北京邮电大学

网络空间安全学院

软件详细设计报告

项目：基于源代码检测和动态执行的二阶 SQL 注入漏洞检测

组员：王硕、彭致远、李懿飞、王晨旭

2020 年 11 月 6 日

目录

1 引言

1.1 目的

编写该详细设计说明书是为说明该项目——基于源代码检测和动态执行的二阶 SQL 注入漏洞检测工具的设计考虑，包括程序描述、输入/输出、算法和流程逻辑等，为软件编程和系统维护提供基础。本说明书的预期读者为系统设计人员、软件开发人员、软件测试人员和项目评审人员。其中系统设计人员、软件开发人员、软件测试人员为小组内部成员组成，软件评审人员为课程老师或助教。

1.2 背景及范围

- 项目名称：基于源代码检测和动态执行的二阶 SQL 注入漏洞检测工具
- 项目成员：北京邮电大学网络空间安全学院“网络安全分析实践”课程开发小组
  - 王硕（组长）：2018213641
  - 彭致远：2018213646
  - 李懿飞：2018213632
  - 王晨旭：2018213636
- 系统范围：具有 PHP7.4 环境的 Windows 系统计算机
- 用户：无限制
- 实现项目的计算机网络：校园网

1.3 定义、术语或缩略语

序号	术语或缩写	解释
1	SQL	结构化查询语言(Structured Query Language)简称 SQL，是一种特殊目的的编程语言，是一种数据库查询和程序设计语言，用于存取数据以及查询、更新和管理关系数据库系统。
2	AST	抽象语法树（Abstract Syntax Tree，AST），是源代码语法结构的一种抽象表示。它以树状的形式表现编程语言的语法结构，树上的每个节点都表示源代码中的一种结构。
3	CFG	控制流程图，是一个过程或程序的抽象表现，是用在编译器中的一个抽象数据结构，由编译器在内部维护，代表了一个程序执行过程中会遍历到的所有路径。它用图的形式表示一个过程内所有基本块执行的可能流向，也能反映一个过程的

		实时执行过程。本项目用于描述变量在流动过程的执行过程
4	PHP-Parser	开源工具，用于生成 PHP 文件的语法分析树
5	SQL-Parser	开源工具，用于生成 SQL 语句的语法分析树

2 软件系统结构

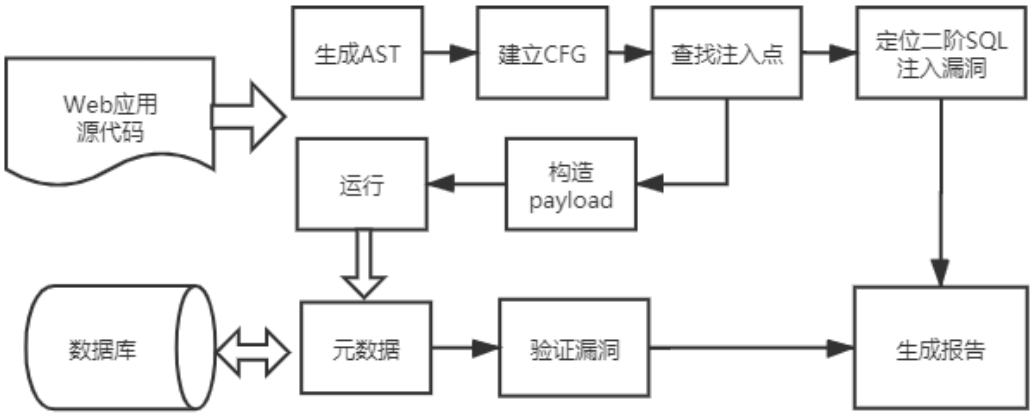
2.1 需求概述

开发一个基于源代码检测和动态执行的二阶 SQL 注入漏洞检测工具。源代码检测是系统中的主要部分，输入主要是 PHP 网站项目的源代码，包括前端生成应用程序源代码的 CFG 和后端的查找注入点、触发点，以及通过污点分析定位二阶 SQL 的注入漏洞，并生成漏洞报告。动态分析起到辅助作用，包括前期信息搜集获取的元数据和后期漏洞的验证。

2.2 子模块划分

2.2.1 系统模型图

本工具的系统模型图如下所示：



2.2.2 系统流程描述

本工具实现的过程总结如下：

- 以 PHP 项目源码作为输入，对每个 PHP 文件生成语法分析树，进而对每个 PHP 文件内的变量生成变量的控制流图，通过“超全局变量”“引用”“函数调用”等建立各个 PHP 文件之间的关系，从而生成所有变量的控制流图。
- 分析变量在流动过程中的来源、去向、变化。发现敏感数据即发现二阶 SQL 注入

的注入点和触发点，保存为静态检测的结果

- 通过动态的执行，注入点输入 payload，观察数据库内数据的变化，如果符合二阶 SQL 注入漏洞的特点，即生成报告

2.2.3 子模块划分

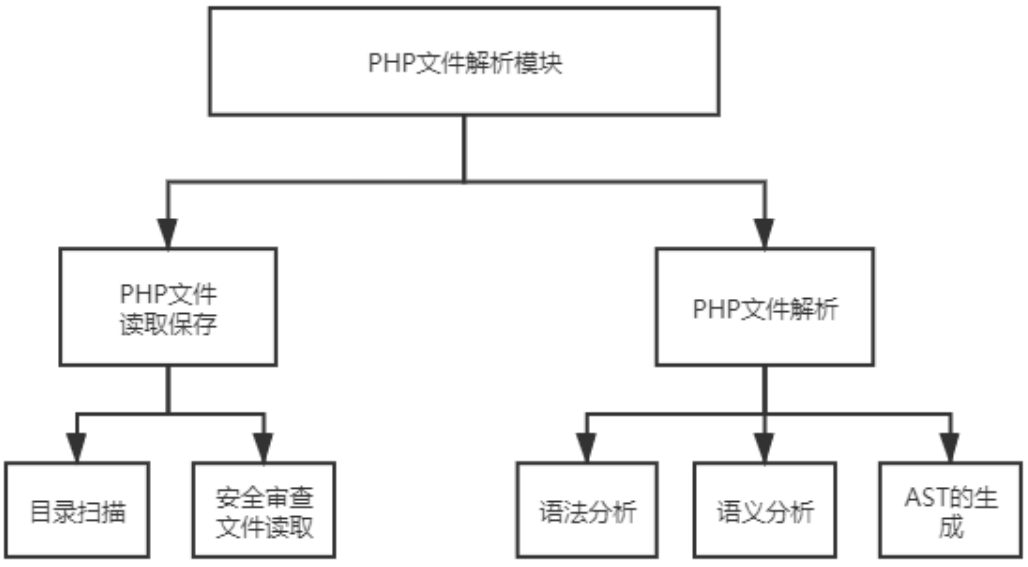
根据系统模型，可以将本模块划分为五个模块，分别是：

1） PHP 文件解析模块
2） 变量控制流图生成模块
3） SQL 语言解析模块
4） 回溯扫描查找模块
5） 动态注入测试模块

3 各模块详细设计

3.1 PHP 文件解析模块

3.1.1 模块功能描述



本模块的主要功能为：扫描用户 PHP 网站项目目录，从中提取 PHP 文件，对每个 PHP 文件，进行词法分析、语法分析、语义分析等，最终生成每个 PHP 文件对应的语法分析树，并提供一套递归的遍历方法，从而便于下一个模块从中递归的提取变量，并判断变量的深度（全局、局部、循环内等）

3.1.2 获取 PHP 项目文件

3.1.2.1 思路

如果考虑图形化界面，直接拖拽项目即可。这里采用控制台交互，故引导用户将 web 项目源代码放入指定文件夹内，输入项目名称，即可开始扫描。并判断文件是否存在，对用户的输入做检查。

