IDE Support for PHP Code Analysis

author: Natália Tyrpáková, supervisor: Mgr. David Hauzar, Charles University in Prague

Motivation – Using Code Analysis For Error Detection

- PHP language is the most common language for web application development. Code errors in these applications are especially dangerous since they might lead to security vulnerabilities.
- Danger usually originates from user input.
- To detect code errors, automated code analysis is convenient however it has to be supported by additional manual review of the analysis results

Some of the possible application attacks:
- Cross-site scripting
- SQL injection
- File include manipulation
- Semantic URL attacks
- Password sniffing
- Session hijacking
- Command injection

An example of SQL injection vulnerable code:

```
…
$email = $_GET ['email '];
if ($email != '') {
    $query = 'SELECT * FROM users
        WHERE email =" ' .$email. ' " ';
    $result = mysql_query ($query);
}
else {
    $result = ' ';
}
…
```

Problem:
The $_GET['email'] information comes from user input and it is not checked for validity.

Expectation:
Input: user@example.com
Output: the user with given email address

Possible situation:
Input: " OR 1 OR email = "
Resulting query: 'SELECT * FROM users WHERE email ="" OR 1 OR email = "" '
Output: all the records from the table *users*

As a result, attacker might be able to access the information which is most likely sensitive.

Our Solution

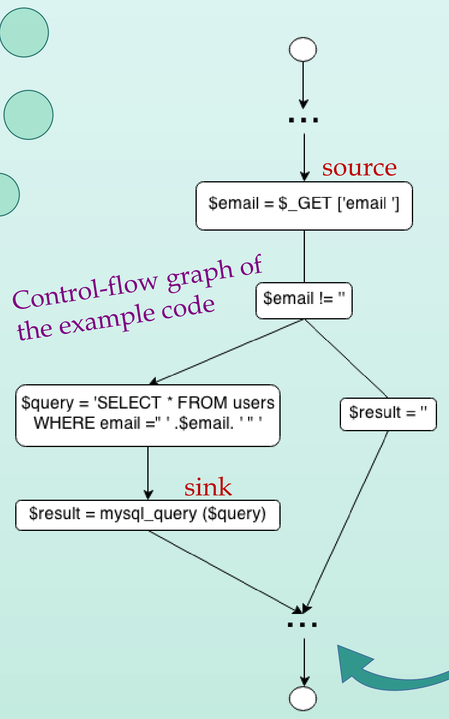
Create an IDE support for PHP code analysis

Code Flaws Analysis
- We analyzed the most common code flaws leading to security vulnerabilities
- As a result, we have found which information is needed to detect them

Weverca Analyzer
- We used Weverca analyzer to get this information
- Weverca is an open-source static analysis framework for PHP
- Provides possibility to define end-user analyses

Taint Analysis
- We extended Weverca analyzer by taint analysis
- Keeps track of the values originating in user input

Eclipse Plug-ins
- We visualized the gathered information in a form of Eclipse plug-ins

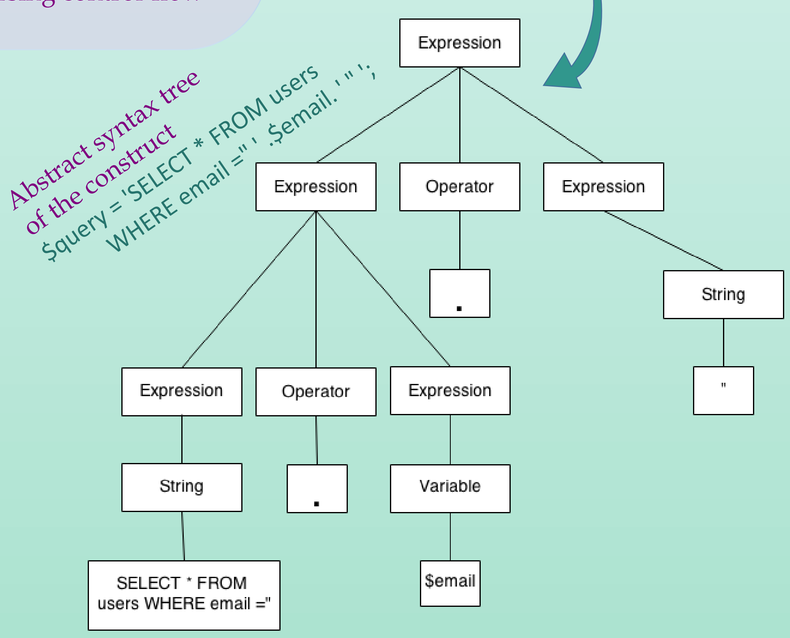


Control-flow graph of the example code

Code Analysis

Static analysis
- Gathers run-time information of a program independently of its inputs
- Result is valid for all possible program executions
- Provides:
  - Error search
  - Information about possible variable values and types
  - Unreachable code detection
  - Unused local variables detection
  - …
- Based on traversing control-flow graph

