Runtime Instrumentation for Precise Flow-Sensitive Type Analysis

Etienne Kneuss, Philippe Suter and Viktor Kuncak

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE, SWITZERLAND

Photo: Hisao Suzuki

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Runtime Instrumentation for Precise Flow-Sensitive Type Analysis for PHP

Etienne Kneuss, Philippe Suter and Viktor Kuncak

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE, SWITZERLAND

Photo: Hisao Suzuki

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Our Starting Point

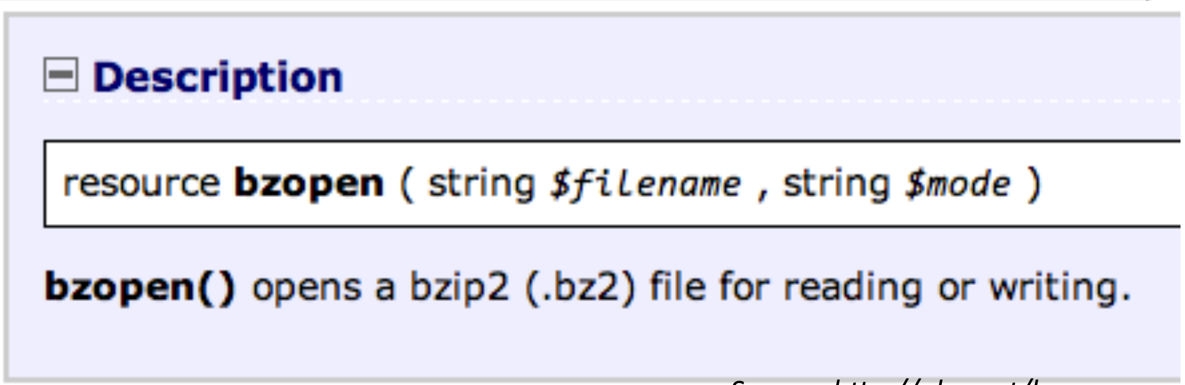
- PHP: The language of the WEB
  - 2007: 20,917,850 domains, 1,224,183 IP addresses

Our Starting Point

- Characteristics of PHP
  - Weakly and dynamically typed (≈ untyped)
    - Implicit conversions for each basic type
    - Versatile arrays/maps
  - No static checks
  - Very few dynamic checks

Many bugs go unnoticed

```
function bzfile($file) {
    $bz = bzopen($file, "r“);
    $str = "";
    while (!feof($bz)) {
        $str = $str . bzread($bz,8192);
    }
    bzclose($bz);
    return $str;
}
```



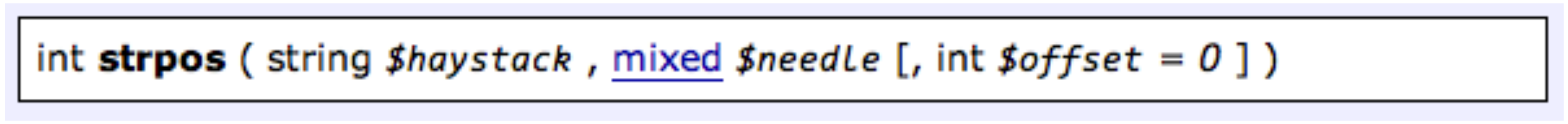
Description

resource **bzopen** (string *$filename* , string *$mode*)

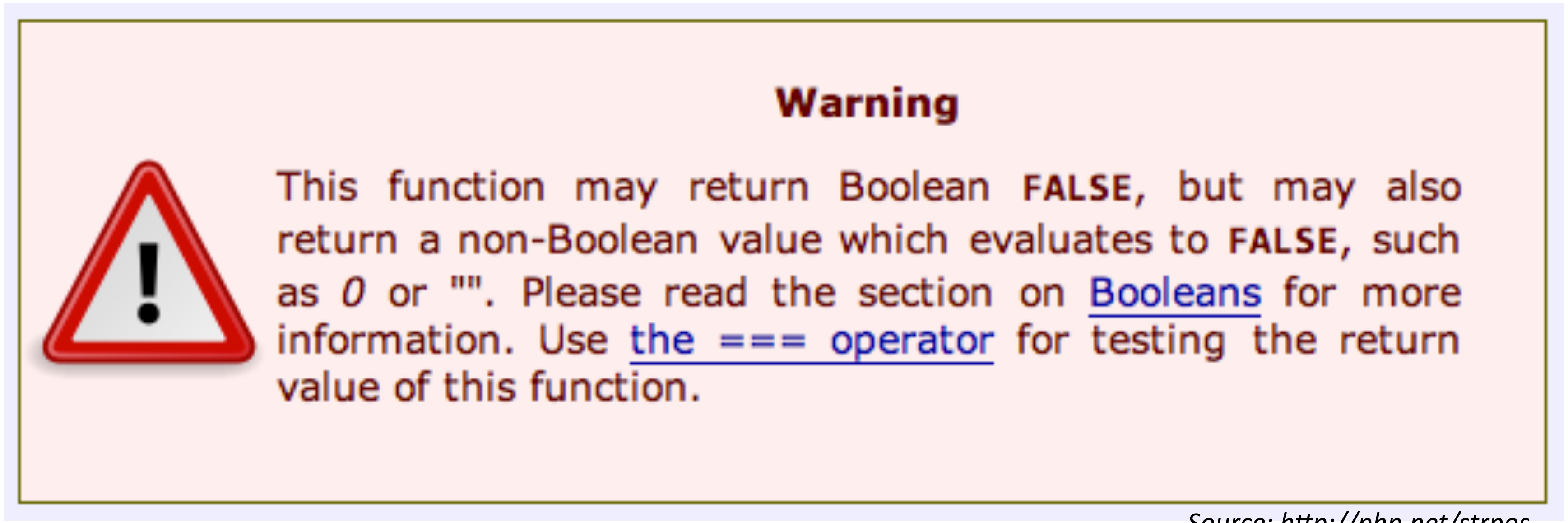
bzopen() opens a bzip2 (.bz2) file for reading or writing.