3.1.3.2 算法

```
1.  echo "请将 php 项目放入桌面\n 请输入项目名称: ";
2.
3.  $name = fgets(STDIN);
4.  while (true){
5.      $name = substr($name, 0, strpos($name, "\n"));
6.      $dir = "C:\\Users\\23959\\Desktop\\" . $name;
7.      if(is_dir($dir)){
8.          break;
9.      }
10.     else{
11.         echo "项目文件不存在，请重新输入";
12.         $name = fgets(STDIN);
13.     }
14. }
15. echo "\n 正在进行扫描项目".$name."...\n\n";
16.
17. fetchPhpFile('C:/Users/23959/Desktop/'.$name, $file_name_list, $file_content
    s_list);
18. // 输出待扫描的 php 文件
19. echo "Scan Files:\n";
20. foreach ($file_name_list as $item){
21.     echo $item."\n";
22. }
23.
24. //获取所有 php 文件，文件名+文件内容
25. function fetchPhpFile($dir, &$array_name, &$array_contents) {
26.     foreach(glob($dir.'/*') as $file) {
27.         if(is_dir($file)) {
28.             fetchPhpFile($file, $array_name, $array_contents);
29.         }else{
30.             $file_name = basename($file);
31.             if(substr($file_name, -4) == '.php'){
```

```
                $handle = fopen($file, "r");
                $contents = fread($handle, filesize ($file));
                array_push($array_contents, $contents);
                array_push($array_name, $file_name);
                fclose($handle);
            }
        }
    }
}
```

3.1.3 PHP 文件生成语法分析树

3.1.3.1 思路

我们的检测模型主要针对 PHP 语言，选用开源工具 PHP-Parser，通过语法分析，语义分析等步骤生成抽象语法树即可。

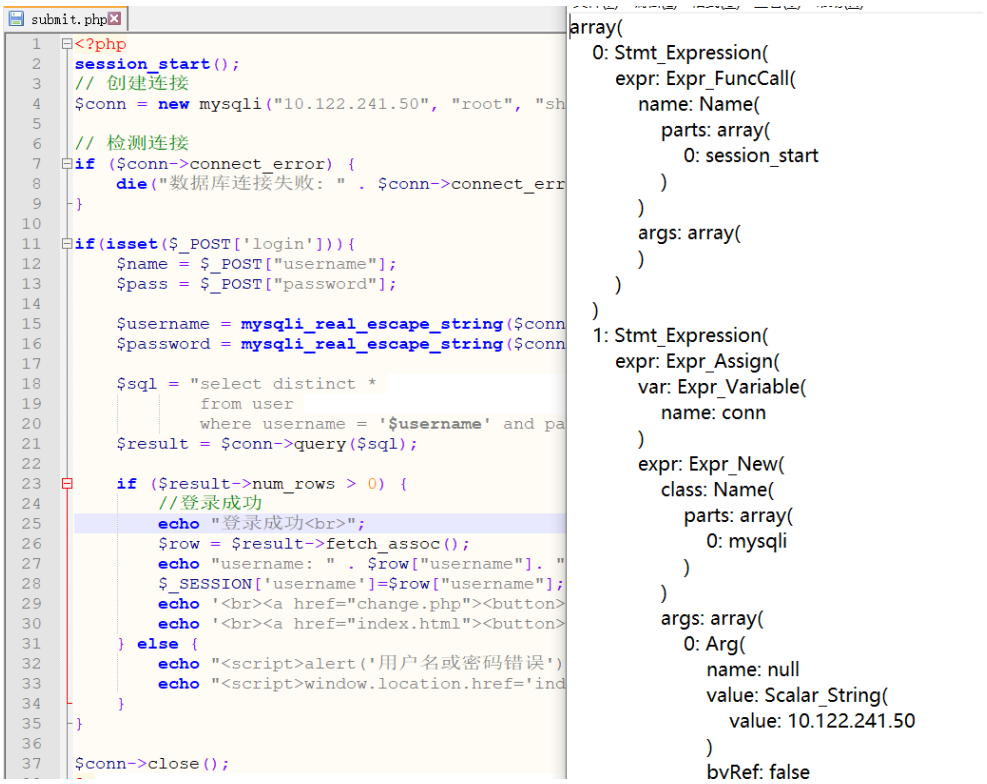
3.1.3.2 算法

有了 PHP-Parser 工具，生成 AST 的算法较为简单，如下所示：

```
// 生成语法树
$parser = (new ParserFactory)->create(ParserFactory::PREFER_PHP7);
try {
    for($i = 0; $i < count($file_name_list); $i++){
        $ast = $parser->parse($file_contents_list[$i]);
        array_push($ast_list, $ast);
        $dumper = new NodeDumper;
        array_push($dumper_list, $dumper->dump($ast));
        //echo $dumper->dump($ast)."\n";
    }
} catch (Error $error) {
    echo "Parse error: {$error->getMessage()}\n";
    return;
}
```

3.1.3.3 输出效果

生成的语法分析树实际上是树状数据结构，根节点即当前 PHP 文件，子节点可以是类、函数、表达式、数组、变量等等，利用 print_r（）函数将语法分析树打印出来效果如下：源文件为左边的 submit.php 文件，输出后为右边的树状结构。



3.1.4 AST 结点遍历的算法

由于是树状结构，且每个结点的类型都是 Node 类型，递归的进行遍历即可，部分算法如下：

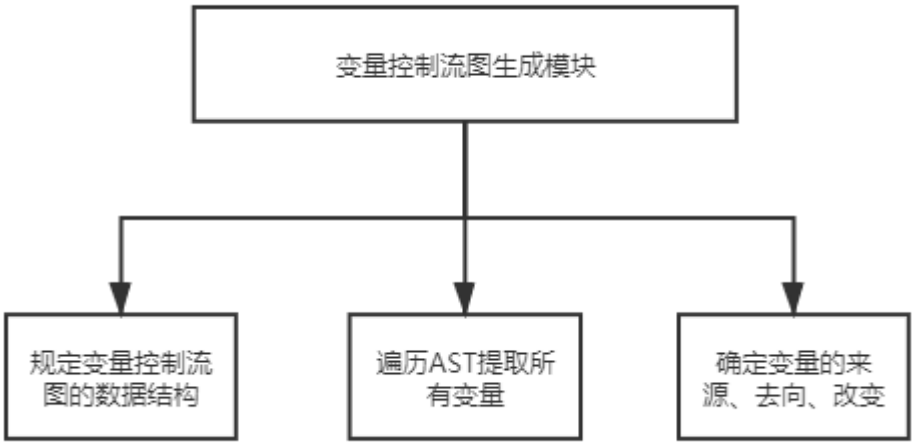
```
/**
* 递归遍历当前节点
* @param Node $node 待遍历的节点
* @return Node 遍历的结果（可能是新节点、也可能是旧节点）
*/
rotected function traverseNode(Node $node) : Node {
    foreach ($node->getSubNodeNames() as $name) {
        $subNode =& $node->$name;
        // 子节点还是数组，继续递归遍历
        if (\is_array($subNode)) {
            $subNode = $this->traverseArray($subNode);
            if ($this->stopTraversal) {
                break;
            }
        }
        //不是的话，断掉
        elseif ($subNode instanceof Node) {
            $traverseChildren = true;
            $breakVisitorIndex = null;
```

```
            // 对节点的每一个属性，进行赋值（例如节点类型、左右节点、节点名称等）
            foreach ($this->visitors as $visitorIndex => $visitor) {
                $return = $visitor->enterNode($subNode);
                if (null !== $return) {
                    if ($return instanceof Node) {
                        $this->ensureReplacementReasonable($subNode, $return);
                        $subNode = $return;
                    } elseif (self::DONT_TRAVERSE_CHILDREN === $return) {
                        $traverseChildren = false;
                    } elseif (self::DONT_TRAVERSE_CURRENT_AND_CHILDREN === $return) {
                        $traverseChildren = false;
                        $breakVisitorIndex = $visitorIndex;
                        break;
                    } elseif (self::STOP_TRAVERSAL === $return) {
                        $this->stopTraversal = true;
                        break 2;
                    } else {
                        throw new \LogicException(
                            'enterNode() returned invalid value of type ' . gettype($return)
                        );
                    }
                }
            }
            if ($traverseChildren) {
                $subNode = $this->traverseNode($subNode);
                if ($this->stopTraversal) {
                    break;
                }
            }
            foreach ($this->visitors as $visitorIndex => $visitor) {
                $return = $visitor->leaveNode($subNode);

                if (null !== $return) {
                    if ($return instanceof Node) {
                        $this->ensureReplacementReasonable($subNode, $return);
                        $subNode = $return;
                    } elseif (self::STOP_TRAVERSAL === $return) {
                        $this->stopTraversal = true;
                        break 2;
                    } elseif (\is_array($return)) {
```

```
                        throw new \LogicException(
                            'leaveNode() may only return an array ' .
                            'if the parent structure is an array'
                        );
                    } else {
                        throw new \LogicException(
                            'leaveNode() returned invalid value of type ' . gettype($return)
                        );
                    }
                }
                if ($breakVisitorIndex === $visitorIndex) {
                    break;
                }
            }
        }
    }
    return $node;
```

