Tainted-Object-Propagation-Analyse

für PHP5

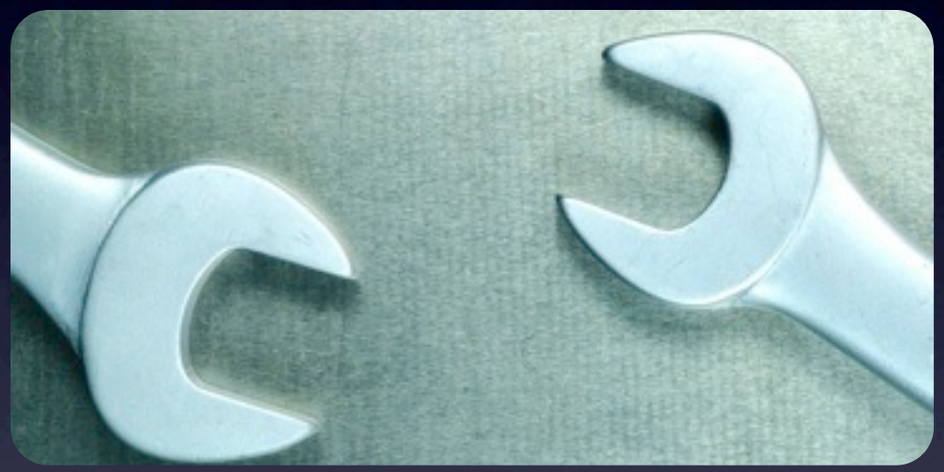
basierend auf

Pixy

Oliver Klee
pixy@oliverklee.de

Einstieg

Tainting



Alias-Analyse

Objekte in PHP

Objektreferenzen

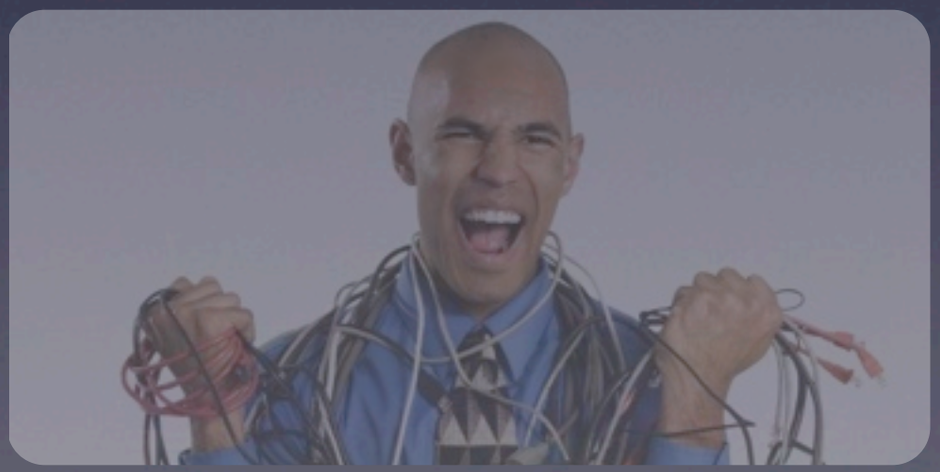
Ausblick

Einstieg

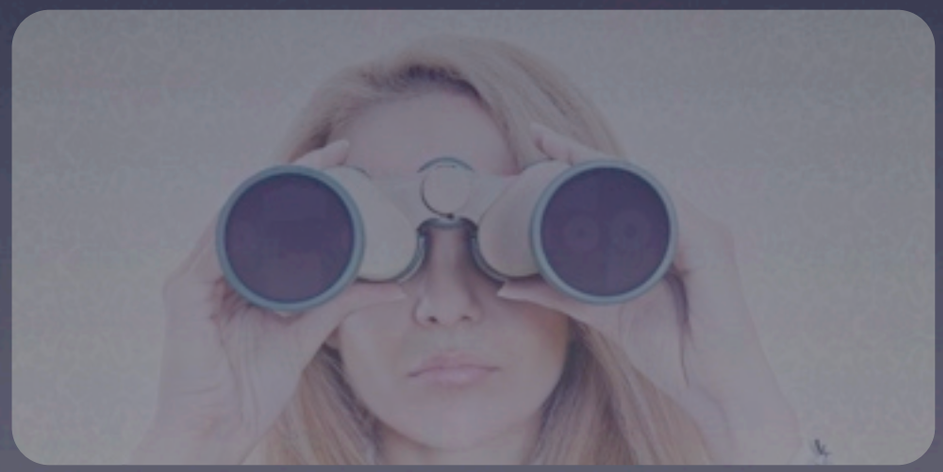
