**PHP代码审计流程分析法**

1. 思路
2. 分析每一个函数。
3. 标记出危险的函数。
4. 通过手工输入，通过程序梳理全部的流程。
5. 人工判断每个流程是否危险。
6. 数据库设计
   1. del\_paths

|  |  |  |  |
| --- | --- | --- | --- |
| **列名** | **类型** | **长度** | **备注** |
| id | int |  | 自增长 |
| md5 | text |  | 路径的md5值 |
| points\_id | text |  | 点的ID |
| check | int |  |  |
| start\_file | text |  | 路径开始的文件 |
| start\_line | int |  | 路径开始的行 |
| end\_file | text |  | 路径结束的文件 |
| end\_line | int |  | 路径结束的行 |
| remark | text |  | 备注 |

* 1. files

|  |  |  |  |
| --- | --- | --- | --- |
| **列名** | **类型** | **长度** | **备注** |
| id | int |  | 自增长 |
| file | text |  | 文件名 |
| edit | int |  | 是否继续编辑，0表示编辑，1表示不编辑。目前没有启用。 |

* 1. functions

|  |  |  |  |
| --- | --- | --- | --- |
| **列名** | **类型** | **长度** | **备注** |
| id | int |  | 自增长 |
| file | text |  | 函数所在的文件名 |
| file\_id | int |  | 函数所在的文件ID |
| name | text |  | 函数名及其参数，自动提取，为函数第一行。 |
| points | text |  | 函数所包含的点的ID，使用#号分隔 |
| remark | text |  | 备注 |
| paths | text |  | 路径数组，数组为二级数组，每一个一级数组为一条路径，二级数组中有两个元素，path元素包含路径所包含点的ID数组，end为此路径是否结束的判断true为路径结束，false未结束。 |

* 1. lines

|  |  |  |  |
| --- | --- | --- | --- |
| **列名** | **类型** | **长度** | **备注** |
| id | int |  | 自增长 |
| point\_u\_id | int |  | 函数ID，没有为null |
| point\_v\_id | int |  | 函数ID，没有为null |
| variables\_ids | text |  | 这条路径所需要variables的ID，使用  Addslashes serialize |

* 1. paths

|  |  |  |  |
| --- | --- | --- | --- |
| **列名** | **类型** | **长度** | **备注** |
| id | int |  | 自增长 |
| md5 | text |  | 路径的md5值 |
| points\_id | text |  | 点的ID数组 |
| lines\_id | text |  | 线的ID数组 |
| check | int |  |  |
| start\_file | text |  | 路径开始的文件 |
| start\_line | int |  | 路径开始的行 |
| end\_file | text |  | 路径结束的文件 |
| end\_line | int |  | 路径结束的行 |
| remark | text |  | 备注 |

* 1. points

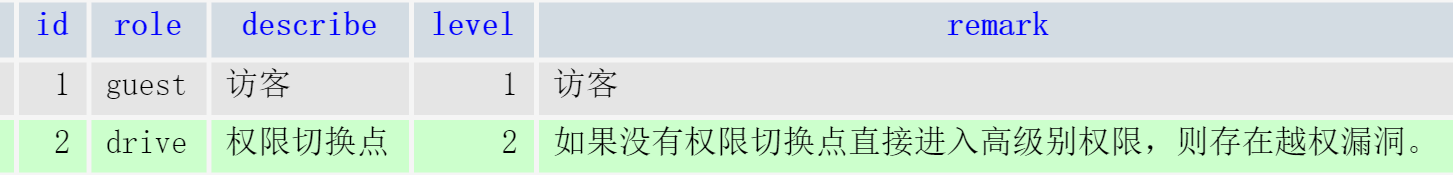
|  |  |  |  |
| --- | --- | --- | --- |
| **列名** | **类型** | **长度** | **备注** |
| id | int |  | 自增长 |
| file | text |  | 路径和文件名 |
| file\_id | int |  | 文件ID |
| line\_number | int |  | 行号 |
| level | int |  | 是否危险，1代表危险，0代表不危险。-2代表不判断，-3代表未判断。 |
| position | int |  | 是否是入口行，1代表是，0代表不是,-1代表结束行。-2代表不判断，-3代表未判断。 |
| content | text |  | 行的内容 |
| role\_id | int |  | 对应role表的ID |
| db\_action | text |  | 对数据库的操作in表示数据插入，out表示数据取出。 |
| db\_line | text |  | 关联的表，使用&分隔。 |
| remark | text |  | 备注 |
| is\_function | int |  | 这个点是否是函数 |
| function\_line | text |  | 这个点关联了的函数ID，数组使用serialize之后存储。 |

* 1. role

角色中一定要有两个权限设置。

guest ：访客

drive ：权限切换点，如果没有权限切换点直接进入高级别权限，则存在越权漏洞。



|  |  |  |  |
| --- | --- | --- | --- |
| **列名** | **类型** | **长度** | **备注** |
| id | int |  | 自增长 |
| role | text |  | 角色名 |
| descibe | text |  | 角色描述 |
| level | int |  | 级别从小到大越来越高 |
| remark | text |  | 备注 |

* 1. variables

|  |  |  |  |
| --- | --- | --- | --- |
| **列名** | **类型** | **长度** | **备注** |
| id | int |  | 自增长 |
| variable | text |  | 变量名 |
| value | text |  | 变量值 |
| use | int |  | 是否已经用过，0代表未用过，1代表已用过。 |
| from | int |  | 变量来源，0代表用户，1代表数据库 |

1. 详细设计
   1. 流程设计
      1. 路径计算算法（完成，未测试）
2. 找出所有的起点，start数组。
3. 从start里提取第一个数组。
4. 判断数组是否调用函数，调用函数则遍历函数的所有路径，添加到全部路径中。
5. 数组所有的线。没有线返回。
6. 提取一根线，以终点为开始。
7. 判断数组是否调用函数，调用函数则遍历函数的所有路径，添加到全部路径中。
8. 如果为终点为结束点，结束。否则回到2。
   * 1. 源代码基础信息输入（ok）
9. 输入文件路径，选择文件，上传。
10. 输入点的数据：文件路径、行号（从0开始）、级别（自动判断，默认为null）、位置（是否为入口或结束，默认为NULL）、行的内容。
11. 插入数据库。
    * 1. 代码合并（ok）
12. 点击与上一行合并，当前代码行与上一行合并。
13. 将当前代码行标记为不判断。。
    * 1. 源代码基础信息完善（）
14. 选择路径、文件。
15. 列出文件未完善基础信息的行。（级别和位置为NULL值的）
16. 添加变量。（添加到temp表中）
17. 选择变量，输入线。（1、需要校验线的终点是否已经添加。2、变量从temp表中取）
18. 输入：级别、位置。
    * 1. 单独添加线
19. 选择目录、选择文件；
20. 选择线头；
21. 输入线尾所在文件和行；或输入线尾所在文件和函数名，有类名可以输入类名
22. 输入说明(备注)，这条线可行吗？。
23. 选择变量（这个一定要写清楚。）；
24. 插入数据库。
    * 1. 二次注入判断（需更改）

未考虑函数的情况。

* + 1. 其它说明

1、是否危险的判断标准：

* 1. 测试用例

测试用例需增加函数的部分。