3.2 变量控制流图生成模块

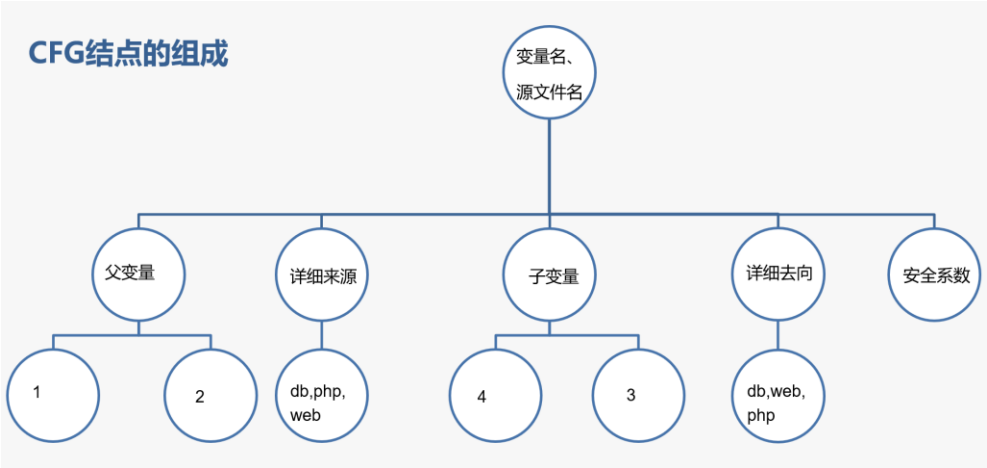
3.2.1 模块功能描述



本模块的功能主要是通过 PHP 文件生成的语法分析树以及上一个模块提供的遍历 AST 的方法，提取 PHP 文件内部的变量，并通过变量之间的传递，建立变量的控制流图。通过“超全局变量”“引用”“函数调用”等建立各个 PHP 文件之间的关系，从而生成所有变量的控制流图。

3.2.2 CFG 结点的数据结构

要想确定变量的来源、去向，以及它在传递过程中发生了哪些变化，首先要确定变量的位置，即文件名+变量名，为了确定来源，就要有父变量的数组；为了确定去向，就要有自变量的数组。为了能检测出来二阶 SQL 注入漏洞的危险变量，就要确定它的最终来源和去向，即是否来源于/去向数据库、是否来源于/去向 web 网页，是否来源于/去向于 PHP 文件内部。所以 CFG 结点的结构图如下：



3.2.3 通过 AST 提取所有变量名

3.2.3.1 VariableVisitor 类

利用上一个模块提供的遍历方法，编写 VariableVisitor 类，遇到节点类型为 PhpParser\Node\Expr\Variable 即返回，代码如下：

```
1.  class VariableVisitor extends NodeVisitorAbstract {
2.      public array $v_name;
3.      public function __construct() {
4.          $this->v_name = array();
5.      }
6.      public function enterNode(Node $node)
7.      {
8.          if ($node instanceof PhpParser\Node\Expr\Variable){
9.              array_push($this->v_name, $node->name);
10.             //echo $node->name . "\n";
11.         }
12.     }
13.     public function getVName(): array
14.     {
15.         return $this->v_name;
```

```
    }
}
```

3.2.3.2 提取变量名

循环遍历每一棵语法分析树即可，代码如下：

```
//提取变量
$variable_list = array();
foreach ($file_name_list as $item=>$value){
    $traverser = new NodeTraverser();
    $var_visitor = new VariableVisitor();
    $traverser->addVisitor($var_visitor);
    $modifiedStmts = $traverser->traverse($ast_list[$item]);
    fetchVariableList($value, $var_visitor->getVName(),$variable_list);
}

function fetchVariableList($file_name, $var_list_tmp, &$var_list){
    foreach ($var_list_tmp as $var){
        //[var_name, var_dir, src(db,php,web), src_var, dst(db,php,web), dst_var, safe_level]
        $arr = array();
        $var_info = [$var, $file_name, "", $arr, "", $arr, 0];
        if($var == "_POST" || $var == "_GET" || $var == "_FILE")
            $var_info[2] = "web";
        elseif ($var == "_SESSION")
            $var_info[2] = "db";
        $flag = false;
        for($i = 0; $i < count($var_list); $i++){
            if($var_list[$i][0] == $var && $var_list[$i][1] == $file_name){

                $flag = true;
            }
        }
        if($flag == false)
            array_push($var_list, $var_info);
    }
}
```

3.2.4 根据变量流动确定变量的来龙去脉

变量的来源/去向主要分类如下几类：

- 表达式 x=y

- 数组获取 x=y[“index”]
- 函数调用 x=f(y, z)
- 方法调用 x=y->f(y)
- 超全局变量的传递
- Web 表单
- 数据库

下面进行逐个实现：

3.2.4.1 表达式

普通的表达式即 x=y+z 之类的其实在实现下一块“数组”一节后，实际上就是“零维数组”的运算，只需要查看下面数组一节如何实现的就可以。

3.2.4.2 数组

对于多维数组，其实也是一棵树，需要重新写一个遍历方法来遍历数组内的每一个元素，思路同上述遍历节点的算法，遍历数组的算法如下：

```
1.  function traverseArray(array $nodes) : array {
2.        $doNodes = [];
3.        foreach ($nodes as $i => &$node) {
4.            if ($node instanceof Node) {
5.                $traverseChildren = true;
6.                $breakVisitorIndex = null;
7.                foreach ($this->visitors as $visitorIndex => $visitor) {
8.                    $return = $visitor->enterNode($node);
9.                    if (null !== $return) {
10.                       if ($return instanceof Node) {
11.                           $this->ensureReplacementReasonable($node, $return);

12.                           $node = $return;
13.                       } elseif (self::DONT_TRAVERSE_CHILDREN === $return) {
14.                           $traverseChildren = false;
15.                       } elseif (self::DONT_TRAVERSE_CURRENT_AND_CHILDREN ===
    $return) {
16.                           $traverseChildren = false;
17.                           $breakVisitorIndex = $visitorIndex;
18.                           break;
19.                       } elseif (self::STOP_TRAVERSAL === $return) {
20.                           $this->stopTraversal = true;
21.                           break 2;
22.                       } else {
23.                           throw new \LogicException(
```

```
                                'enterNode() returned invalid value of type ' .
    gettype($return)
                            );
                        }
                    }
                }
                if ($traverseChildren) {
                    $node = $this->traverseNode($node);
                    if ($this->stopTraversal) {
                        break;
                    }
                }
                foreach ($this->visitors as $visitorIndex => $visitor) {
                    $return = $visitor->leaveNode($node);

                    if (null !== $return) {
                        if ($return instanceof Node) {
                            $this->ensureReplacementReasonable($node, $return);

                            $node = $return;
                        } elseif (\is_array($return)) {
                            $doNodes[] = [$i, $return];
                            break;
                        } elseif (self::REMOVE_NODE === $return) {
                            $doNodes[] = [$i, []];
                            break;
                        } elseif (self::STOP_TRAVERSAL === $return) {
                            $this->stopTraversal = true;
                            break 2;
                        } elseif (false === $return) {
                            throw new \LogicException(
                                'bool(false) return from leaveNode() no longer
    supported. ' .
                                'Return NodeTraverser::REMOVE_NODE instead'
                            );
                        } else {
                            throw new \LogicException(
                                'leaveNode() returned invalid value of type ' .
    gettype($return)
                            );
                        }
                    }
                    if ($breakVisitorIndex === $visitorIndex) {
                        break;
```

```
                }
            }
        } elseif (\is_array($node)) {
            throw new \LogicException('Invalid node structure: Contains nested arrays');
        }
    }
    if (!emptyempty($doNodes)) {
        while (list($i, $replace) = array_pop($doNodes)) {
            array_splice($nodes, $i, 1, $replace);
        }
    }
    return $nodes;
}
```