Tainting

Alias-Analyse

Objekte in PHP

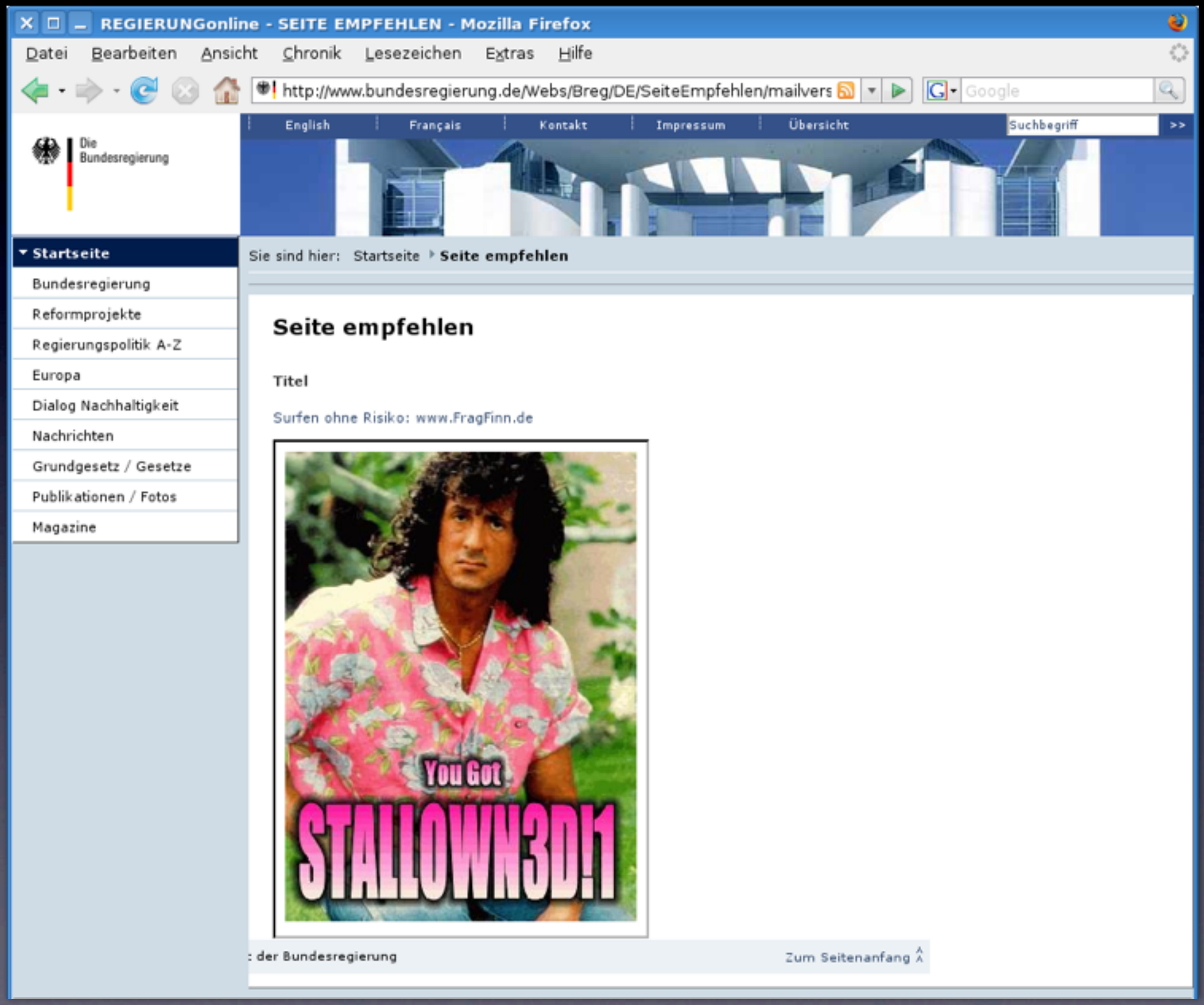


Objektreferenzen



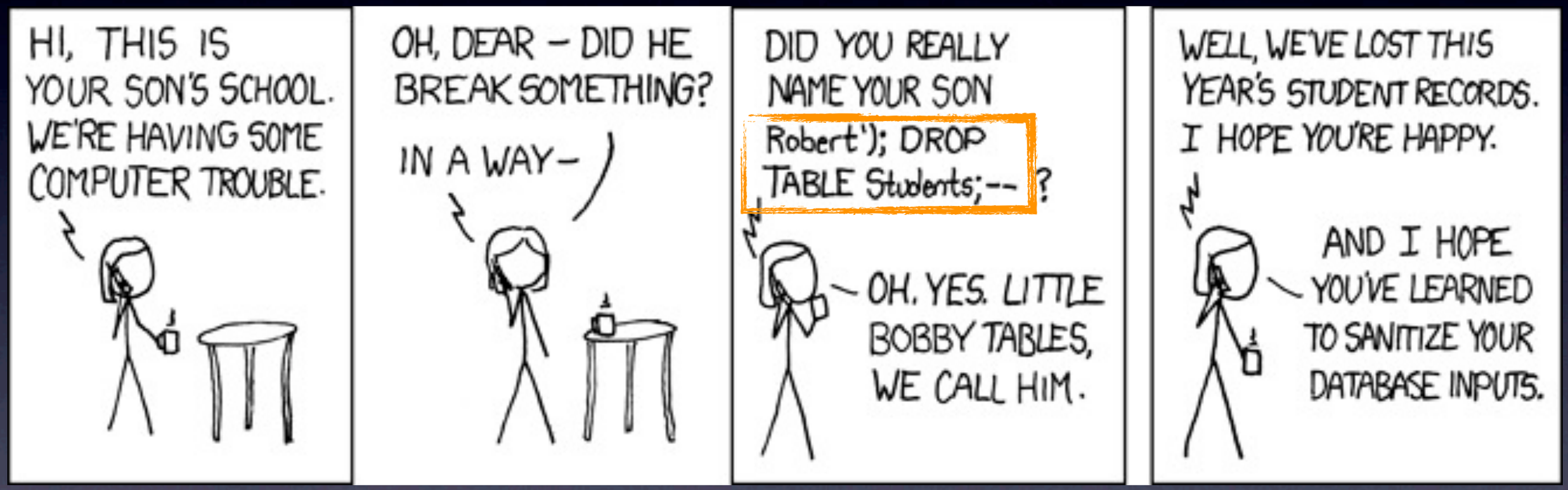
Ausblick

Cross-site-Scripting (XSS)



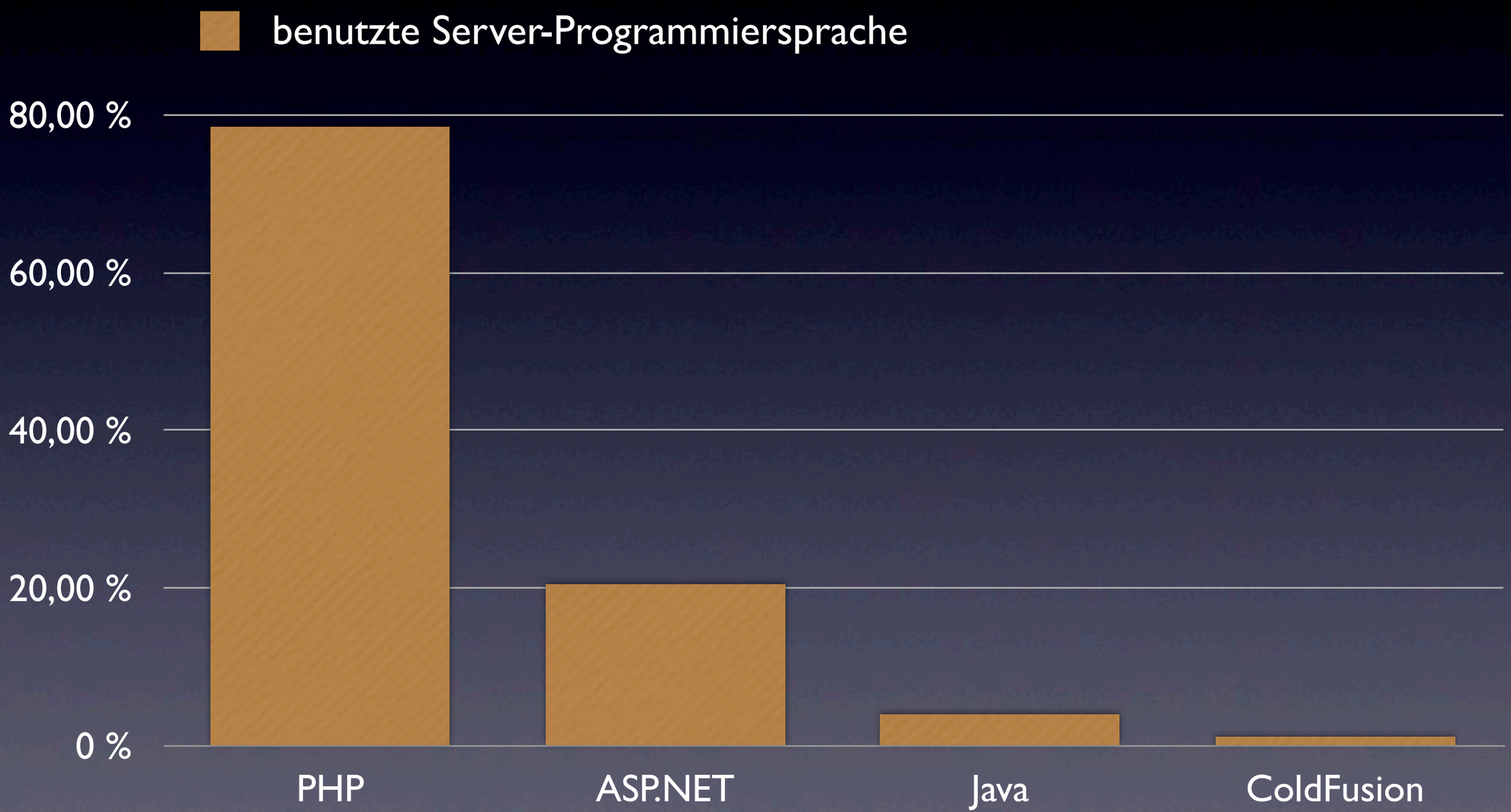
http://www.heise.de/security/meldung/XSS-Schwachstelle-auf-Bundesregierung-de-170817.html

