PHP反序列化与Wordpress一些意外Bug的有趣结合

5ecu**** / 2018-03-13 14:34:56 / 浏览数 12270 安全技术 技术讨论 顶(0) 踩(0)

译者：kyhvedn@5ecurity.cn
原作者：Nicky Bloor@nickstadb
原文地址：https://nickbloor.co.uk/2018/02/28/popping-wordpress/

译者按：作者证明这是最新的WordPress
4.9.4版本存在的问题，默认安装，虽然bug有些不值一提，但会导致RCE，作者希望能够多多讨论这些事情的思路和努力，而不是过多地关注漏洞详细信息。
PoC视频URL：https://videopress.com/v/tYSsU4Ch

0x00 前序

几个月前，我正在编写一篇关于PHP反序列化漏洞的博客文章，决定为这篇文章找一个真实目标，能够让我将测试数据传输给PHP unserialize
()函数来实现演示目的。于是我下载了一批WordPress插件，并开始通过grepping来寻找调用unserialize ()的代码实例：

```
$url = 'http://api.wordpress.org/plugins/info/1.0/';
$response = wp_remote_post ($url, array ('body' => $request));
$plugin_info = @unserialize ($response ['body']);
if (isset ($plugin_info->ratings)) {
```

这个插件的问题在于发送明文HTTP请求，并且将该请求响应传递给了unserialize()函数。就真实攻击而言，它并不是最佳入口点，但是如果我能通过这种微不足道的方式向u

0x01 PHP反序列化攻击

简单来说，当攻击者能够将他的数据提供给应用程序，而该应用程序将数据转化为运行对象时没有作适当验证的时候就会出现反序列化漏洞。如果攻击者数据被允许去控制这
Java Deserialization ，其中的一般概念适用于任何基础技术。

在PHP应用程序的现状来看，POP小工具最为人熟知和最可靠的原因在于类的__wakeup()方法（*PHP“魔术方法”，unserialize()函数会检查是否存在__wakeup()，如

除了__wakeup()和__destruct()方法之外，
PHP还有其他“魔术方法”，可以在类中定义，也可以在反序列化之后调用，这取决于反序列化对象的使用方式。在一个更大更复杂的应用程序中可能很难追踪到反序列化对象

0x02 通用的PHP POP小工具

为了简化这个过程，我编写了一个PHP类，它定义了所有魔术方法并且在调用任何魔术方法时将详细信息写入日志文件。特别有趣的是魔术方法__get()和__call()，如果
object上设置的属性，以便操纵并使用这些属性的代码，而后者可以用来识别POP小工具触发使用的非魔术方法（并且可以将它们自身用作POP小工具）。
该类的__wakeup()方法还使用了get_declared_classes()函数来检索和记录可以利用exploit
payload的已声明类的列表（虽然这不会反映当前未声明但可以自动加载的类）。

```
<?php
if(!class_exists("UniversalPOPGadget")) {
class UniversalPOPGadget {
   private function logEvent($event) {
       file_put_contents('UniversalPOPGadget.txt', $event . "\r\n", FILE_APPEND);
   }

   public function __construct() { $this->logEvent('UniversalPOPGadget::__construct()'); }
   public function __destruct() { $this->logEvent('UniversalPOPGadget::__destruct()'); }
   public function __call($name, $args) {
       $this->logEvent('UniversalPOPGadget::__call(' . $name . ', ' . implode(',', $args) . ')');
   }
   public static function __callStatic($name, $args) {
       $this->logEvent('UniversalPOPGadget::__callStatic(' . $name . ', ' . implode(',', $args) . ')');
   }
   public function __get($name) { $this->logEvent('UniversalPOPGadget::__get(' . $name . ')'); }
   public function __set($name, $value) { $this->logEvent('UniversalPOPGadget::__set(' . $name . ', ' . $value . ')'); }
   public function __isset($name) { $this->logEvent('UniversalPOPGadget::__isset(' . $name . ')'); }
   public function __unset($name) { $this->logEvent('UniversalPOPGadget::__unset(' . $name . ')'); }
   public function __sleep() { $this->logEvent('UniversalPOPGadget::__sleep()'); return array(); }
   public function __wakeup() {
       $this->logEvent('UniversalPOPGadget::__wakeup()');
       $this->logEvent("  [!] Defined classes:");
       foreach(get_declared_classes() as $c) {
           $this->logEvent("    [+] " . $c);
```

```
        }
    }
    public function __toString() { $this->logEvent('UniversalPOPGadget::__toString()'); }
    public function __invoke($param) { $this->logEvent('UniversalPOPGadget::__invoke(' . $param . ')'); }
    public function __set_state($properties) {
        $this->logEvent('UniversalPOPGadget::__set_state(' . implode(',', $properties) . ')');
    }
    public function __clone() { $this->logEvent('UniversalPOPGadget::__clone()'); }
    public function __debugInfo() { $this->logEvent('UniversalPOPGadget::__debugInfo()'); }
}}
?>
```