实现遍历数组的方法后，就可以对数组的元素进行获取了，下面列出对一维数组和二维数组的遍历提取方法，数组的来源和去向相当于对遍历路径的每一个结点，都要进行分析。如果有某个结点的参数为某个遍历，如果该变量存在风险，那么表达式左边的变量也存在风险。只有当数组某个结点的遍历路径均没有危险时，表达式左边的遍历才不存在危险。

```
if ($node instanceof PhpParser\Node\Stmt\Expression){
 if($node->expr->getType() == "Expr_Assign"){
    if($node->expr->var->getType() == "Expr_ArrayDimFetch")
        $variable_left = $node->expr->var->var->name;
    else
        $variable_left = $node->expr->var->name;
    $index_left = findIndex($variable_left, $this->dir, $this->variable_list);

    if($node->expr->expr->getType() == "Expr_ArrayDimFetch"){
        if($node->expr->expr->var->getType() == "Expr_ArrayDimFetch")
            $variable_right = $node->expr->expr->var->var->name;
        else
            $variable_right = $node->expr->expr->var->name;
        $index_right = findIndex($variable_right, $this->dir, $this->variable_list);

        if($this->variable_list[$index_right][2] != ""){
            $this->variable_list[$index_left][2] = $this->variable_list[$index_right][2];
        }

        if($variable_right == "_POST"){
```

```
            $dim_value_right = $node->expr->expr->dim->value;
            $this->variable_list[$index_left][2] = "web_form(".$dim_value_right.")";
        }elseif($variable_right == "_GET"){
            $dim_value_right = $node->expr->expr->dim->value;
            if($variable_left == "fName" && $this->variable_list[$index_left][1] == "download.php"){
                $this->variable_list[$index_left][2] = "db(file(".$dim_value_right."))";
            }
            else
                $this->variable_list[$index_left][2] = "web_form(".$dim_value_right.")";
        }elseif($variable_right == "_FILES"){
            $dim_value_right = $node->expr->expr->dim->value;
            $this->variable_list[$index_left][2] = "web_file(".$dim_value_right.")";
        } elseif ($variable_right == "_SESSION"){
            $dim_value_right = $node->expr->expr->dim->value;
            $this->variable_list[$index_left][2] = "db(user(".$dim_value_right."))";
        }
        array_push($this->variable_list[$index_left][3], $index_right);
        array_push($this->variable_list[$index_right][5], $index_left);

    }
}
```

3.2.4.3 函数调用

函数调用，主要分析函数内部的参数，只要函数的某一个参数来源于 web 表单或者数据库，那么表达式左边的变量即存在风险，标记为对应的来源于 web 还是来源于数据库，算法相比数组较为简单，代码如下所示：

接上面的 if，下面为 else if，表示在遇到变量流动时候的一些条件：

```
else if($node->expr->expr->getType() == "Expr_FuncCall" || $node->expr->expr->getType() == "Expr_New"){
    $flag = true;
    foreach($node->expr->expr->args as $item){
        if($item->value->getType() == "Expr_Variable"){
            $flag = false;
            $variable_right = $item->value->name;
            $index_right = findIndex($variable_right, $this->dir, $this->variable_list);
```

```
8.                 array_push($this->variable_list[$index_left][3], $index_right);
9.                 array_push($this->variable_list[$index_right][5], $index_left);
10.                if($this->variable_list[$index_left][2] == ""){
11.                    if($this->variable_list[$index_right][2] != "php"){
12.                        $this->variable_list[$index_left][2] = $this->variable_list[$index_right][2];
13.                    }
14.                }
15.            }
16.        }
17.        if($flag){
18.            $this->variable_list[$index_left][2] = "php";
19.        }
20.    }
21.    else if($node->expr->expr->getType() == "Expr_FuncCall"){
22.        $flag = true;
23.        foreach($node->expr->expr->args as $item){
24.            if($item->value->getType() == "Expr_Variable"){
25.                $flag = false;
26.                $variable_right = $item->value->name;
27.                $index_right = findIndex($variable_right, $this->dir, $this->variable_list);
28.                array_push($this->variable_list[$index_left][3], $index_right);
29.                array_push($this->variable_list[$index_right][5], $index_left);
30.                if($this->variable_list[$index_left][2] == ""){
31.                    if($this->variable_list[$index_right][2] != "php"){
32.                        $this->variable_list[$index_left][2] = $this->variable_list[$index_right][2];
33.                    }
34.                }
35.            }
36.        }
37.        if($flag){
38.            $this->variable_list[$index_left][2] = "php";
39.        }
40.
41.        $white_list = [
42.            "mysql_real_escape_string", "mysqli_real_escape_string",
43.            "addslashes", "htmlspecialchars", "strip_tags",
44.            "magic_quotes_gpc", "register_globals",
```

```
    ];
    $func_name = $node->expr->expr->name->parts[0];
    $flag = true;
    foreach ($white_list as $safe_func){
        if($func_name == $safe_func){
            $this->variable_list[$index_left][6] = 1;
            $flag = false;
            break;
        }
    }
    if ($flag){
        $rtn = analyseFunction($func_name);
        $this->variable_list[$index_left][6] = $rtn;
    }
}
```

3.2.4.4 方法调用

方法调用相比于函数调用，它没有等于号，即不算是一个表达式，但是方法调用的时候，也会对变量进行操作，也有可能存在方法参数，而参数的来源不安全，也会导致调用方法的变量存在风险。所以也是存在变量的流动的。只有当方法内部的变量、参数都安全时，被调用方法的变量才安全，算法如下：

```
else if($node->expr->expr->getType() == "Expr_MethodCall") {
  $flag = true;
  $variable_right = $node->expr->expr->var->name;
  $index_right = findIndex($variable_right, $this->dir, $this->variable_list);
  array_push($this->variable_list[$index_left][3], $index_right);
  array_push($this->variable_list[$index_right][5], $index_left);
  if ($this->variable_list[$index_right][2] != "" && $this->variable_list[$index_right][2] != "php") {
      $this->variable_list[$index_left][2] = $this->variable_list[$index_right][2];
  } else {
      foreach ($node->expr->expr->args as $item) {
          if ($item->value->getType() == "Expr_Variable") {
              $flag = false;
              $variable_right = $item->value->name;
              $index_right = findIndex($variable_right, $this->dir, $this->variable_list);
              array_push($this->variable_list[$index_left][3], $index_right);
              array_push($this->variable_list[$index_right][5], $index_left);
              if ($this->variable_list[$index_left][2] == "") {
```

```
                if ($this->variable_list[$index_right][2] != "php") {
                    $this->variable_list[$index_left][2] = $this->variable_
list[$index_right][2];
                }
            }
        }
    }
    if ($flag) {
        $this->variable_list[$index_left][2] = "php";
    }
}
```