SQL-Injection (SQLi)



http://xkcd.com/327/

PHP ist weit verbreitet



http://w3techs.com/technologies/overview/programming_language/all

PHP ist . . .

eine Skriptsprache
objektorientiert
dynamisch

```
class Foo {}
$foo = new Foo();
$foo = 42;
$variableName = 'foo';
echo $$variableName;
```

Pixy ist der Scanner der Wahl

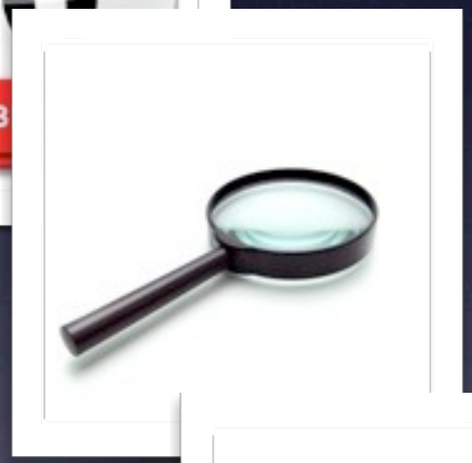
	Typ	Open Source	läuft	Recall	Precision
SWAAT	String-Matching	?	✓	✗	✗
Code Secure Verifier	Datenfluss-Analyse	✗	keine Testversion	?	?
PHP-SAT	Datenfluss-Analyse	✓	✗	?	?
Pixy	Datenfluss-Analyse	✓	✓	✓	✓
YASCA ohne Plugins	Pattern-Matching	✓	✓	✗	✗

Pixy

Autor: Nenad Jovanonic (TU Wien, Dissertation)
Betreuer: Christopher Krügel
(Associate-Professor an der University of California, Santa Barbara)

