Tainted object propagation analysis for PHP 5 based on Pixy

Diploma thesis

Oliver Klee
Bonner Str. 63, 53173 Bonn
pixy@oliverklee.de

Bonn, January 2, 2013

Rheinische Friedrich-Wilhelms-Universität Bonn
Institut für Informatik III
Professor Dr. Armin B. Cremers

universitätbonn

Contents

1 Introduction

1.1 Motivation: Why a current static PHP security scanners is important

Currently, there is no free and high-quality static code analysis tool available (and still maintained) that can find vulnerabilities in PHP 5.4.x code. This is a problem because new vulnerabilities in web applications are found almost daily [osv11], and PHP is used for more than 75 % of the top-million sites [W3T12a], including Facebook (using the HipHop PHP compiler [Zha10], Wikipedia and WordPress.com [W3T12b].

1.2 Research problems and approach

This thesis builds on the PHP security scanner **Pixy** ([JKK07], p. 11) and its subproject **PhpParser** [Jov06].

1.2.1 Research goals

This thesis tackles the following research goals:

- Create an alias analysis that takes PHP 5's pass-by-reference for objects by default into account.
- Enhance the lexer and parser (both part of PhpParser) with most of the new keywords and concepts introduced in PHP 5.0 through 5.4.
- Analyze the security ramifications of the new keywords and concepts introduced in PHP 5.0 through 5..

1.2.2 Technical goals

In addition to the research goals, there are a few technical goals that needed to be achieved in order to achieve the research goals mentioned above:

- Adapt Pixy to work with Java 7 without any warnings. (Pixy was created using at most Java 6, but probably only using Java 1.5.)
- Get Pixy to parse PHP 5 code in the first place. (Pixy currently could handle PHP code only up to PHP version 4.2.)
- Enhance Pixy to also load PHP class files that are not directly included, but are supposed to be loaded via a PHP autoloader.

The technical goals are mostly necessary due to the fact that the Pixy code base had not been maintained (or even touched) since 2006, and both PHP (i.e., the scanned language) as well as Java (i.e., the scanner's language) had evolved in the meantime. In addition, the product code should be maintainable and well-structured so that it will be of real future use instead of a throw-away prototype.

2 PHP

PHP [RBS07] is a server-side web scripting language. In its current version, it is object-oriented and dynamically typed. However, it provides some minimal type safety using type hinting, i.e., function parameters can be typed using class names or "array", and `@var` annotations (which are not evaluated on execution). PHP provides lots of powerful built-in functions for cryptography, string handling, ZIP handling, networking, XML, and more.

2.1 Challenges in static analysis for PHP

In PHP, it is possible to use variables for variable names (which is called "variable variables"), field names, class names or for the inclusion of other classes. This practically is the same as multiple pointers in C++, and poses a problem for static analysis [WHKD00] that forces static analysis to fall back on approximations.

For example, the following constructs are possible, making static analysis a lot harder than e.g. for Java:

```
1  // bar contains the name of the class to instantiate.
2  $foo = new $bar();
3
4  // foo contains the name of the variable that gets assigned a 1.
5  $$foo = 1;
6
7  // classFile includes the path of the class file to include.
8  require_once($classFile);
9
10 // To correctly resolve this include, a scanner would need to parse how
11 // t3lib_extMgm::extPath creates paths.
12 require_once(t3lib_extMgm::extPath('seminars') .
13   'pi2/class.tx_seminars_pi2.php');
```

```php
// Depending on the value of classFlavor, different version of the same
// class will be used. This results in runtime class resolution.
switch ($classFlavor) {
  case FLAVOR_ORANGE:
    require_once('Orange.php');
    break;
  case FLAVOR_VANILLA:
    require_once('Vanilla.php');
    break;
  default;
    require_once('Default.php');
    break;
}
$bar = new MyClass();

// The class file for this class has not been included and will be
// implictly loaded on-demand by the autoloader.
$container = new SmartContainer();
```