Source: http://php.net/bzopen

Infinite loop due to unhandled return value



int **strpos** (string *$haystack* , mixed *$needle* [, int *$offset* = *0*])



Warning

This function may return Boolean **FALSE**, but may also return a non-Boolean value which evaluates to **FALSE**, such as *0* or "". Please read the section on Booleans for more information. Use the === operator for testing the return value of this function.

Source: http://php.net/strpos

Functions often have multiple return types phantm

PHP Analyzer for Type Mismatch

- Precise static analyzer
  - Type reconstruction using abstract interpretation
    - Representation of nested data types
    - Union types
  - Flow sensitive
  - Precise handling of conditionals (if, while, foreach)
  - Interprocedural analysis
- Combines static and dynamic analysis
- Practical tool
  - Reduction of false alarms
  - supports latest PHP

Precise Abstract Domain

- Arrays in PHP
  - Maps from strings and/or integers to any value

```
if ($path != “”) {
   $a = array(
      ‘file’ => fopen($path, “r”),
      ‘src’  => $path,
   );
} else {
   $a = “error”;
}
$a ...
```

TString(“error”) ∪ TArray (
  ‘file’ $\rightarrow$ TResource ∪ TFalse,
  ‘src’ $\rightarrow$ TString ,
  else $\rightarrow$ TUndef)

Interprocedural Analysis

- Type information for every built-in PHP functions, constants, and classes (> 4’000)
  - Automatically extracted from PHP’s C source code
- In-code annotations
  - Support for widespread PHPDoc format
- Selective function inlining
  - Function calls are analyzed in their context
- Function prototypes inference

Practical Analysis

- Type refinement
  - Prevent cascading alarms

```
function abs($v) {
  if ($v < 0) {
    return -$v;
  } else {
    return $v;
  }
}
```

Error!

OK!

- Error filtering heuristics
  - Various verbosity levels to hide common false positives

Challenges

Portability

```
if (PHP_VERSION < “5.2.0”) {
    // defined in std lib
    // as of PHP 5.2
    function foo() {
    …
```

Configurability

```
$config =
  parse_ini_file(“conf.ini”);
```

Persistent Storage

```
$r = mysql_query(
   “SELECT * FROM users …”);
$u = mysql_fetch_assoc($r);
$age = $u[‘age’];
```

Pluggable Components

```
foreach($moduleList as $m) {
  include ‘inc/’. $m .’.php’;
}
```

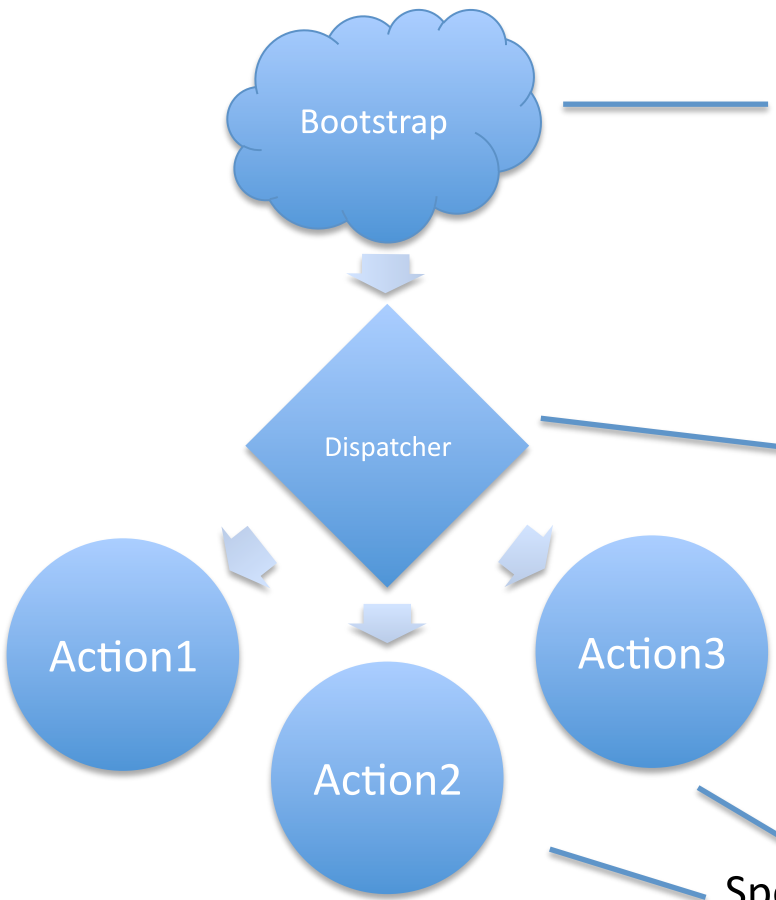
Challenges

- Few (configuration) variables control most of the application:

```
foreach ($moduleList as $module) {
   include ‘inc/’ . $module . ’.php’;
}
```