6 Publikationen auf
Konferenzen und Workshops

2006-2007 entstanden



scannt
Tainted-Object-Propagation
ausgefeilte Alias-Analyse
scannt nur PHP 4.x

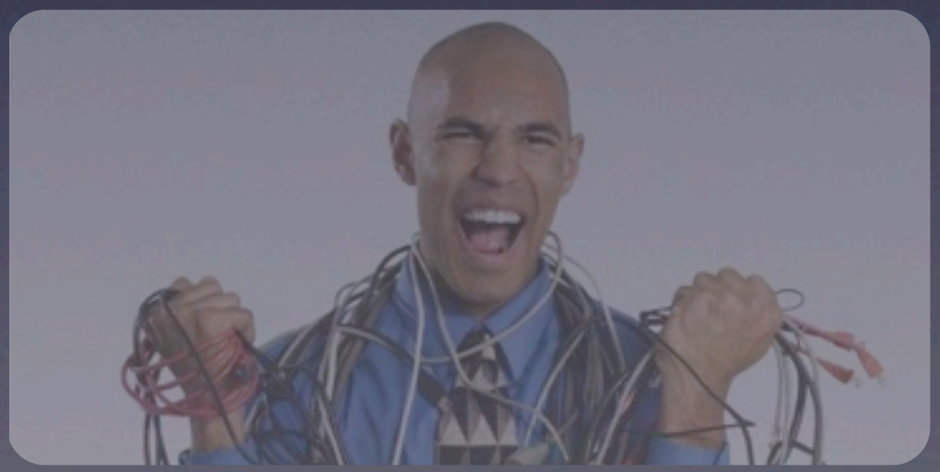
in Java 5/6
sehr wenig JavaDoc

Einstieg

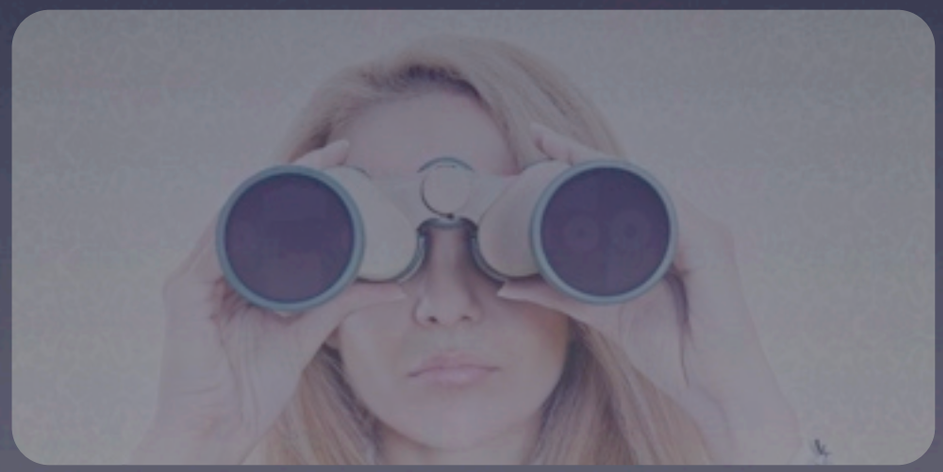
Tainting

Alias-Analyse

Objekte in PHP



Objektreferenzen



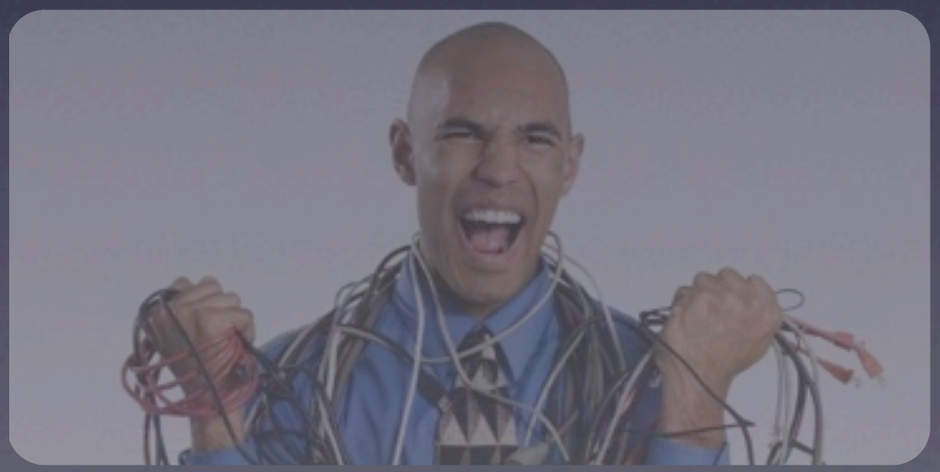
Ausblick

Einstieg

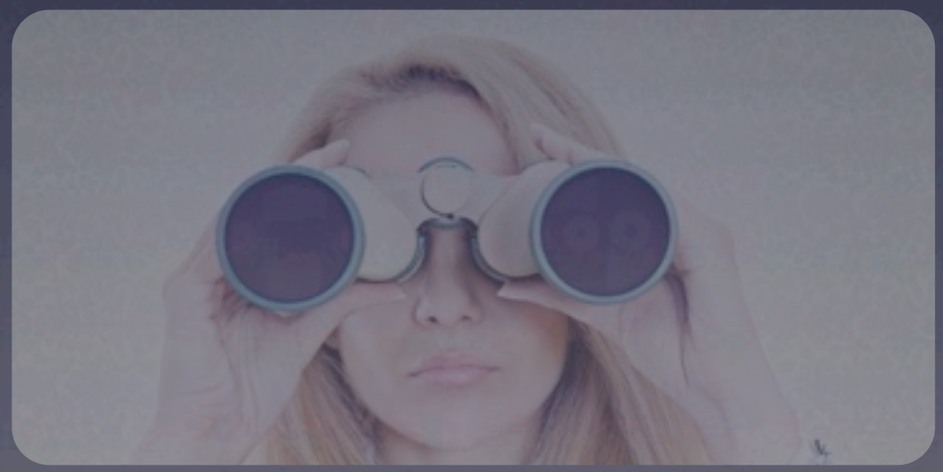
Tainting

Alias-Analyse

Objekte in PHP



Objektreferenzen



Ausblick

Tainted-Object-Propagation-Analyse

nur für Strings

Sources

Datenfluss-Analyse

Sinks

Literale sind immer untainted

Sources sind immer tainted

```
$x = $_GET['x'];
echo $x;
```

Sanitizing-Funktionen helfen gegen Tainting

Source

Sanitation

```
$name = $_GET['name'];

$safeName = htmlspecialchars($name);

echo '<p>Hello ' . $safeName . '!</p>';
```

Sink

Pixy ist konservativ beim Tainting

```
$x = $_GET['x'];
if (...) {
   $x = intval(x);
}
echo $x;
```

x:tainted

x:untainted

x:tainted

Sanitizing ist angriffs-spezifisch

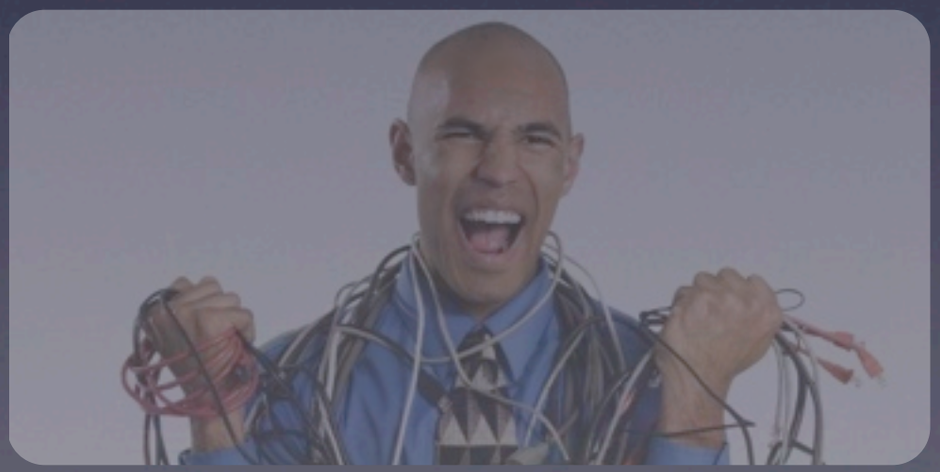
```
$id = $_GET['id'];
$id = htmlspecialchars($id);
mysql_query(
   'SELECT * from users ' .
   'WHERE id=' . $id;
);
```