3.2.4.5 超全局变量的传递

（1）PHP 的超全局变量

PHP 中的许多预定义变量都是“超全局的”，这意味着它们在一个脚本的全部作用域中都可用。在函数或方法中无需执行 global $variable; 就可以访问它们。

- $GLOBALS　一个包含了全部变量的全局组合数组。变量的名字就是数组的键。
- $_SERVER　是一个包含了诸如头信息(header)、路径(path)、以及脚本位置(script locations)等等信息的数组。这个数组中的项目由 Web 服务器创建。
- $_GET　通过 URL 参数（又叫 query string）传递给当前脚本的变量的数组。
- $_POST　当 HTTP POST 请求的 Content-Type 是 form-data 时，会将变量以关联数组形式传入当前脚本。
- $_FILES　通过 HTTP POST 方式上传到当前脚本的项目的数组。
- $_COOKIE　通过 HTTP Cookies 方式传递给当前脚本的变量的数组。
- $_SESSION　当前脚本可用 SESSION 变量的数组。
- $_REQUEST　— HTTP Request 变量，默认情况下包含了 $_GET，$_POST 和 $_COOKIE 的数组。
- $_ENV　通过环境方式传递给当前脚本的变量的数组。

（2）利用超全局变量可以建立 PHP 文件间的联系

实际上超全局变量为基本上是数组的形式，如 GET[“username”]等等，所以在数组里加入超全局变量的解析即可，部分算法如下：

```
if($node->expr->expr->getType() == "Expr_ArrayDimFetch"){
 if($node->expr->expr->var->getType() == "Expr_ArrayDimFetch")
     $variable_right = $node->expr->expr->var->var->name;
 else
     $variable_right = $node->expr->expr->var->name;
 $index_right = findIndex($variable_right, $this->dir, $this->variable_list)
;

 if($this->variable_list[$index_right][2] != ""){
```

```
        $this->variable_list[$index_left][2] = $this->variable_list[$index_right][2];
    }

    $dim_value_right = $node->expr->expr->dim->value;
    if($variable_right == "_POST"){
        $var = "_POST[".$dim_value_right."]";
        $file_name = $this->variable_list[$index_left][1];
        $arr = array();
        $var_info = [$var, $file_name, "", $arr, "", $arr, 0];
        array_push($var_list, $var_info);
        $this->variable_list[$index_left][2] = "web_form(".$dim_value_right.")";
    }elseif($variable_right == "_GET"){
        $var = "_POST[".$dim_value_right."]";
        $file_name = $this->variable_list[$index_left][1];
        $arr = array();
        $var_info = [$var, $file_name, "", $arr, "", $arr, 0];
        array_push($var_list, $var_info);
        analyseGlobalVariable("_GET", $dim_value_right,$this->dir, $this->variable_list);
    }elseif($variable_right == "_FILES"){
        $this->variable_list[$index_left][2] = "web_file(".$dim_value_right.")";
    } elseif ($variable_right == "_SESSION"){
        $var = "_SESSION[".$dim_value_right."]";
        $file_name = $this->variable_list[$index_left][1];
        $arr = array();
        $var_info = [$var, $file_name, "", $arr, "", $arr, 0];
        array_push($var_list, $var_info);
        analyseGlobalVariable("_SESSION", $dim_value_right, $this->dir, $this->variable_list);
    }
```

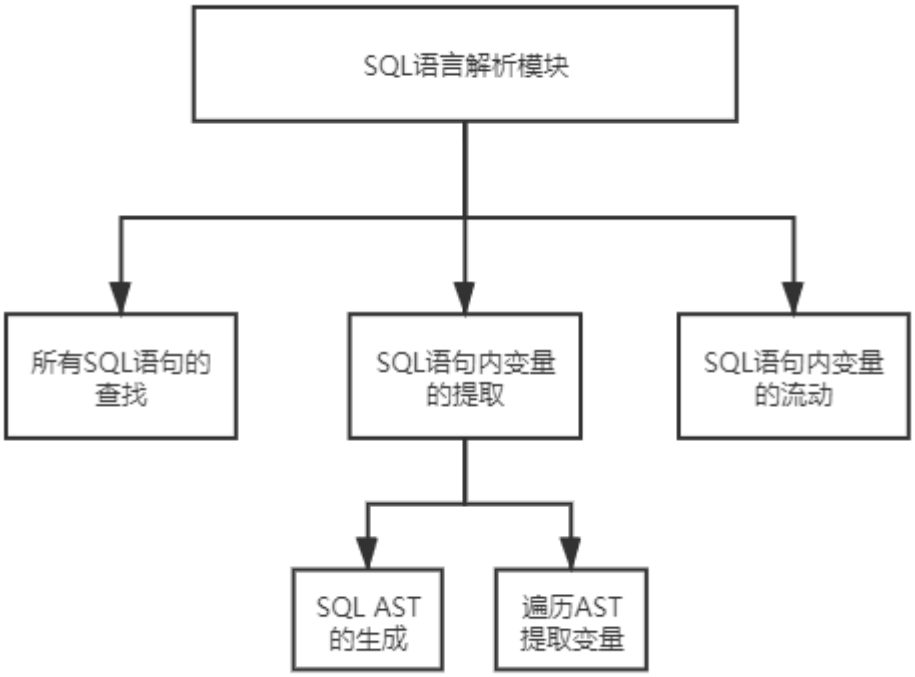
（3）其中 analyseGlobalVariable() 函数为分析超全局变量的流向，更新变量的控制流图，其代码如下：

```
function analyseGlobalVariable($name, $dim, $dir, $var_list){
    $left_index = findIndex($name.$dim, $dir, $var_list);
    $list = searchGlobalVariableLocation($dim);
    $var_right_name = $list[0];
    $var_right_dir = $list[1];
    $var_right_index = findIndex($var_right_name, $var_right_dir, $var_list);
```

```
7.      if($var_list[$var_right_index][2] != "" && ($var_list[$var_right_index][
    2][1] == 'w' ||  $var_list[$var_right_index][2][1] == 'd')){
8.          $var_list[$left_index][2] = $var_list[$var_right_index][2];
9.      }elseif($var_list[$var_right_index][2] != "" && $var_list[$var_right_ind
    ex][2][1] == 'p'){
10.         $var_list[$left_index][2] = "php";
11.     }
12. }
```

3.3 SQL 语言解析模块

3.3.1 模块功能描述



本模块主要是用来获取和解析 php 源代码中的 SQL 语句，首先通过 php-parser 解析树的解析结果获取 sql 语句的信息，然后再利用 sql-parser 解析获取的 sql 语句，将 sql 语句中的关键信息（如变量名、表项名、数据库名、变量流动）提取和整理，保存在数组中，供其他 php 分析模块使用。

3.3.2 Sql 语句的获取

Sql 语句的获取主要通过开源的 sql-parser 来实现, sql-parser 会根据 php 源码生成 php 解析树，并根据解析的结果为每一个元素定义类型。在这里，sql 语句主体的类型为 **EncapsedStringPart**，而 sql 语句中的变量类型为 **Variable**，这些变量类型都是 sql-parser

中自己定义的。通过 Visitor 访问各个节点，并判断当前访问的节点是否为该类型的数据，就可以将 php 源代码中的 sql 语句进行识别和拼接，方便我们的分析。

具体代码如下所示：

```
if ($node instanceof Encapsed)
    {
        $sql_array = $node->parts;
        $sql_string = "";
        foreach ($sql_array as $sql){
            if ($sql instanceof EncapsedStringPart){
                $sql_string = $sql_string . $sql->value;
            }
            if ($sql instanceof Variable){
                $sql_string = $sql_string . $sql->name;
            }
        }
        echo $sql_string . "\n";
```