Purely static analyses is insufficient in practice

Typical Web Application

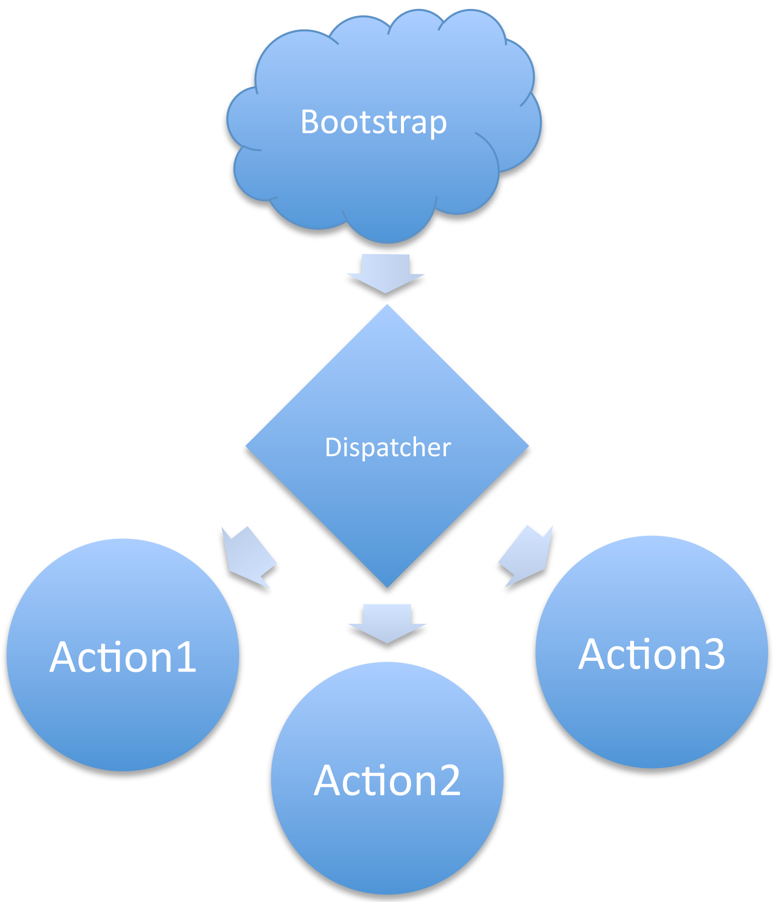


Loading configuration, initializing global structures, connecting to persistent storage layer

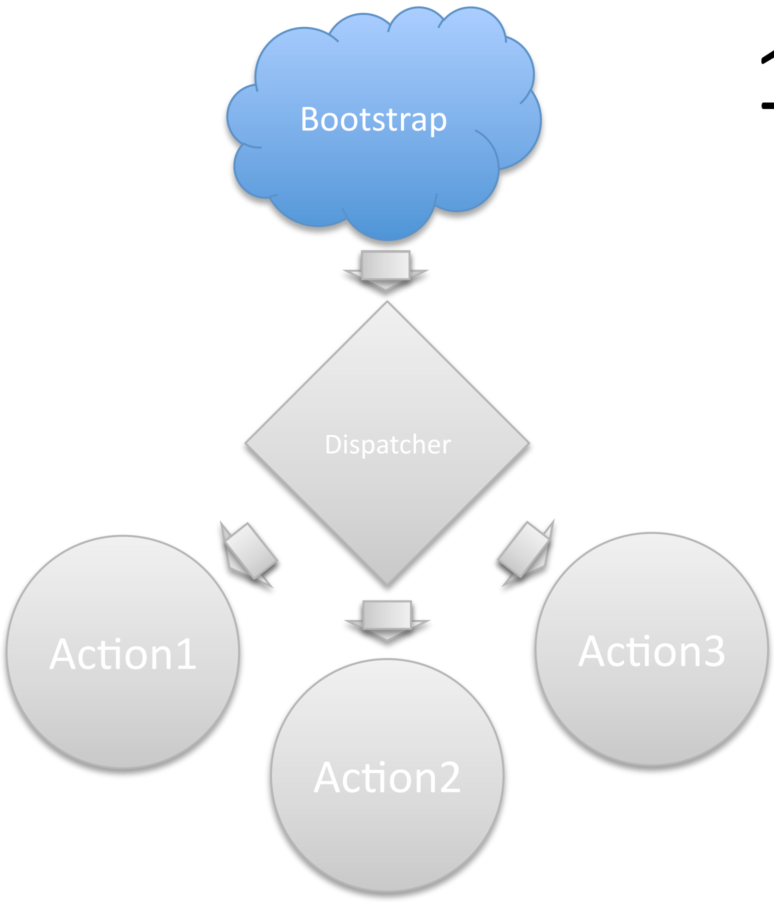
Dispatch control to specific tasks, based on user input (URL, forms, …)

