

# Phantm: PHP Analyzer for Type Mismatch

Continued as SAV project  
Spring 2010

Etienne Kneuss

# PHP

- Weak & Dynamic Typing
- Compiler optimized for speed, not safety
- Large internal API (> 2500 functions)
- All kinds of dynamic features
  - ... \$\$var, \$name(), new \$class, \$class::\$\$property, eval(), autoloaders, error handlers, ticks ...

# The problem

- Implicit type conversions potentially hiding bugs
- Most errors are non-fatal and happen at runtime
- Until recently, PHP was shipped to not even report those errors by default
  - Lots of broken or badly written scripts

# Why do types matter?

- PHP does type juggling
  - switch
  - ctype\_digit

→ Relying on it is a problem waiting to happen:

[#50696](#), [#49057](#), [#34772](#), [#25763](#), [#24905](#), ...

- Non-scalar types

```
$a = 0;
switch($a) {
    case "foo":
        echo "this";
        Break;
    default:
        echo "that";
        break;
}
```

# Phantom

- several implemented analyses and techniques:
  - Structural checks
  - Semantic checks
  - Data-flow analysis
    - Independent or **context-sensitive / interprocedural**
  - **Pure statements checks**
  - **Runtime instrumentation**
- ~10'000 lines of Scala code

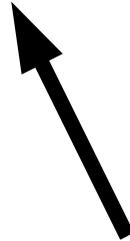
# Analysis phases

- Runtime dumps collection
- *Lexing (Jflex) + Parsing (modified CUP)*
- AST Pruning
- *AST checks*
- Pure statements checks
- *API Importation*
- *Includes and Constants resolutions*
- *Semantic analysis*
- Call graph generation and analysis
- *CFG generations*
- Type analysis
- *API Exportation*

# Pure statement checks

- Detect pure statements, usually indicating bugs:

```
<?php  
  
if ($a == "foo") {  
    $mode = "this";  
} else {  
    $mode == "that";  
}
```



# Runtime Instrumentation

- Run the application, and collect its precise state at some program point
- Analyze statically from that program point, injecting the runtime state.

# Context-sensitive analysis

- It is often not precise enough to specify function prototypes, for instance:

```
<?php
function identityOrFalse($val) {
    If (is_int($val)) return $val; else return false;
}
```

```
IdentityOrFalse(2) + 2; // we don't expect any error
IdentityOrFalse("foo") + 2; // we expect an error
```

# Short Demo

Thank you