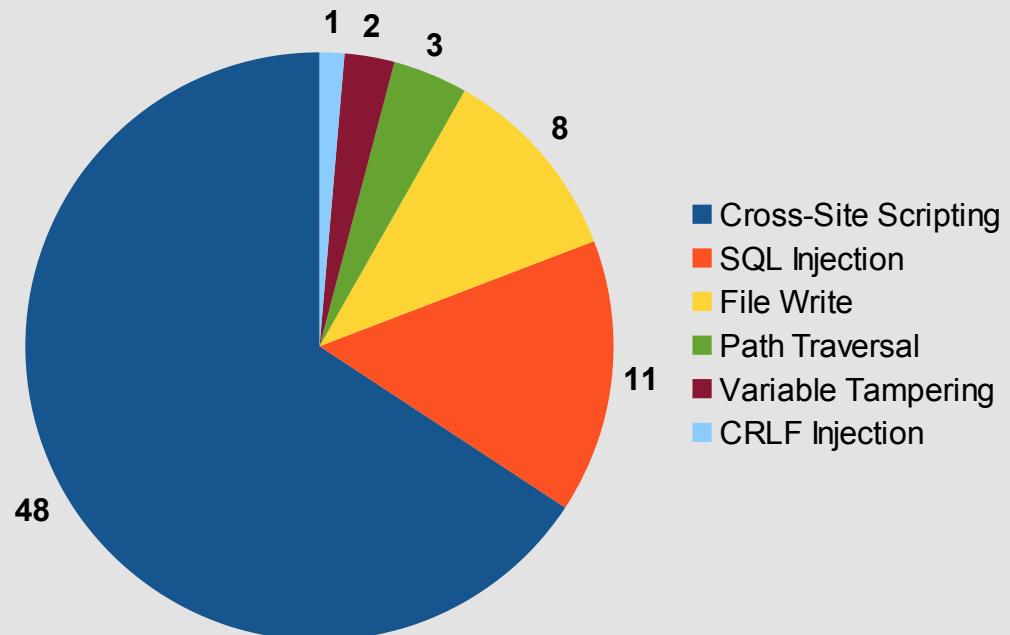


Simulation of Built-in PHP Features for Precise Static Code Analysis

1. Introduction
2. Implementation
3. Evaluation
4. Conclusion

Vulnerability Detection

- 73 True Positives (72%)
- 29 False Positives (28%)
 - 19 FP in OsCommerce
 - Root cause: Path-sensitivity
- 10 False Negatives (24%)
 - 42 CVE entries
 - 8 FN in MyBB
 - Root cause: OOP

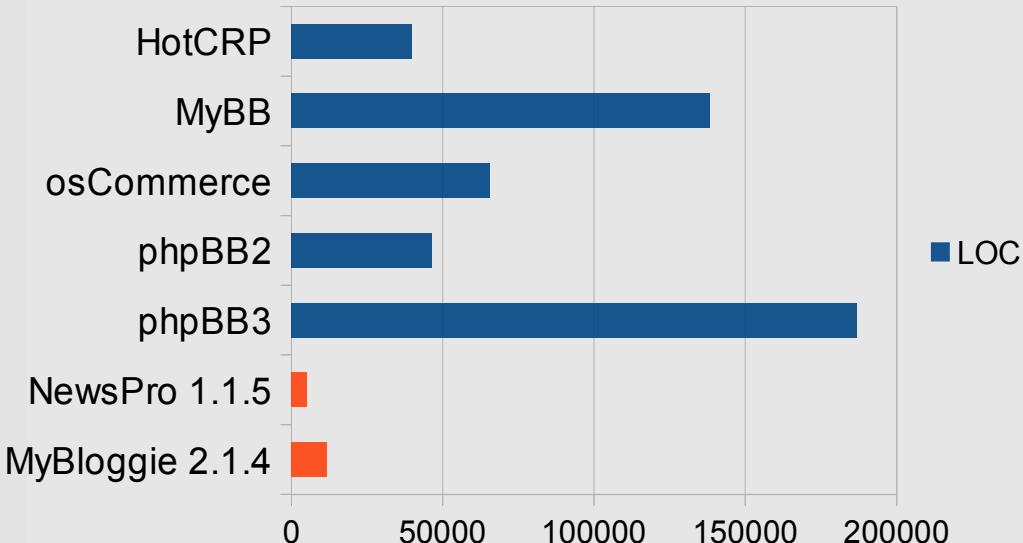


Simulation of Built-in PHP Features for Precise Static Code Analysis

1. Introduction
2. Implementation
3. Evaluation
4. Conclusion

Software in Related Work

- Criteria
 - Available
 - Follow-up version exists
 - Patch-only
- Our results
 - 31 new vulnerabilities detected
 - 0 false positives
 - Precise simulation pays off