Einstieg

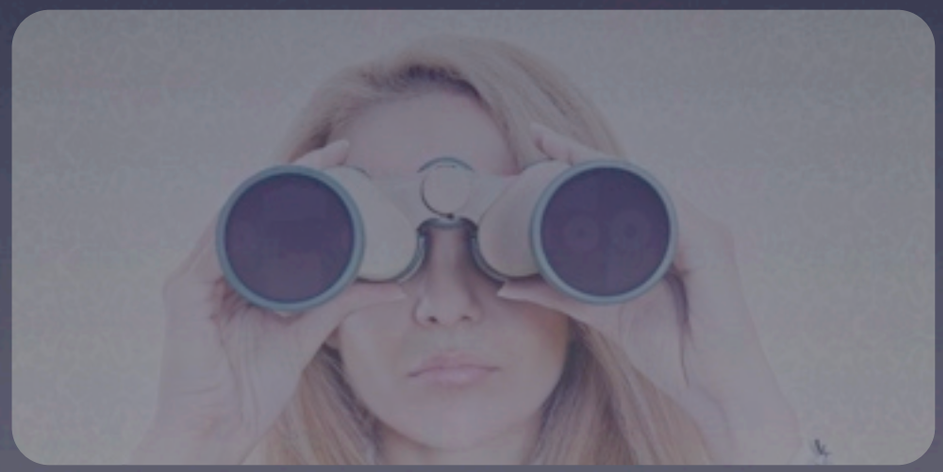
Tainting

Alias-Analyse

Objekte in PHP



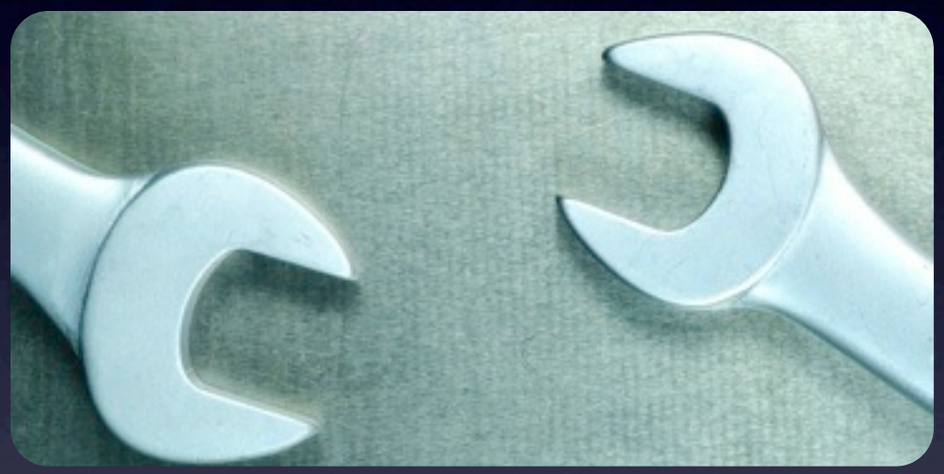
Objektreferenzen



Ausblick

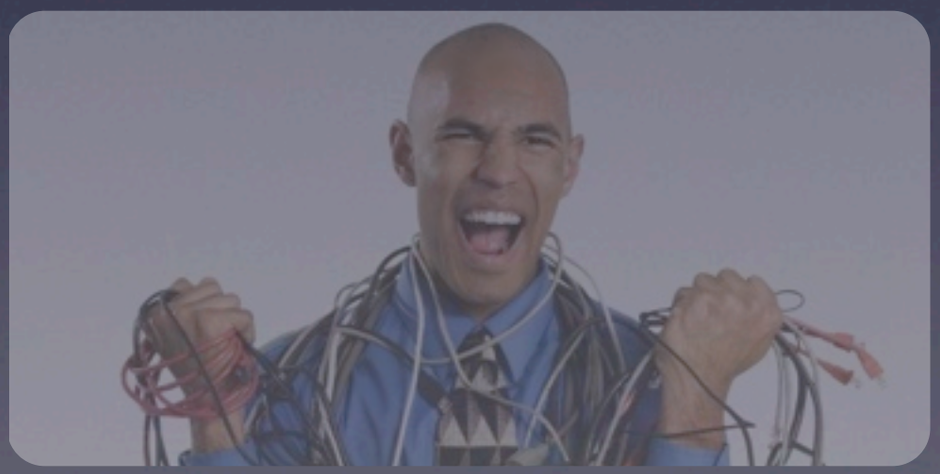
Einstieg

Tainting

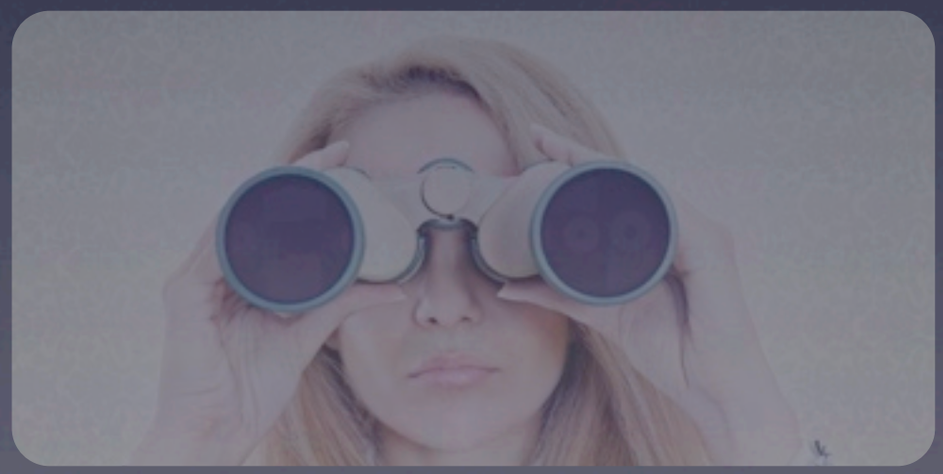


Alias-Analyse

Objekte in PHP



Objektreferenzen



Ausblick

Referenzen in PHP sind keine Pointer

```
$a = 42;
$b = $a;
$bRef = &$b;
```

Es gibt Must- und May-Aliase

```
$a1 = '42';
$a2 = &$a1;
$b1 = 'Hello world';
if (...) {
   $b2 = &$b1;
} else {
   $b2 = 'Coffee.';
}
```

