Second Meeting on PHP DoS Project

22th Auguest

Summary of Problems

1. Constraints are full of built-in functions
2. Method Calls of user defined classes in OOP
3. “Constraint explosion” problem

Constraints are full of built-in functions

example of a constraint snippet

True && !is_dir($files . '/' . $file) && is_file($files . '/' . $file) || True && !is_dir($files . '/' . $file) && !is_file($files . '/' . $file)) && ($file != '.' && $file != '..') || True && !($file != '.' && $file != '..')) && scandir($files) && is_array(scandir($files)) && is_dir($files) && isset($_GET['img']) && isset($_GET['logout']) || (((True && !$this->addDir($files . '/' . $file) || True && is_dir($files . '/' . $file) || True && !is_dir($files . '/' . $file) && is_file($files . '/ ' . $file) || True && !is_dir($files . '/' . $file) && !is_file($files . '/' . $file)) && ($file != '.' && $file != '..') || True && !($file != '.' && $file != '..')) && scandir($files) && is_array(scandir($files)) && is_dir($files) && isset($_GET['img']) && !isset($_GET['logout'])) && (isset($_SESSION[FM_SESSION_ID]['logged'])

True || True && ($file != '.' && $file != '..') || True && !($file != '.' && $file != '..')) || True|| True || True || True && ($file != '.' && $file != '..') || True

True && len($file) > 0 || True && ($file != '.' && $file != '..') || True && !($file != '.' && $file != '..')) || True|| True || True || True && ($file != '.' && $file != '..') || True

Method Calls of user defined classes in OOP

We won't know the object type until we walk to its definition

```
$x = new Obj();
…...
if ($x->method() > 1){
        ……..
        # tainted loop here;
}
```

Change object property value inside object method calls

```
public function f(){
        $this->x ++;
        return $this->y;
}
```

“Constraint explosion” problem

1. Constraints - Return value pair built for each function, in short, c-v pair
2. Each time the return value is used, constraints will duplicate into num(c-v pair) copies
3. Constraints accumulate throughout function calls

e.g. function f has **n1** c-v pairs, g has **n2** c-v pairs, so function h has n1 * n2 c-v pairs

```
function h(){
        return f() + g();
}
```

```
if (h() > 0){
        ……
        #tainted loop here;
}
```