Simulation of Built-in PHP Features for Precise Static Code Analysis

1. Introduction
2. Implementation
3. Evaluation
4. Conclusion

Vulnerability Example

- Blind SQL Injection in HotCRP 2.60
- Fixed in version 2.61
- HotCRP stores credentials in plaintext

The screenshot shows a web application interface for 'hotCRP test'. At the top, there's a navigation bar with 'Paper #2' and user information 'johannes.dahse@rub.de' along with links for 'Profile', 'Help', and 'Sign out'. Below the navigation, there's a search bar with 'Your submissions #3▶ (All)' and a 'Search' button. A prominent red error message box contains the following text:
Database error while updating paper options (insert into PaperOption (paperId, optionId, value, data) values (2, 1, (SELECT extractvalue(1,concat(0x2e,(SELECT concat(email,0x3a,password) FROM contactInfo limit 1)))), null)): XPATH syntax error: 'administrator@rub.de:secretpass1'
The word 'extractvalue' is highlighted in red, and the email address 'administrator@rub.de:secretpass1' is underlined in red. At the bottom of the page, there are 'Main' and 'Edit' buttons.

Simulation of Built-in PHP Features for Precise Static Code Analysis

1. Introduction
2. Implementation
3. Evaluation
4. Conclusion

Test 2014

Paper #1

test@test.de

[Profile](#) | [Help](#) | [Sign out](#)

 Main

 Edit

 Review

 Assign

All papers

(All)

Search

#1 Simulation of Built-in PHP Features for Precise Static Code Analysis

Submitted



233kB

① 16 Feb 2014 7:43pm UTC

3f230a41569a5a9fe16404a090e0fd45a02d41c0

You are an **author** of this paper.

+ ABSTRACT

The World Wide Web grew rapidly during the last decades and is used by millions of people every day for online shopping, banking, networking, and other activities. Many of [\[more\]](#)

AUTHORS

+ *Hidden for blind review*



You have used administrator privileges to view and edit reviews for this paper.
[\(Unprivileged view\)](#)

Review #1A test@test.de 

OveMer RevExp

5 4



[Edit your review](#)



[Assign reviews](#)



[Add comment](#)

Simulation of Built-in PHP Features for Precise Static Code Analysis

1. Introduction
2. Implementation
3. Evaluation
4. Conclusion

```
sitelanguage'] = $GLOBAL
$OBALS['elan'] = $eln;
tracking'] == "session")
language_subdomain'] ==
: elseif($eln == $slng
 392: $slng = new la
$OBALS['elan'] = $pref[
tracking'] == "session")
language_subdomain'] ==
: $pref['sitelanguage
```

4. Conclusion



Conclusion

- New approach to PHP static code analysis
 - 20 vulnerability types, 45 markup contexts
 - 900+ built-in features simulated
- 73 new vulnerabilities, 28% false positives
 - Current vulnerabilities base on complex PHP features
 - Modeling these features precisely is crucial, missed by previous work
- Future work
 - Path-sensitivity
 - OOP

Simulation of Built-in PHP Features for Precise Static Code Analysis

1. Introduction
2. Implementation
3. Evaluation
4. Conclusion

Questions ?

johannes.dahse@rub.de

Simulation of Built-in PHP Features for Precise Static Code Analysis

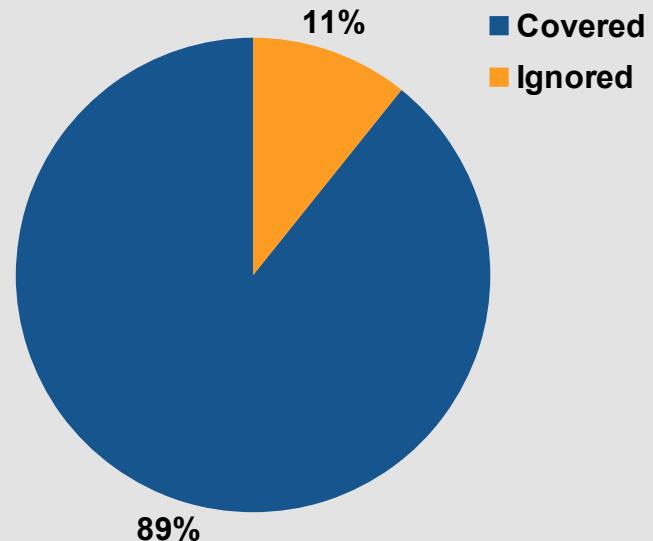
**Thank you!
Enjoy the conference.**

Simulation of Built-in PHP Features for Precise Static Code Analysis

Backup Slides

Built-in Function Coverage

- Every 13th line of code calls a built-in function
 - Static point of view
- 970 **unique** calls
 - 70% covered
- 37 651 **total** calls
 - 89% covered
- Remaining calls are less relevant
 - Do not influence our analysis results