Must = {}, May = {}

Must = {(a1, a2)}, May = {}

Must = {(a1, a2), (b1, b2)},
May = {}

Must = {(a1, a2)}, May = {}

Must = {(a1, a2)},
May = {(b1, b2)}

Must-Aliase und Tainting

```
$a1 = $_GET['x'];
```

```
$a1 = intval($a1);
```

MustAliases = {a1, a2}

a1:tainted, a2:tainted

a1:untainted, a2:untainted

May-Aliase und Tainting

```
$a1 = $_GET['x'];
```

```
$a1 = intval($a1);
```

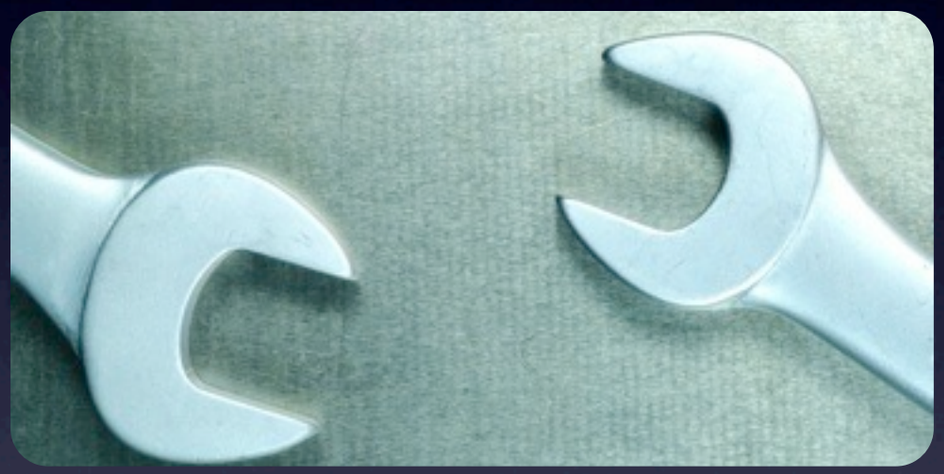
MayAliases = {a1, a2}

a1:tainted, a2:tainted

a1:untainted, a2:tainted

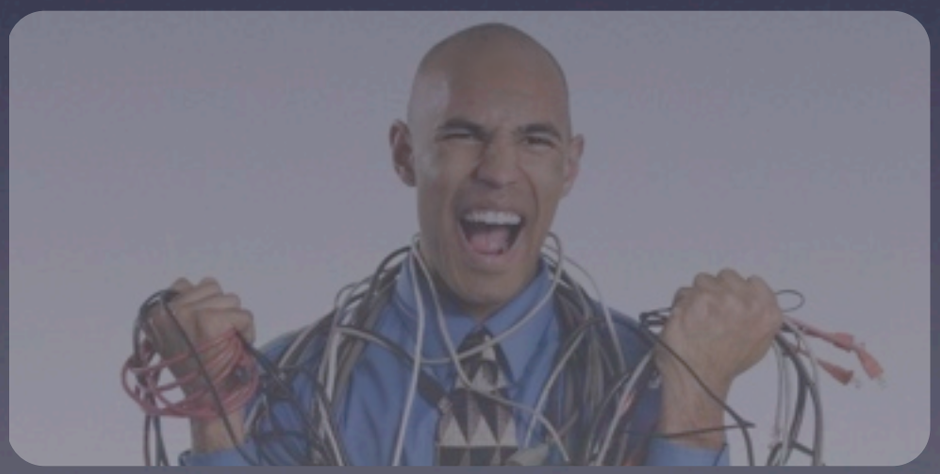
Einstieg

Tainting

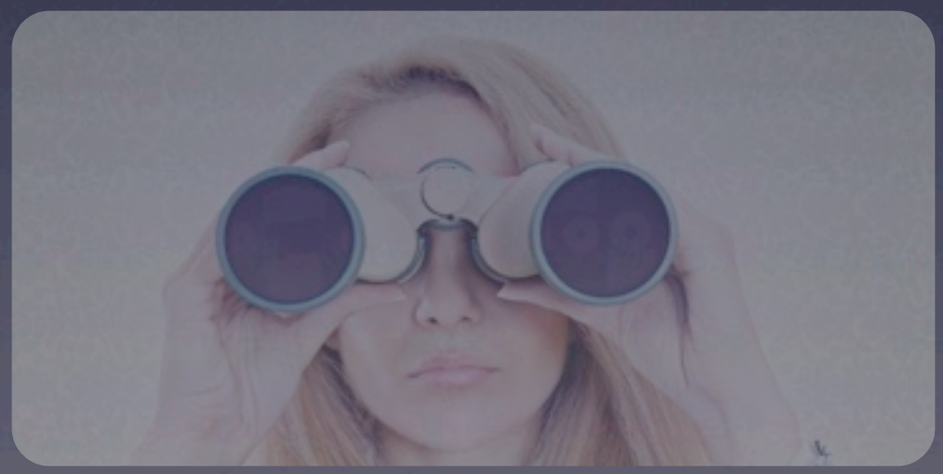


Alias-Analyse

Objekte in PHP



Objektreferenzen



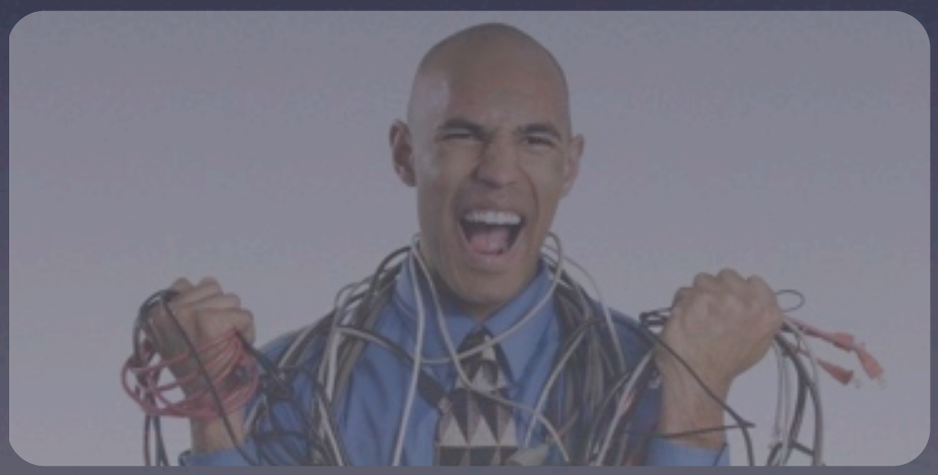
Ausblick

Einstieg

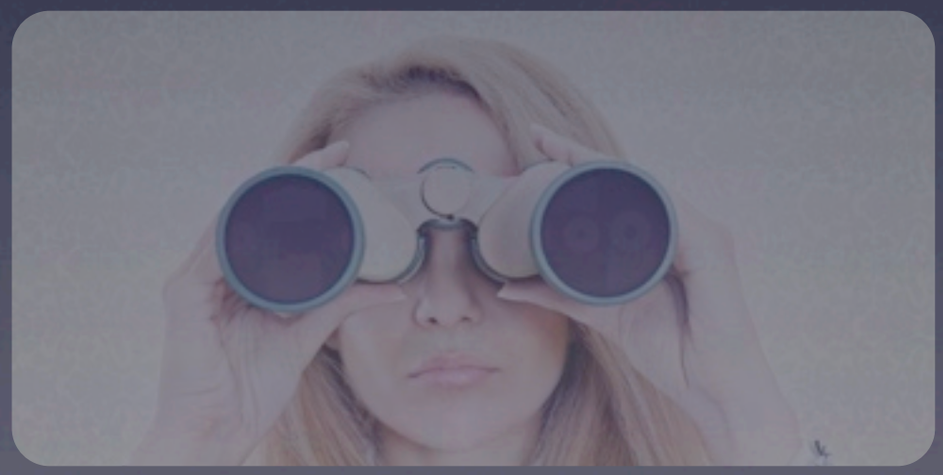
Tainting

Alias-Analyse

Objekte in PHP



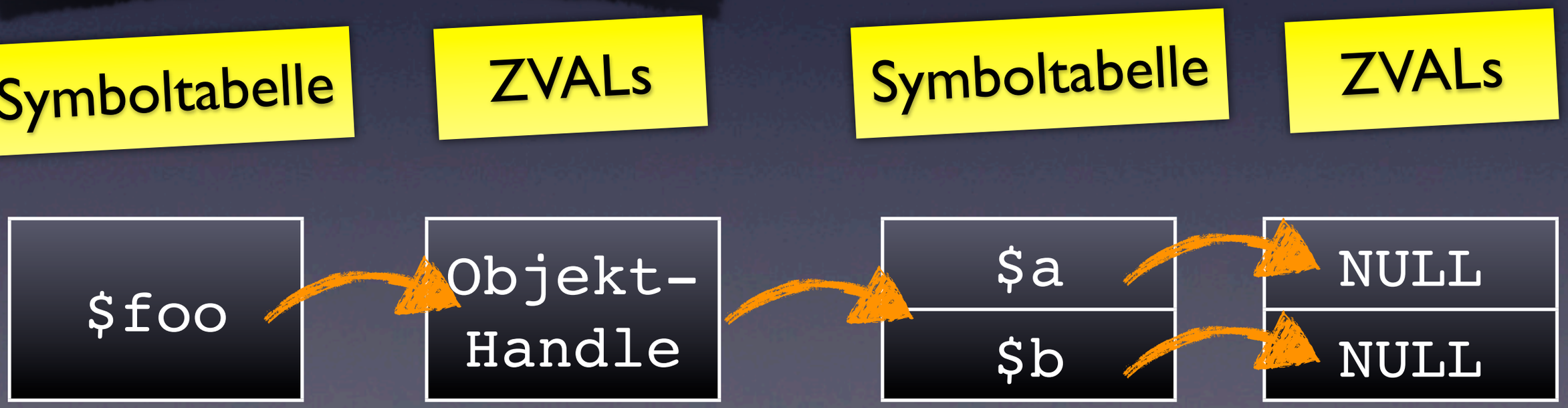
Objektreferenzen



Ausblick

Objekte in PHP sind schichtweise implementiert

```
class Foo {
  public a;
  public b;
}
$foo = new Foo();
```



Neu: Taint-Analyse auf Feld-Ebene

```
class Foo {
  public a;
  public b;
}

$foo = new Foo();
$foo->a = $_GET['a'];
```