Specific, independent tasks

Our Approach

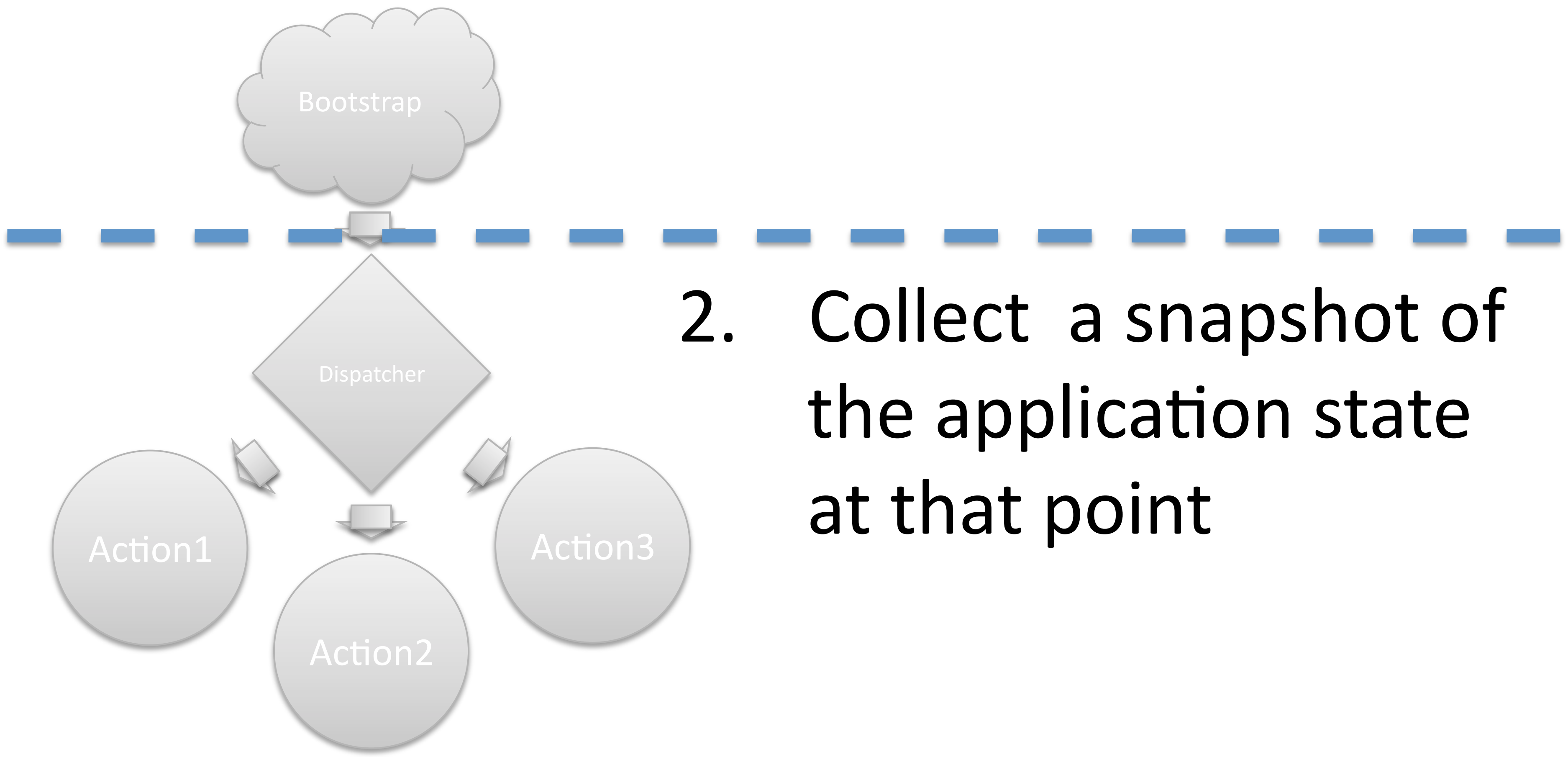


Our Approach



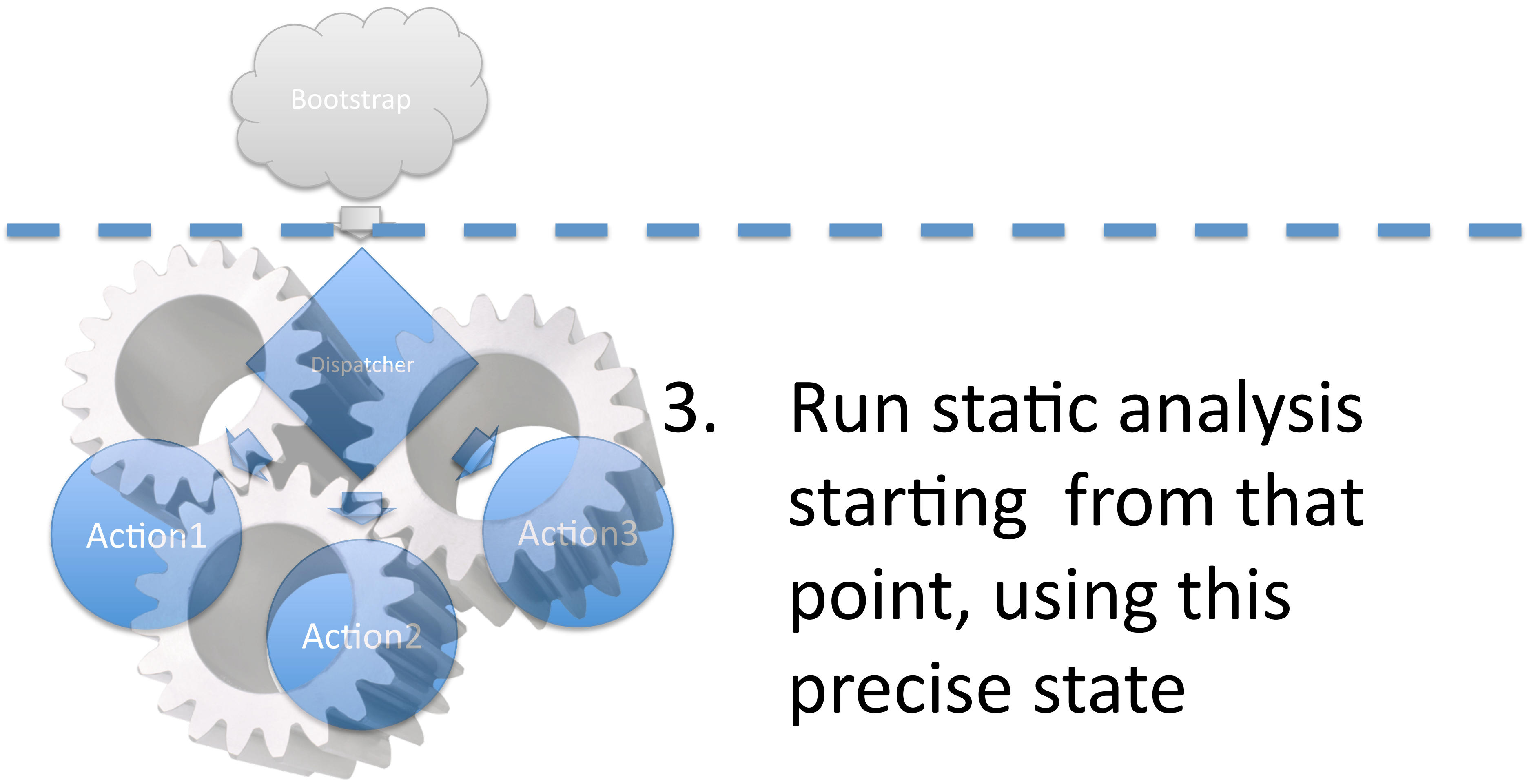
1. Run the application in a realistic environment up to a point

Our Approach



2. Collect a snapshot of the application state at that point

Our Approach



3. Run static analysis starting from that point, using this precise state

Benefits

- We actually know what code to analyze!
- Analysis starts with a "perfect" state
  - PHP interpreter exposes nearly all its state (including the heap)
  - With that state, Phantm will:
    - Represent each values precisely with singleton types
    - Disambiguate function/class declarations

Experimental Results

Photo: Hisao Suzuki

Analyzed Software

DokuWiki

"... a standards compliant, simple to use Wiki, mainly aimed at creating documentation of any kind."

WebMail

"... a free webmail for reading and sending e-mail while on the road from an Internet browser."

SimplePie

"... a very fast and easy-to-use class, written in PHP, that puts the 'simple' back into 'really simple syndication'."