Simulation of Built-in PHP Features for Precise Static Code Analysis

1. Introduction
2. Implementation
3. Evaluation
4. Conclusion

Target: Taint-style Vulnerabilities

- SQL injection

```
<?php
    $id = mysql_real_escape_string($_GET['id']);
    $sql = "SELECT data FROM users WHERE id = $id ";
    mysql_query($sql);
?>
```



- Cross-Site Scripting

```
<?php
    $name = htmlentities($_GET['name']);
    $html = "<h1>Hello $name</h1>";
    print($html);
?>
```

Simulation of Built-in PHP Features for Precise Static Code Analysis

Path-sensitive sanitization

```
1 function tep_output_string($string, $protected = false) {  
2     if ($protected == true) {  
3         return htmlspecialchars($string);  
4     } else {  
5         return $string;  
6     }  
7 }
```

Simulation of Built-in PHP Features for Precise Static Code Analysis

Supported vulnerability types

- 1) Code Execution
- 2) Command Execution
- 3) Connect Injection
- 4) Cross-Site Scripting
- 5) Denial of Service
- 6) Env. Manipulation
- 7) File Inclusion
- 8) File Upload
- 9) File Write
- 10) HTTP Resp. Splitting
- 11) LDAP Injection
- 12) Open Redirect
- 13) Path Traversal
- 14) Reflection Injection
- 15) Session Fixation
- 16) SQL Injection
- 17) Unserialize
- 18) Variable Tampering
- 19) XML/XXE Injection
- 20) XPath Injection

Simulation of Built-in PHP Features for Precise Static Code Analysis

TABLE I: Evaluation results for popular real-world applications.

Software	Files	LOC	TA	TBC	TBI	UBC	UBI	MP	ST	TP	FP	FN	CVE
HotCRP	72	39 938	19 420	5 171	289	170	51	293	55	7	4	0	0
MyBB	327	138 357	55 917	8 152	1 287	225	115	1 117	188	2	0	8	10
osCommerce	545	65 556	7 453	9 059	860	184	85	476	60	48	19	1	29
phpBB2	176	46 287	10 623	3 666	340	144	56	289	29	13	6	1	2
phpBB3	270	186 814	43 616	7 554	1 273	269	192	1 143	252	3	0	0	1
Total	1 390	476 952	137 029	33 602	4 049	676	294	3 318	584	73	29	10	42
Average	278	95 390	27 406	89%	11%	70%	30%	664	117	72%	28%	24%	8

TABLE II: Compared evaluation results for previously studied real-world applications.

Software	Files	LOC	TB	UB	RIPS				Jovanovic et al.				Xie & Aiken		
					XSS	SQLi	XSS	SQLi	TP	FP	TP	FP	TP	FP	
NewsPro	1.1.4	23	5 047	827	56	5	0	18	0	4	14	14	34	8	0
NewsPro	1.1.5	23	5 077	841	57	4	0	6	0	-	-	-	-	-	-
myBloggie	2.1.3b	91	11 487	1 218	122	15	0	26	3	13	3	31	11	16	0
myBloggie	2.1.4	92	11 772	1 235	124	13	0	8	0	-	-	-	-	-	-
Total	229	33 383	4 121	134	37	0	58	3	17	17	45	45	24	0	
Average	57	8 346	1 030	90	100%	0%	95%	5%	50%	50%	50%	50%	100%	0%	

Simulation of Built-in PHP Features for Precise Static Code Analysis

SQL Injection in phpBB2

```
1 $style_name = urldecode($_GET['style']);
2 $install_to = urldecode($_GET['install_to']);
3 $template_name = $$install_to;
4 for($i = 0; $i < count($template_name); $i++) {
5     if($template_name[$i]['style_name'] == $style_name) {
6         while(list($key, $val) = each($template_name[$i])) {
7             $db_fields[] = $key;
8             $db_values[] = addslashes($val);
9         }
10    }
11 }
12 $sql = "INSERT INTO ".THEMES_TABLE." (";
13 $sql .= implode(',',$db_fields);
14 $sql .= ") VALUES (";
15 $sql .= "'" . implode("'", "'",$db_values) . "'";
16 $sql .= ")";
17 mysql_query($sql);
```

admin_styles.php?style=rips&install_to=_GET&0[style_name]=rips&0[template_name]VALUES('sql','sql')-- -]=1