0x03 PHP检测

将上面的代码保存到一个PHP文件中，我们可以通过这个在其他任何PHP脚本中插入一个include'/path/to/UniversalPOPGadget.php'语句，并使这个类可用。以下

```
import os
import sys

#Set this to the absolute path to the file containing the UniversalPOPGadget class
GADGET_PATH = "/path/to/UniversalPOPGadget.php"

#File extensions to instrument
FILE_EXTENSIONS = [".php", ".php3", ".php4", ".php5", ".phtml", ".inc"]

#Check command line args
if len(sys.argv) != 2:
 print "Usage: GadgetInjector.py <path>"
 print ""
 sys.exit()

#Search the given path for PHP files and modify them to include the universal POP gadget
for root, dirs, files in os.walk(sys.argv[1]):
 for filename in files:
   for ext in FILE_EXTENSIONS:
     if filename.lower().endswith(ext):
       #Instrument the file and stop checking file extensions
       fIn = open(os.path.join(root, filename), "rb")
       phpCode = fIn.read()
       fIn.close()
       fOut = open(os.path.join(root, filename), "wb")
       fOut.write("<?php include '" + GADGET_PATH + "'; ?>" + phpCode)
       fOut.close()
       break
```

0x04 分析反序列化入口点

回到刚刚那个调用unserialize()函数的WordPress插件代码片段，我不知道该如何去实际触发unserialize()函数的调用，我所知道的是这个插件应该向http://api.wordpr
payload：

```
<?php
   include('UniversalPOPGadget.php');
   print serialize(new UniversalPOPGadget());
```

在使用这种手段后，我开始像往常一样使用WordPress实例，特别注意了与目标WordPress插件相关的所有功能，同时查看UniversalPOPGadget日志文件。很快地，生成了

```
UniversalPOPGadget::__wakeup()
 [!] Defined classes:
 [...Snipped...]
UniversalPOPGadget::__get(sections)
UniversalPOPGadget::__isset(version)
UniversalPOPGadget::__isset(author)
UniversalPOPGadget::__isset(requires)
UniversalPOPGadget::__isset(tested)
UniversalPOPGadget::__isset(homepage)
UniversalPOPGadget::__isset(downloaded)
UniversalPOPGadget::__isset(slug)
UniversalPOPGadget::__get(sections)
```

```
UniversalPOPGadget::__get(sections)
UniversalPOPGadget::__isset(banners)
UniversalPOPGadget::__get(name)
UniversalPOPGadget::__get(sections)
UniversalPOPGadget::__isset(version)
UniversalPOPGadget::__isset(author)
UniversalPOPGadget::__isset(last_updated)
UniversalPOPGadget::__isset(requires)
UniversalPOPGadget::__isset(tested)
UniversalPOPGadget::__isset(active_installs)
UniversalPOPGadget::__isset(slug)
UniversalPOPGadget::__isset(homepage)
UniversalPOPGadget::__isset(donate_link)
UniversalPOPGadget::__isset(rating)
UniversalPOPGadget::__isset(ratings)
UniversalPOPGadget::__isset(contributors)
UniversalPOPGadget::__isset(tested)
UniversalPOPGadget::__isset(requires)
UniversalPOPGadget::__get(sections)
UniversalPOPGadget::__isset(download_link)
```