3.3.3 Sql 语句的解析

对于每一个获得的 sql 语句，我们都可以将其放入开源库 sql-parser 中进行解析。我们首先定义一个 sql-parser 的解析器，然后对我们刚刚得到的 sql 语句进行解析，并输出解析树验证正确性，同时分析 sql 语句的解析结果，从而可以对每一个我们需要的数据进行分析。

由于 sql 语句有不同的类型，每种类型的 sql 语句有着不同结构，因此对于不用类型的 sql 语句有着不同得解析规则，但是解析得结果都是统一的，即获取 sql 语句中得变量名、表项名、数据库名、并分析变量的流动方向，然后将这些信息存入 variable_list 数组中。

具体实现过程如下所示：

```
echo $sql_string . "\n";
$parser = new PHPSQLParser($sql_string, true);
$sql_ast = $parser->parsed;
print_r($sql_ast);
$table_name = "";
$attributes_list = array();
$var_list = array();
if (strtolower(array_key_first($sql_ast)) == "select"){
    foreach ($sql_ast as $item=>$value){
        if ($item == "SELECT"){
            continue;
        }
        elseif($item == "FROM"){
            if ($value[0]["expr_type"] == "table")
                $table_name = $value[0]["table"];
        }
```

```
        elseif($item == "WHERE"){
            foreach ($value as $part){
                if ($part["expr_type"] == "colref")
                    array_push($attributes_list, $part["base_expr"]);
                elseif ($part["expr_type"] == "const"){
                    $tmp_expr = str_replace("%","",$part["base_expr"]);
                    array_push($var_list, $tmp_expr);
                }
            }
        }
    }
    $this->sql_variable =
        addToSqlVariable($this->sql_variable, $var_list, $table_name, $attri
butes_list, "from_db");
}
elseif (strtolower(array_key_first($sql_ast)) == "insert"){
    foreach ($sql_ast as $item => $value){
        if ($item == "INSERT"){
            foreach ($value as $part){
                if ($part["expr_type"] == "table")
                    $table_name = $part["table"];
                else if ($part["expr_type"] == "column-list"){
                    foreach ($part["sub_tree"] as $attr){
                        if ($attr["expr_type"] == "colref")
                            array_push($attributes_list, $attr["base_expr"])
;
                    }
                }
            }
        }
        elseif ($item == "VALUES"){
            foreach ($value as $part){
                if ($part["expr_type"] == "record"){
                    foreach ($part["data"] as $var){
                        if ($var["expr_type"] == "const")
                            array_push($var_list, $var["base_expr"]);
                    }
                }
            }
        }
    }
    $this->sql_variable =
        addToSqlVariable($this->sql_variable, $var_list, $table_name, $attri
butes_list, "to_db");
```

```
}
elseif (strtolower(array_key_first($sql_ast)) == "update"){
    foreach ($sql_ast as $item=>$value){
        if ($item == "UPDATE"){
            foreach ($value as $part){
                if ($part["expr_type"] == "table")
                    $table_name = $part["table"];
            }
        }
        elseif ($item == "SET"){
            foreach ($value as $set_part){
                if ($set_part["expr_type"] == "expression"){
                    foreach ($set_part["sub_tree"] as $part){
                        if ($part["expr_type"] == "colref")
                            array_push($attributes_list, $part["base_expr"])
;
                        elseif ($part["expr_type"] == "const")
                            array_push($var_list, $part["base_expr"]);
                    }
                }
            }
            $this->sql_variable =
                addToSqlVariable($this->sql_variable, $var_list, $table_name
, $attributes_list, "to_db");
            $var_list = array();
            $attributes_list = array();
        }
        elseif ($item == "WHERE"){
            foreach ($value as $where_part){
                if ($where_part["expr_type"] == "colref")
                    array_push($attributes_list, $where_part["base_expr"]);

                if ($where_part["expr_type"] == "const"){
                    $temp_expr = str_replace("%","", $where_part["base_expr"
]);
                    array_push($var_list, $temp_expr);
                }
            }
            $this->sql_variable =
                addToSqlVariable($this->sql_variable, $var_list, $table_name
, $attributes_list, "from_db");
        }
    }
}
```

```
97. echo "\n";
98. print_r($this->sql_variable);
99. echo "\n";
100. print_r($this->sql_variable);
```

3.3.3 输出示例

对于如下的 SQL 语句，输出的语法分析树如下图所示：

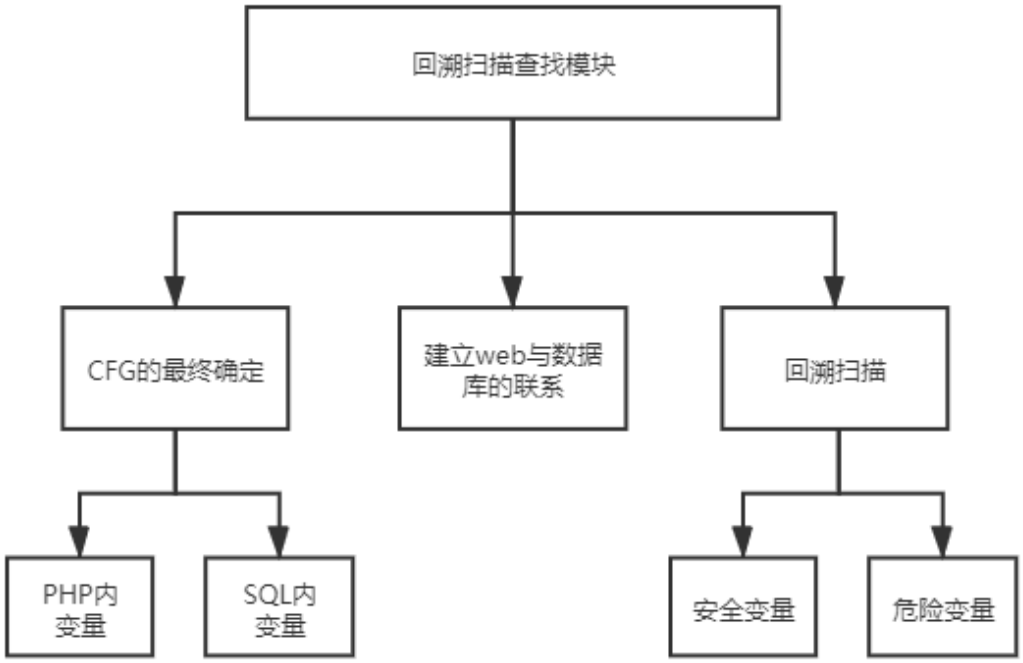
```
SELECT DISTINCT * FROM user WHERE username = 'username'

object(PhpMyAdmin\SqlParser\Statements\SelectStatement)#1284 (17) {
  ["expr"]=>
  array(1) {
    [0]=>
    object(PhpMyAdmin\SqlParser\Components\Expression)#1286 (7) {
      ["database"]=>
      NULL
      ["table"]=>
      NULL
      ["column"]=>
      NULL
      ["expr"]=>
      string(1) "*"
      ["alias"]=>
      NULL
      ["function"]=>
      NULL
      ["subquery"]=>
      NULL
    }
  }
```

```
["from"]=>
array(1) {
  [0]=>
  object(PhpMyAdmin\SqlParser\Components\Expression)#1287 (7) {
    ["database"]=>
    NULL
    ["table"]=>
    NULL
    ["column"]=>
    NULL
    ["expr"]=>
    string(4) "user"
    ["alias"]=>
    NULL
    ["function"]=>
    NULL
    ["subquery"]=>
    NULL
  }
}
```

```
["where"]=>
array(1) {
  [0]=>
  object(PhpMyAdmin\SqlParser\Components\Condition)#1288 (3) {
    ["identifiers"]=>
    array(1) {
      [0]=>
      string(8) "username"
    }
    ["isOperator"]=>
    bool(false)
    ["expr"]=>
    string(21) "username = 'username'"
```