Without Runtime Instrumentation

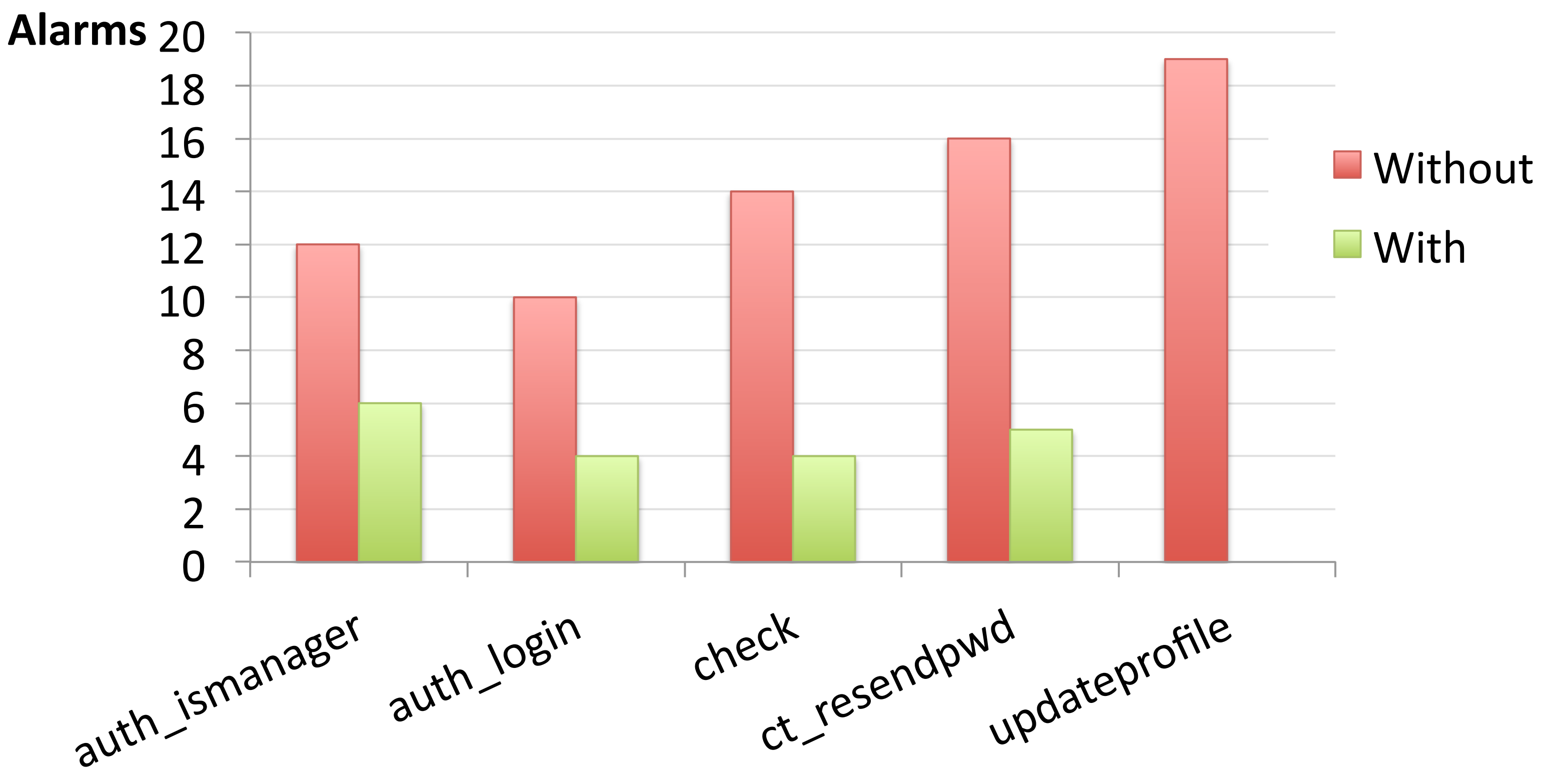
	Lines	Filtered Alarms	Problems	Time
DokuWiki	31486	270	76	244s
WebMail	3621	59	43	11s
SimplePie	15003	327	84	21s
Total:	*50110*	*656*	*203*	*276s*

Quantifying Benefits of Instrumentation

- Instrumenting affects the code analyzed
  - Absolute number of alarms emitted is not a good metric
- We compare the number of alarms, for functions analyzed completely in both cases

Benefits of Instrumentation

- On Average: 12% improvement
- Selected functions that benefited the most:



Bugs Found

```
function encrypt_pass($password, $key) {
  $res = "";
  foreach(str_split($password) as $char) {
    $res .= substr($key, strpos(ALPHABET, $char), 1);
  }
  return $res;
}
```

```
ALPHABET:     "abcd"
$key:         "1398"

$pass:        "ccdf"
⇒ $res:       "9981"
⇒ decrypt:    "ccda"
```

Bugs Found

```
if (!$file->success && !($file->method &
SIMPLEPIE_FILE_SOURCE_REMOTE === 0)) {
  // …
}
```

```
$file->method &
(SIMPLEPIE_FILE_SOURCE_REMOTE === 0)
```

Bugs Found

```
if (strtolower(trim($attribs['mode']) === 'base64’)) {
     // …
}
```

Project Information

- Written in Scala

- Homepage
  - http://lara.epfl.ch/dokuwiki/phantm

- Open source, available from github:
  - http://github.com/colder/phantm

Related Work

- S.H. Jensen, A. Møller, P. Thiemann: *Type analysis for Javascript*. SAS 2009
- M. Furr, J.-h. An, J. S. Foster: *Profile-guided static typing for dynamic scripting languages*. OOPSLA 2009
- N. Jovanovic, C. Kruegel, E. Kirda*: Pixy: A static analysis tool for detecting web applications vulnerabilities*. IEEE Symp. Security and Privacy 2006

phantm

PHP Analyzer for Type Mismatch

- Precise static analyzer
  - Type reconstruction using abstract interpretation
    - Representation of nested data types
    - Union types
  - Flow sensitive
  - Precise handling of conditionals (if, while, foreach)
  - Interprocedural analysis
- Combines static and dynamic analysis
- Practical tool
  - Reduction of false alarms
  - supports latest PHP

Thank you!