In addition, type-hinted parameters can be overwritten within a function:

```php
protected function foo(array $bar) {
  if (empty($bar)) {
    // bar changes its type from an array to an integer.
    $bar = 42;
  }
}
```

3 Vulnerabilities in PHP web applications

4 Static analysis

5 Review of existing static PHP vulnerability scanners

6 Pixy

7 PHP 5.4

Pixy in its current version is only able to deal with PHP 4 code. However, in the meantime PHP has progressed to version 5.4. This version has brought some major changes over 4.x that affects static code analysis:

New language feature	**Effect on static code analysis**
new keywords	language definition for the lexer/parser
constants	the "place" abstraction for variables (three-address code *P-TAC*)
default pass-by-reference	alias analysis
type hinting	lexer/parser, type inference
visibility keywords *private, protected, public*	lexer/parser, control-flow analysis, data-flow analysis
autoloader	loading of class files
namespaces	lexer/parser, loading of class files
late static binding	lexer/parser, control-flow graph
anonymous functions (from PHP 5.4)	lexer/parser, control-flow analysis

Table 7.1: Major changes in PHP 5.4 over 4.x

Note: The new visibility keywords affect both the control-flow analysis as well as the data-flow analysis as they influence which methods can be reached from a class at all and which fields are visible.

The following example demonstrates this issue:

```php
class A {
  /**
   * @var string
   */
  public $publicField = 'public ... ';
  /**
   * @var string
   */
  protected $protectedField = 'protected ... ';
  /**
   * @var string
   */
  private $privateField = 'private ...';
}

class B extends A {
  /**
   * @return string
   */
  public function getFields() {
    return $this->publicField . $this->protectedField .
      $this->privateField;
  }
}

$b = new B();
echo $b->getFields;
```

This example will echo `public ... protected ...` as `$this->privateField` accesses an (undeclared) field `B::privateField` (which will have a default value of `NULL`, which will be automatically cast to an empty string) instead of the existing, but inaccessible `A::privateField`.

8 Alias analysis for the new default pass-by-reference in PHP 5

9 Implementation details and problems encountered

10 Experimental evaluation of the modified version of Pixy

11 Discussion

11.1 Related work

11.2 Conclusions

11.3 Further work

Bibliography

[JKK07] Nenad Jovanovic, Christopher Kruegel, and Engin Kirda. Pixy: XSS and SQLI Scanner for PHP Programs. `http://pixybox.seclab.tuwien.ac.at/pixy/` (retrieved 2010-01-12), 2007.

[Jov06] Nenad Jovanovic. PhpParser. `http://www.seclab.tuwien.ac.at/people/enji/infosys/PhpParser.html` (retrieved 2010-01-12), 2006.

[osv11] OSVDB: The Open Source Vulnerability Database. `http://osvdb.org/` (retrieved 2011-01-10), 2011.

[RBS07] Dagfinn Reiersøl, Marcus Baker, and Chris Shiflett. *PHP in Action*. Manning, Greenwich, 2007.

[W3T12a] W3Techs. Usage of server-side programming languages for websites. `http://w3techs.com/technologies/overview/programming_language/all` (retrieved 2012-11-16), 2012.

[W3T12b] W3Techs. Usage statistics and market share of PHP for websites. `http://w3techs.com/technologies/details/pl-php/all/all` (retrieved 2012-11-16), 2012.

[WHKD00] Chenxi Wang, Jonathan Hill, John Knight, and Jack Davidson. Software Tamper Resistance: Obstructing Static Analysis of Programs. Technical report, Charlottesville, VA, USA, 2000.

[Zha10] Haiping Zhao. HipHop for PHP: Move Fast. `https://developers.facebook.com/blog/post/2010/02/02/hiphop-for-php--move-fast/` (retrieved 2012-11-16), 2010.

List of Figures

List of Tables