3.4 回溯扫描查找模块

3.4.1 模块功能描述



本模块的主要功能为：建立 PHP 和 SQL 的变量关系，生成最终的变量控制流图，并回溯扫描将变量分类，分成安全变量和危险变量，并记录二阶 SQL 注入漏洞的注入点和触发点，作为静态扫描的结果。

3.4.2 CFG 的最终确定

在遍历每一条 SQL 语句的时候，通过 SQL-Parser 以及上一个模块生成了变量在数据库内的流动关系，更新变量的控制流图，算法较为简单，代码如下：

```
1.  // 对每一条 SQL 语句进行分析并调用下面的函数更新 CFG
2.  foreach ($file_name_list as $item=>$value){
3.      $traverser3 = new NodeTraverser();
4.      $sql_visitor = new SqlVisitor();
5.      $traverser3->addVisitor($sql_visitor);
6.      $modifiedStmts = $traverser3->traverse($ast_list[$item]);
7.      $sql_variable = $sql_visitor->getSqlVariable();
8.      improveDatabaseVariable($value, $sql_variable, $variable_list);
9.      // print_r($sql_variable);
10. }
11.
```

```php
// 更新 CFG
function improveDatabaseVariable($file_name, $sql_var_list, &$variable_list)
{
    if(count($sql_var_list) != 0){
        foreach ($sql_var_list as $value){

            $var_name = str_replace("%", "", substr($value[0], 1, strlen($va
lue[0])-2));

            $index = findIndex($var_name, $file_name, $variable_list);
            if($value[3] == "from_db"){
                  $variable_list[$index][4] = "db(".$value[1]."(".$value[2].
"))";
            }elseif($value[3] == "to_db"){
                $variable_list[$index][4] = "db(".$value[1]."(".$value[2].")
)";
            }
        }
    }
}
```

3.4.3 回溯，确定去向

知道了来源，回溯即可得到每个变量的去向，这是一个将单项链表变为双向链表的过程，代码如下：

```php
//回溯，确定去向
$flag = true;
while($flag){
    $flag = false;
    foreach ($variable_list as $k => $v){
        if($v[4] != ""){
            foreach ($v[3] as $item){
                if($variable_list[$item][4] == ""){
                    $variable_list[$item][4] = $variable_list[$k][4];
                    $flag = true;
                }
            }
        }
    }
}
// 空的填充 php
foreach ($variable_list as $value){
    if($value[2] == ""){
```

```
        $value[2] = "php";
    }
    if($value[4] == ""){
        $value[4] = "php";
    }
}
```

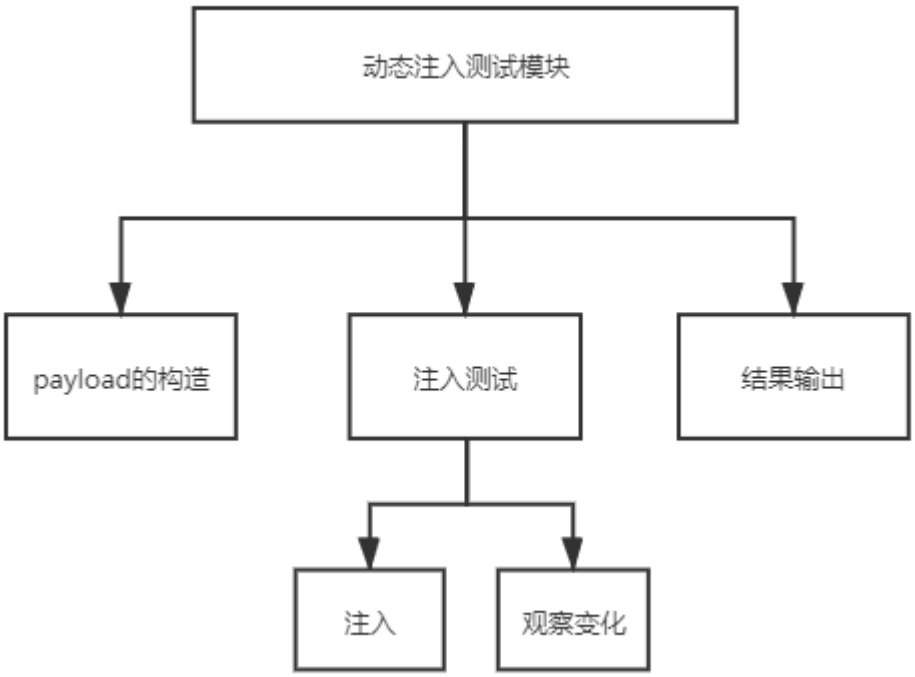
3.4.4 查找注入点和触发点

根据二阶 SQL 注入产生的条件，如果有变量来源于 web，在某个地方进入了数据库的某个 table 里，而在另一处又从数据库内拿出来，并最终进入某条 SQL 语句执行，并对其他变量产生的影响，那么它就存在二阶 SQL 注入的风险。代码如下：

```
//比较，输出结果
$count = 0;             //注入点数
$result = array();      //注入点、触发点等信息
foreach ($variable_list as $sk=>$sv){
    if($sv[2] != "" && $sv[4]!="" && $sv[2][0] == 'd' && $sv[4][0] == 'd'){

        foreach ($variable_list as $fk=>$fv){
            if($fv[2] != "" && $fv[4]!="" && $fv[2][0] == 'w' && $fv[4][0] == 'd' && $fv[2] != "web"){
                if($fv[4] != $sv[2]){ continue; }
                $flag = true;
                foreach ($result as $value){
                    if($value[0] == $fv[1] && $value[1] == $fv[2] && $value[2] == $sv[1] && $value[3] == $sv[2]){
                        $flag = false;
                        break;
                    }
                }
                if($flag){
                    $result[$count] = [$fv[1], $fv[2], $sv[1], $sv[2]];
                    $count++;
                }
            }
        }
    }
}
```

3.5 动态注入测试模块

3.5.1 模块功能描述



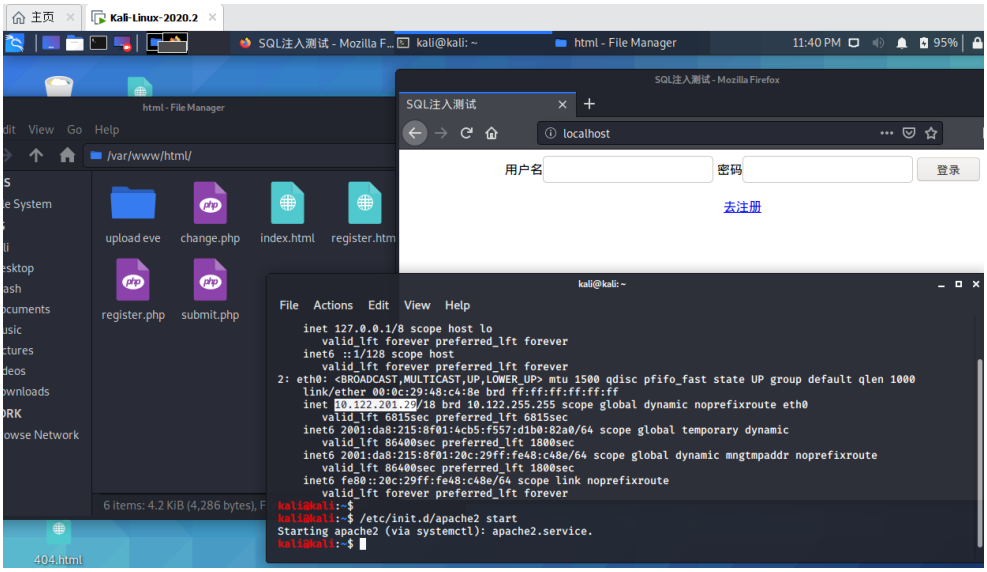
本模块的主要功能为：由于静态的扫描误报率很高，所以动态的去验证是必要的。通过构造 payload，进行数据库的注入，并观察网站和数据库内的变化，如果构造的 payload 与正常的数据产生的变化不同，那么就存在二阶 SQL 注入漏洞。并将结果保存，输入。