Photo: Hisao Suzuki

Code Example

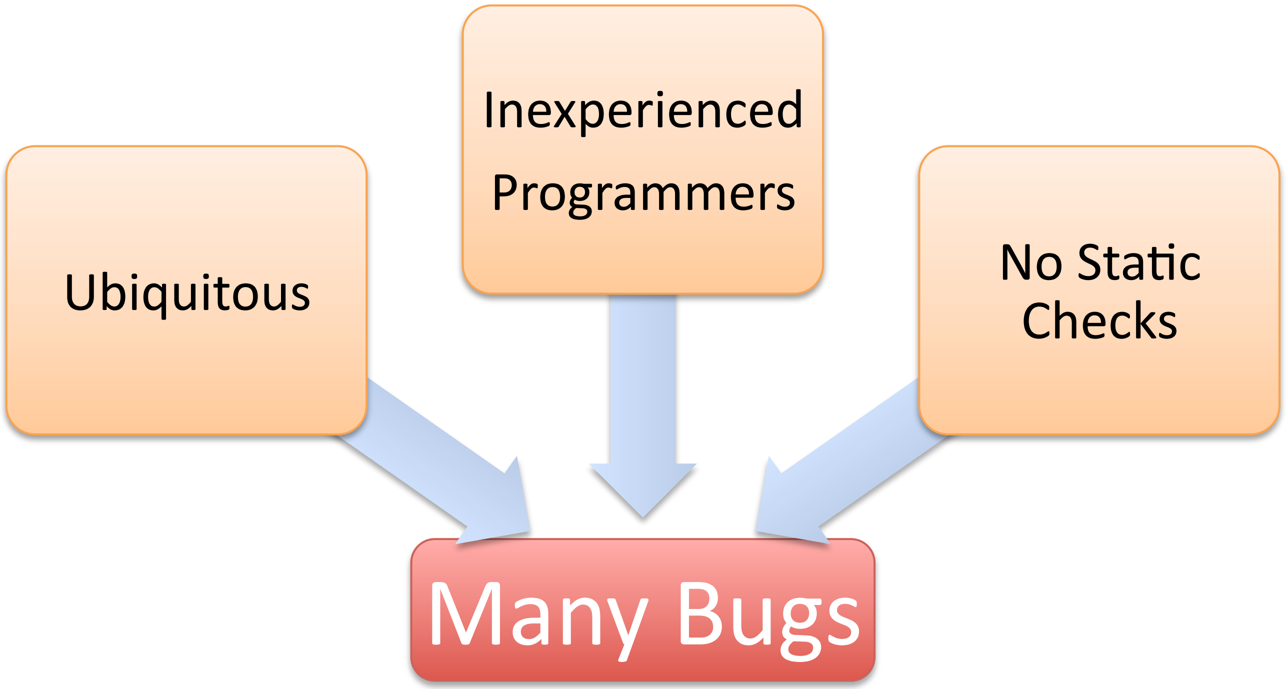
```
include getFile();
$a = array(“foo” => “bar”);
// ...

include ‘path/to/phantm/lib/phantm.php’;
phantm_collect_state(get_defined_vars());

// ...

echo $a[‘foo’];
```

Collected State at ***P***

1. Heap
   – Via serialization
2. Included files
   – Relevant code base
3. Functions and classes defined
   – Disambiguate dynamic definitions

Our Starting Point

- Characteristics of PHP
  - Weakly and dynamically typed (≈ untyped)
    - Implicit conversions for each basic type
    - Versatile arrays/maps

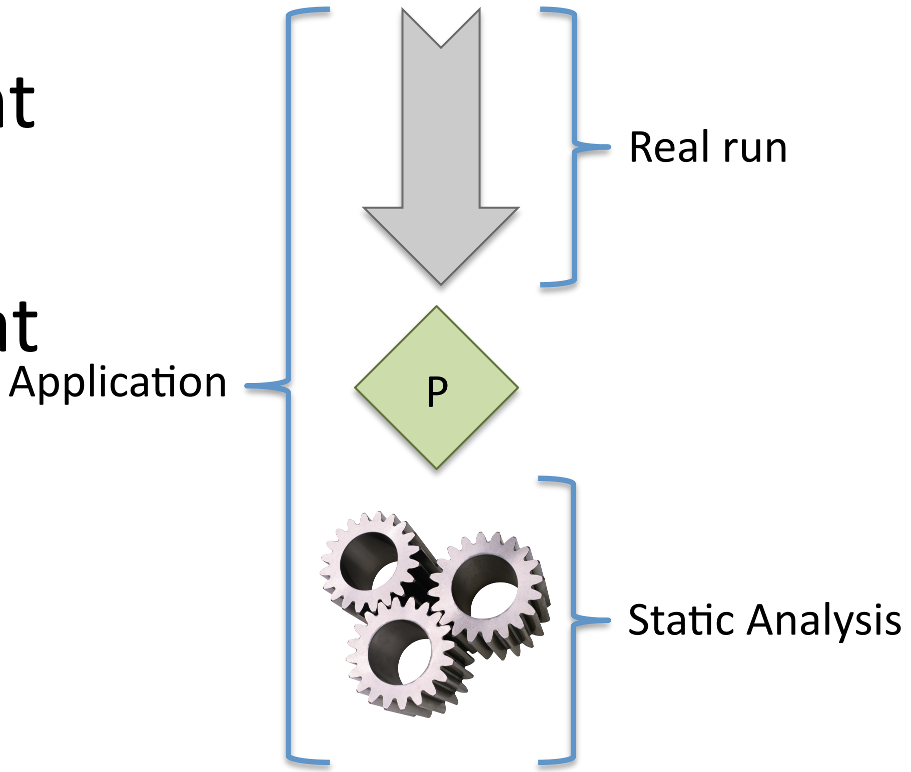


Sources of Imprecisions

- Models are limited by design
- Unknown environment
- User inputs that dictate application behavior

Our Approach

1. Run the application in a realistic environment
2. Collect a snapshot of the application state at point ***P***
3. Run static analysis starting from ***P*** using this precise state