foo.a:untainted,
foo.b:untainted

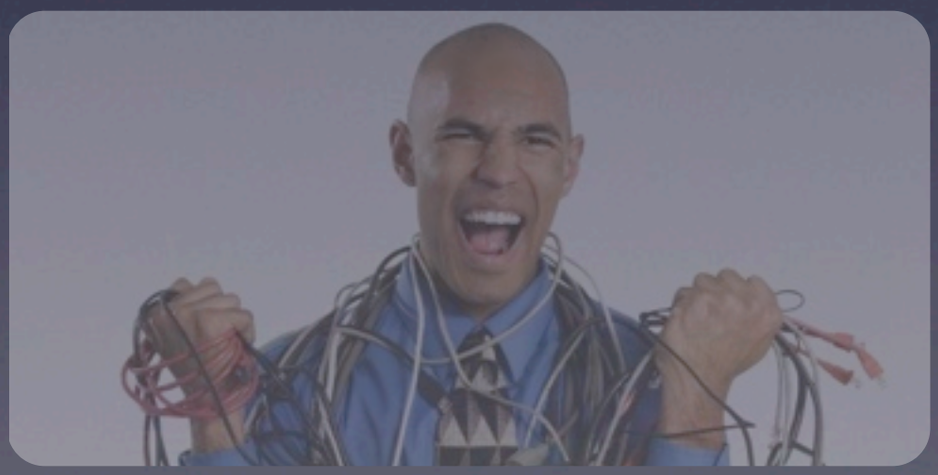
foo.a:tainted,
foo.b:untainted

Einstieg

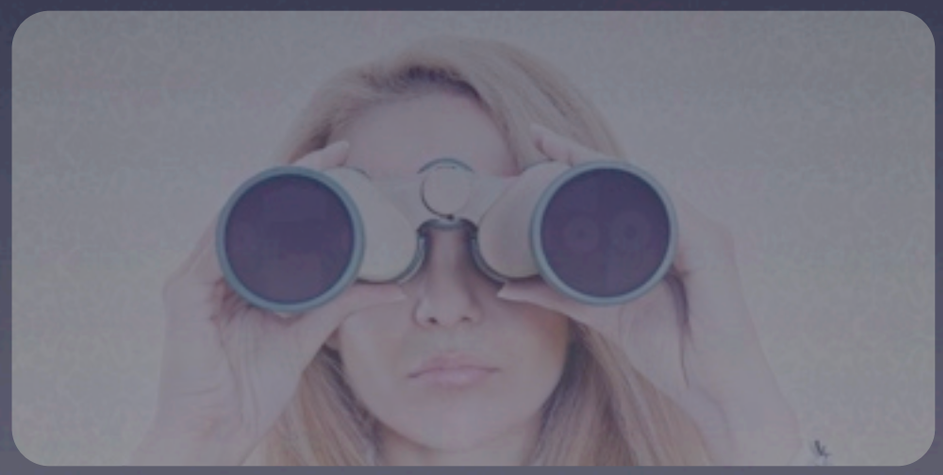
Tainting

Alias-Analyse

Objekte in PHP



Objektreferenzen



Ausblick

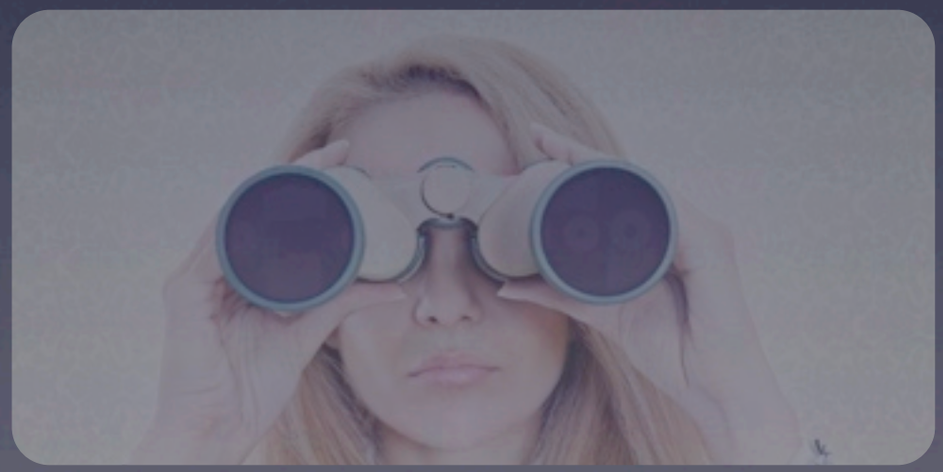
Einstieg

Tainting

Alias-Analyse

Objekte in PHP

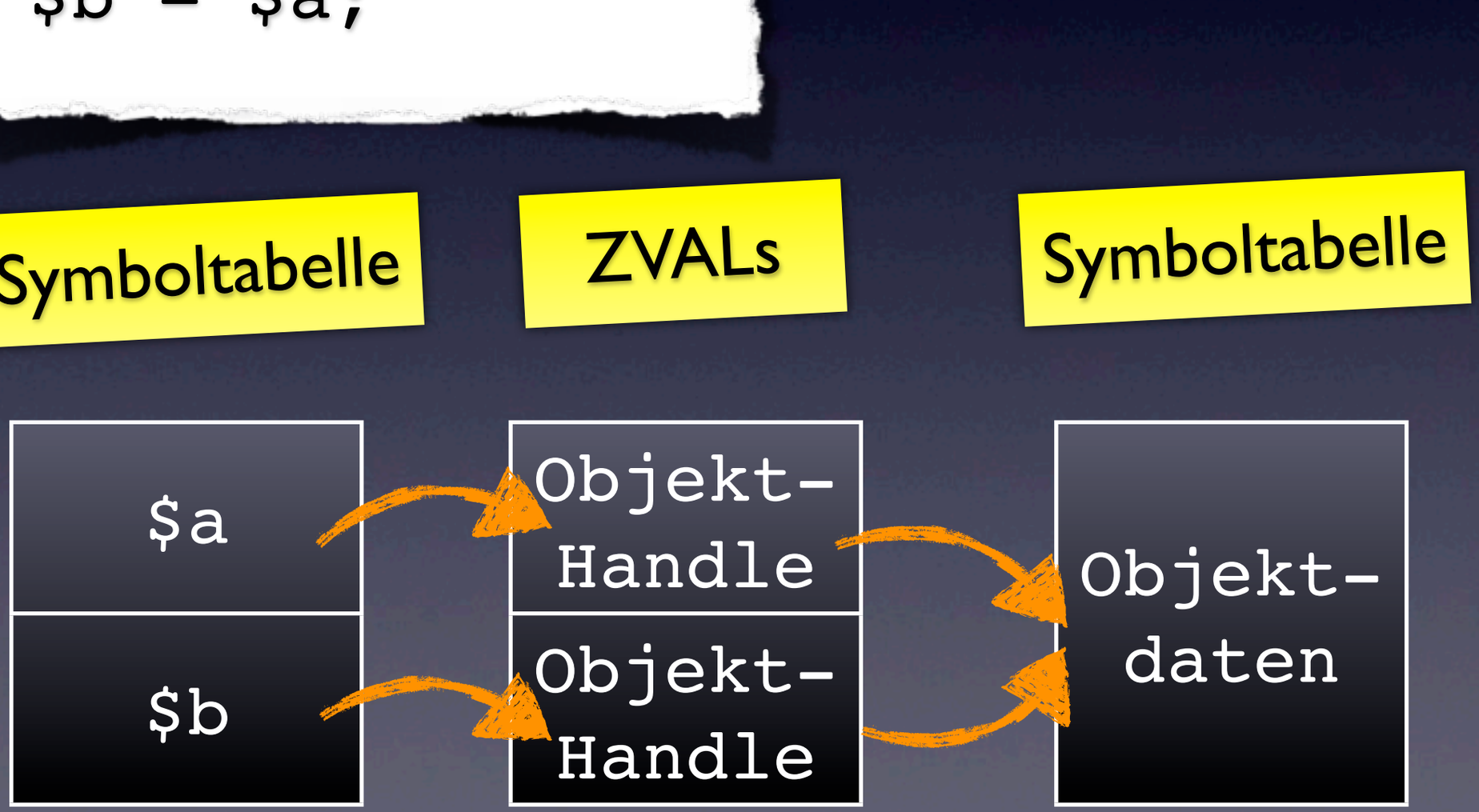
Objektreferenzen



Ausblick

Objekte werden in PHP "per Referenz" übergeben

```
$a = new Foo();
$b = $a;
```



PHP kann auch echte Objekt-Referenzen

```
$a = new Foo();
$b = &$a;
```

Symboltabelle

ZVALs

Symboltabelle

Wir brauchen

Must- und May-Objekt-Referenzen

```
$a1 = new Foo();
$a2 = $a1;
if (...) {
   $a3 = $a1;
}
```