Syntactic analysis
- Ensures that the code conforms to the rules of the programming language grammar
- Provides other information
  - use of obsolete constructs
  - metric information
- Creates an abstract syntax tree representing structure of the source code



Taint Analysis

- While traversing an existing control-flow graph, taint information is collected
- Taint information contains the whole taint flows from source to sink and distinguishes three different types of taint:
  - HTML taint, SQL taint, File path taint
  - Each taint has a priority defined that is high if all the possible flows lead to a tainted value
- Besides the taint information, null values are propagated too

Evaluation

- Our plug-ins help to find code errors that might lead to security vulnerabilities:
- **Automatically** - by directly pointing out the code flaws that may lead to a vulnerability
  - cross-site scripting, SQL injection, file include manipulation
- **Semi-automatically** - by inspecting potential values of variables and their taint information
  - spoofed form submission, semantic URL attacks

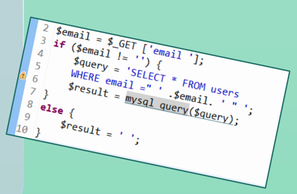
Time Consumption
- Tested application: NOCC (Webmail client) Number of lines: 6135
  - Static analysis
    - Overall time: 2m 49s
    - Overhead of the plug-ins: 11,8%
  - Metric computation
    - Overall time: 673ms
    - Overhead of the plug-ins: 12,5%

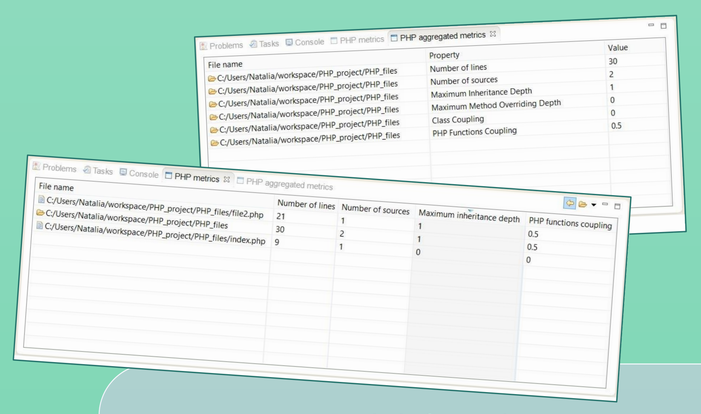
Related Tools
- The most relevant tool that we found is TAJS
- TAJS is an open-source JavaScript analyzer
- Also provides an Eclipse plug-in
- Uses static analysis primarily to find variable types and to create a call graph of calling relations of subroutines
- Does not put focus on security vulnerabilities detection

Eclipse plug-ins

Analysis Visualization
All the information is in a form of code highlight or clickable views that allow simple navigation. This helps to accelerate and simplify manual code review, which was our primary goal.

Constructs Highlight
- A possibility of highlighting specified constructs is available
- Helps to avoid use of dangerous constructs or to notice constructs that might lead to code errors
- The construct occurrences can be marked on-the-fly or they can be searched using a search page
- In our example code, the mysql_query function would be automatically marked as a warning





Metric information
- Software metric is measure of some property or characteristic of a piece of code
- The plug-ins provide an option to compute metric information of selected files and folders
- It is helpful for code optimization or analyzing a source code with security risks

Static Analysis
- Static code analysis can be executed on a single file or a set of files.
- It is always complemented by a taint analysis
- Provides a lot of useful information:
  - Unreachable code
  - Values and types of variables
  - Code errors warnings, including dangerous flows from user input into a sinks such as database or browser
- In our example code, static analysis would raise a security warning with a taint flow from the source at line 2 to the sink at line 6