3.5.2 网站注入示例

二阶 SQL 注入的 payload 构造与一阶 SQL 盲注不一样，这里以我们搭建的网站 demo 进行说明其过程：

3.5.2.1 漏洞分析

服务器及运行效果如下：



网站过滤了一阶 SQL 注入漏洞，即使用 PHP 安全函数进行了字符的转义再执行 SQL 语句，但是存储在数据库内还是原数据，再次从数据库拿出来的时候没有进行转义，所以存在二阶 SQL 注入漏洞。

3.5.2.2 利用过程

这种二阶 SQL 注入漏洞的检测构造成功的 payload 以及操作过程如下：

- 对注入点构造 http 请求，模拟用户输入（test）
- 类似 1，模拟输入（test'#）
- 寻找触发点，构造 http 请求
- 修改攻击（test'#）相关的信息、
- 查看（test）相关的信息是否被修改

3.5.2.3 算法实现

由于可以从源码扫描中知道数据库内表格的信息，所以这里直接进行第一次正常操作如下和第二次的注入操作，将 test 和 test'#写入数据库。

```
1.  $regist_url = "http://localhost:63342/SQL 注入环境
    /register.php?_ijt=m255cl3hk9n8behc1n0p4f9fg";
2.  $cookie = dirname(__FILE__) . '/cookie_ydma.txt'; //设置 cookie 保存的路径
3.  $regist = array(
4.      'username' => 'test3',
5.      'password' => 123456
6.  );
7.
8.  $regist1 = array(
9.      'username' => 'test3#',
10.     'password' => 123456
11. );
```

```
$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, $regist_url);
curl_setopt($ch, CURLOPT_HEADER, 0); //是否显示头信息
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 0); //是否自动显示返回的信息
//curl_setopt($ch, CURLOPT_COOKIEJAR, $cookie); //设置 cookie 信息保存在指定的文件夹中
curl_setopt($ch, CURLOPT_POST, 1); //以 POST 方式提交
curl_setopt($ch, CURLOPT_POSTFIELDS, $regist);//要执行的信息
curl_exec($ch); //执行 CURL
curl_close($ch);

$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, $regist_url);
curl_setopt($ch, CURLOPT_HEADER, 0); //是否显示头信息
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 0); //是否自动显示返回的信息
//curl_setopt($ch, CURLOPT_COOKIEJAR, $cookie); //设置 cookie 信息保存在指定的文件夹中
curl_setopt($ch, CURLOPT_POST, 1); //以 POST 方式提交
curl_setopt($ch, CURLOPT_POSTFIELDS, $regist1);//要执行的信息
curl_exec($ch); //执行 CURL
curl_close($ch);
```

接下来是触发点（本 demo 在修改密码界面）：模拟用户登录后修改密码的过程如下：

```
$login_url = "http://localhost:63342/SQL%E6%B3%A8%E5%85%A5%E7%8E%AF%E5%A2%83/submit.php";
$cookie = dirname(__FILE__) . '/cookie_ydma.txt'; //设置 cookie 保存的路径
$first_login = array(
    'username' => "test",
    'password' => 123456
);

$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, $login_url); //登录提交的地址
curl_setopt($ch, CURLOPT_HEADER, 0); //是否显示头信息
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 0); //是否自动显示返回的信息
curl_setopt($ch, CURLOPT_COOKIEJAR, $cookie); //设置 cookie 信息保存在指定的文件夹中
curl_setopt($ch, CURLOPT_POST, 1); //以 POST 方式提交
curl_setopt($ch, CURLOPT_POSTFIELDS, $first_login);//要执行的信息
curl_exec($ch); //执行 CURL
curl_close($ch);

$cookie = getCookie('test\'#');
```

```php
$change_url = "localhost:63342/SQL%E6%B3%A8%E5%85%A5%E7%8E%AF%E5%A2%83/change.php";
$change = array(
    'old_password' => '123456',
    'new_password' => '123456change'
);

$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, $change_url);
curl_setopt($ch, CURLOPT_HEADER, 0); //是否显示头信息
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 0); //是否自动显示返回的信息
curl_setopt($ch, CURLOPT_COOKIEFILE, $cookie); //设置 cookie 信息保存在指定的文件夹中
curl_setopt($ch, CURLOPT_POST, 1); //以 POST 方式提交
curl_setopt($ch, CURLOPT_POSTFIELDS, $change);//要执行的信息
curl_exec($ch); //执行 CURL
curl_close($ch);
```

3.5.3 输出

采用命令行与用户交互，代码如下：

```php
// 输出 variable_list 内容的函数
function echoScanArray($array){
    foreach ($array as $k => $v) {
        echo $k.": ".$v[0]." ".$v[1]." ".$v[2]." [";
        foreach ($v[3] as $vv){
            echo $vv.", ";
        }
        echo "] ".$v[4]." [";
        foreach ($v[5] as $vv){
            echo $vv.", ";
        }
        echo "] ".$v[6]."\n";
    }
}

// 用户与控制台的交互
while (true){
    echo "[cmd]\n[1]输出扫描结果\n[2]输出变量流\n[3]退出系统\n";
    $cmd = fgets(STDIN);
    if($cmd[0] == '1'){
        echo "\n".$count." result(s) found\n";
        if ($count != 0){
```

```
            for($i = 0; $i < $count; $i++){
                for($j = 0; $j < 4; $j++){
                    echo $result[$i][$j];
                    if ($j < 3){
                        echo " -> ";
                    }
                }
                echo "\n";
            }
        }
    }elseif ($cmd[0] == '2'){
        //输出结果
        echo "\nVariable Info:\n".count($variable_list)."\n";
        echoScanArray($variable_list);
    }elseif ($cmd[0] == '3'){
        echo "\n 欢迎下次使用，bye-bye! ";
        exit(0);
    }
}
```

输出的示例如下：

```
    请将php项目放入桌面
请输入项目名称：demo

正在进行扫描项目demo...

Scan Files:
change.php
register.php
submit.php

Variable Count:
29

[cmd]
[1]输出扫描结果
[2]输出变量流
[3]退出系统
2

Variable Info:
29
0: username change.php db(user(username)) [1, ] db(user(username)) [] 0
1: _SESSION change.php db [] db(user(username)) [0, ] 0
2: _POST change.php web [] db(user(password)) [4, 5, ] 0
3: conn change.php php [] db(user(password)) [6, 7, 9, ] 0
4: old_pass change.php web_form(old_password) [2, ] db(user(password)) [6, ] 0
5: new_pass change.php web_form(new_password) [2, ]  [7, ] 0
6: old_password change.php web_form(old_password) [3, 4, ] db(user(password)) [] 0
7: new_password change.php web_form(new_password) [3, 5, ]  [] 0
8: sql change.php php []  [9, ] 0
9: result change.php  [3, 8, ]  [] 0
10: conn register.php php [] db(user(username)) [14, 15, 17, ] 0
11: name register.php web_form(username) [12, ] db(user(username)) [14, ] 0
12: _POST register.php web [] db(user(username)) [11, 13, ] 0
13: pass register.php web_form(password) [12, ]  [15, ] 0
14: username register.php web_form(username) [10, 11, ] db(user(username)) [] 0
15: password register.php web_form(password) [10, 13, ]  [] 0
16: sql register.php php []  [17, ] 0
```

```
[cmd]
[1]输出扫描结果
[2]输出变量流
[3]退出系统
1

1 result(s) found
register.php -> web_form(username) -> change.php -> db(user(username))
```