Must = {}, May = {}
Must = {(a1, a2)}, May = {}

Must = {(a1, a2, a3)},
May = {}

Must = {(a1, a2)},
May = {(a1, a3), (a2, a3)}

Tainting mit Objekt-Referenzen funktioniert wie mit Must-/May-Aliasen

```
$x1->a = $_GET['x'];

$x1->a = intval($x1->a);
```

Must = {(x1, x2)}
x1.a:untainted,
x2.a:untainted

x1.a:tainted,
x2.a:tainted

x1.a:untainted,
x2.a:untainted

May-Objekt-Referenzen sind konservativ zu behandeln

```
$x1->a = $_GET['x'];

$x1->a = intval($x1->a);
```

May = {(x1, x2)}
x1.a:untainted,
x2.a:untainted

x1.a:tainted,
x2.a:tainted

x1.a:untainted,
x2.a:tainted

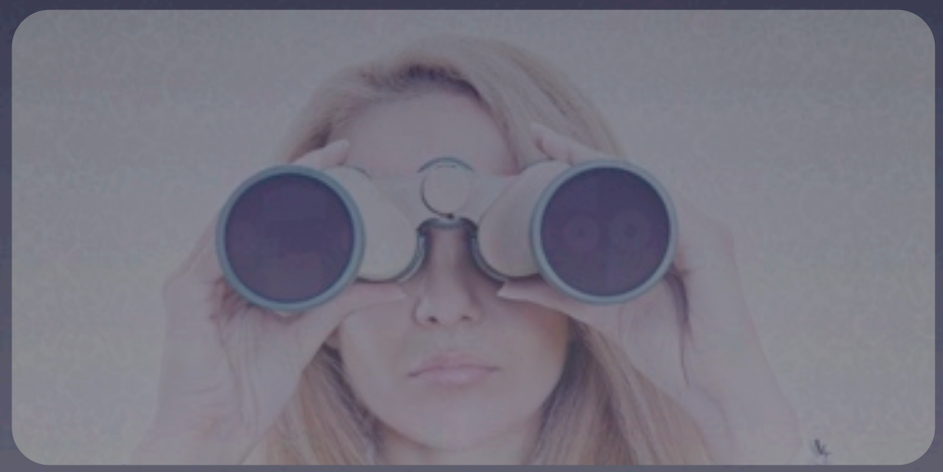
Einstieg

Tainting

Alias-Analyse

Objekte in PHP

Objektreferenzen



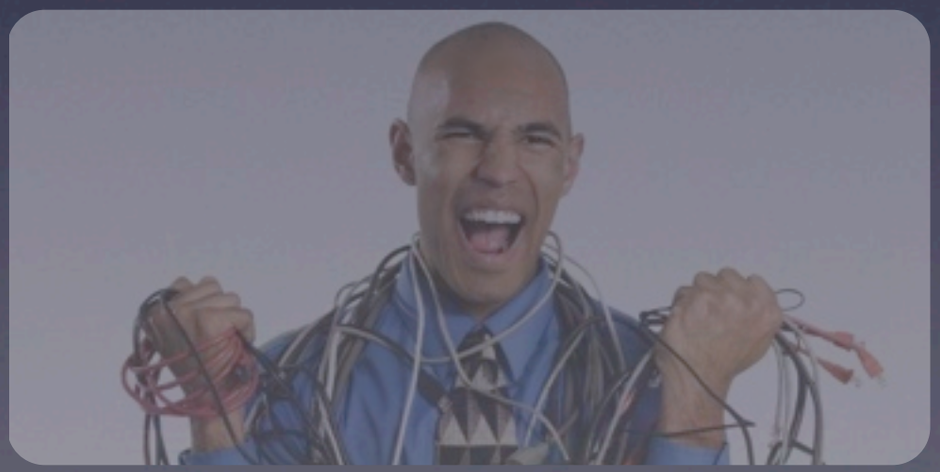
Ausblick

Einstieg

Tainting

Alias-Analyse

Objekte in PHP



Objektreferenzen

Ausblick

PHP 5.x bringt viele weitere Änderungen

neue Schlüsselwörter

```
$dateTime =
date_create_from_format(
  'Y-m-d', '5-Dec-2012'
);
```

Klassenkonstanten

```
class Foo {
  const ANSWER = 42;
}
```

Type-Hinting

Pass-by-Reference per Default für Objekte

```
function bar(Foo $foo) {
  …
}

class Foo {}
$foo = new Foo();
bar($foo);
```

Sichtbarkeit

```
class Foo {
  protected function bar() {}
}
```

Autoloader

```
// nicht mehr nötig
// require_once('Foo.php');
$foo = new Foo();
```

Pixy bietet viele weitere Baustellen

Forschungs-Ideen

- PHP 5-Schlüsselwörter
- Sicherheits-Auswirkungen von PHP5-Neuerungen
- Taint-Analyse für Objekte

PhpParser

- Scanner/Lexer (jFlex)
- Parser (CUP)

Pixy

- TAC
- Kontrollflussgraph (CFG)
- Datenflussgraph (DFG)
- Alias-Analyse
- Tainted Object Propagation

technische Ideen

- PHP5 ohne Fehlermeldung parsen
- Autoloader
- Java 7
- JavaDoc

Am Ende: die Ev(al)uation

Forschungs-Ideen

- PHP 5-Schlüsselwörter
- Sicherheits-Auswirkungen von PHP5-Neuerungen
- Alias-Analyse für Pass-by-Reference

Sicherheitslücken suchen in TYPO3 CMS, Drupal, WordPress, Contao

Open-Source-Projekte parsen: TYPO3 CMS, Drupal, WordPress, Contao

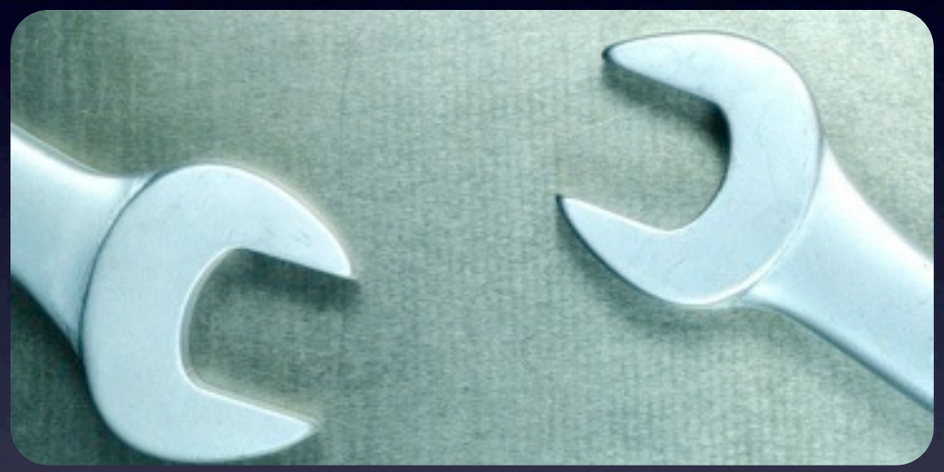
Anzahl der Java-Warnungen

technische Ideen

- PHP5 ohne Fehlermeldung parsen
- Autoloader
- Java 7
- JavaDoc

Einstieg

Tainting



Alias-Analyse

Objekte in PHP

Objektreferenzen

Ausblick

Fragen?

Danke!