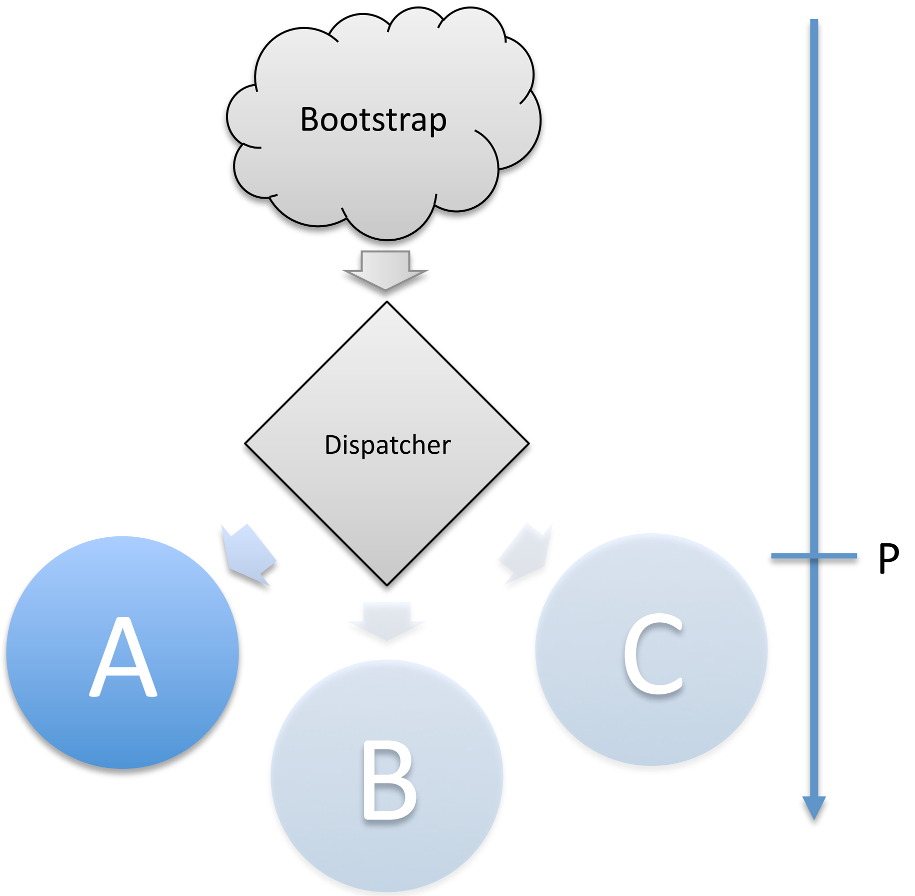


Our Goal:

Useful analysis tool, able to help developers by spotting errors in *realistic, complete applications*

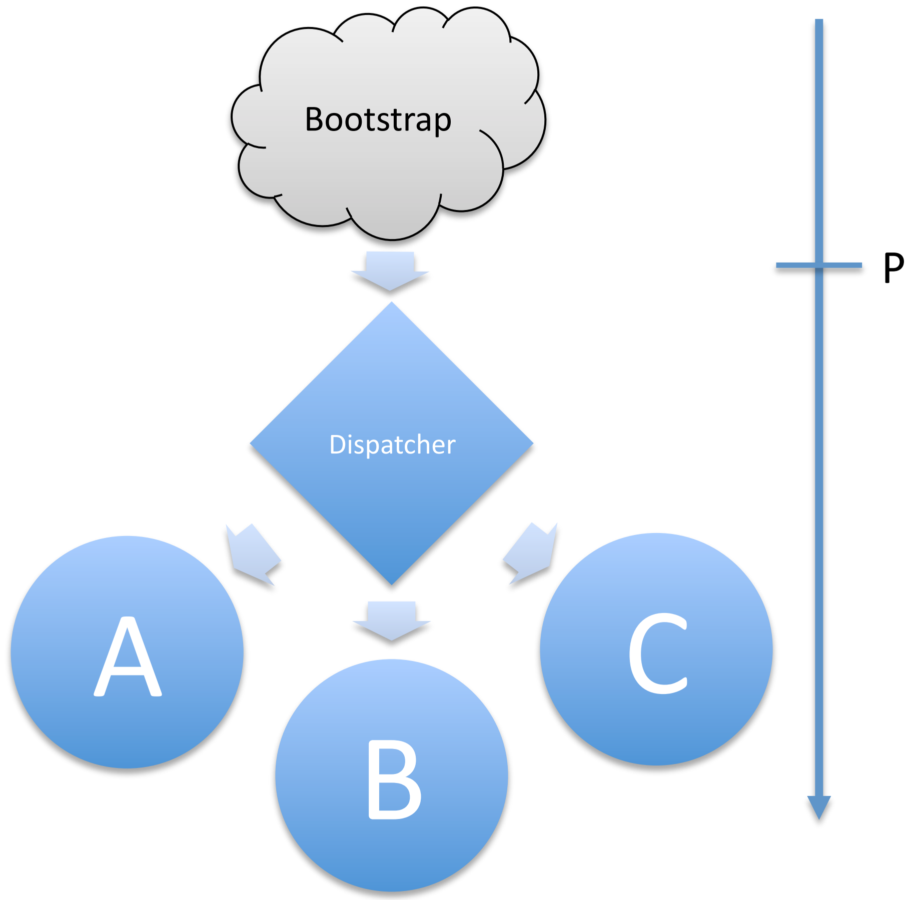
Setting Instrumentation Point ***P***

- Currently, ***P*** is manually placed



Setting Instrumentation Point ***P***

- Currently, ***P*** is manually placed


Bootstrap
Dispatcher
A
B
C
P