日志文件中显示，在UniversalPOPGadget对象被反序列化之后，用程序试图获取或检查是否存在多个属性（段、版本、作者等等）。首先这就告诉我们，通过这个特定的入

0x05 Sections属性？

上面的日志文件显示，与反序列化对象的首次交互是尝试获取名为`sections`的属性。

```
$url = 'http://api.wordpress.org/plugins/info/1.0/';
   $response = wp_remote_post ($url, array ('body' => $request));
   $plugin_info = @unserialize ($response ['body']);
   if (isset ($plugin_info->ratings)) {
```

现在来看最初的目标插件，它在调用unserialize()之后做的第一件事是检查名为`rating`的属性是否存在，那么这个日志并不是我当初注意的第三方插件产生的！

0x06 POPping WordPress出现的意外

对WordPress代码进行一次快速grep，对于上面提到的HTTP
URL，显示该请求是由`wp-admin/includes/plugin-install.php`文件中的WordPress插件API发送的。浏览代码时并不清楚反序列化的payload
object是如何使用的，或者切确地说这个HTTP请求以及随后对unserialize()函数的调用是从哪里触发的。我继续点击WordPress管理界面，发现日志是从■■■■■、■■■■

我记录了一些WordPress发出的HTTP请求并把它们发送到■■■`api.wordpress.org`以获取实例响应，结果响应的是`stdClass`类型的序列化对象，更重要的是示例响应结

```
<?php
   $payloadObject = new stdClass();
   $payloadObject->name = "PluginName";
   $payloadObject->slug = "PluginSlug";
   $payloadObject->version = "PluginVersion";
   print serialize($payloadObject);
```

我开始修改这些对象的属性并刷新相关的WordPress页面，来测试修改内容对结果页面有何影响（如果有的话）。在有些情况下WordPress使用了HTML编码来防止HTML/

在快速尝试一些JavaScript和Python脚本之后我有了假设漏洞的运用证明。这个PoC会导致WordPress管理界面中的"更新和插件"菜单旁显示一个徽章，表示有更新可用（当
payload被注入到该页面中，然后就添加了一个新的管理员账户并将一个基本PHP命令shell注入到现行的WordPress主题的index.php中。

在大多数情况下这种PoC攻击足以实现代码执行，但是我也发现了我可以使用类似方式向WordPress发送一个错误的插件更新来攻击WordPress管理界面的点击更新功能，如

0x07 解答

深入挖掘这一点，我注意到即使没有登录，WordPress也会发送了类似对api.wordpress.org的HTTP请求，我开始对WordPress进行代码审计来了解其中发生了什么，以及

```
function wp_schedule_update_checks() {
   if ( ! wp_next_scheduled( 'wp_version_check' ) && ! wp_installing() )
       wp_schedule_event(time(), 'twicedaily', 'wp_version_check');

   if ( ! wp_next_scheduled( 'wp_update_plugins' ) && ! wp_installing() )
       wp_schedule_event(time(), 'twicedaily', 'wp_update_plugins');

   if ( ! wp_next_scheduled( 'wp_update_themes' ) && ! wp_installing() )
       wp_schedule_event(time(), 'twicedaily', 'wp_update_themes');
```

```
}
```

WordPress会每天两次调用wp_version_check()函数、wp_update_plugins()■■和wp_update_themes()■■。默认情况下，这些更新检查也可以通过wp-cron.p
最终我设法伪造了来自api.wordpress.org的几个响应，来触发对$upgrader->upgrade()的调用，然而以前的伪造插件更新攻击在这里似乎不起作用，之后我在shou

```
/**
    * [...Snipped...]
    *
    * Generally speaking, plugins, themes, and major core versions are not updated
    * by default, while translations and minor and development versions for core
    * are updated by default.
    *
    * [...Snipped...]
    */
```

事实证明这是WordPress试图升级■■Hello
Dolly■■的翻译，我一直试图从downloads.wordpress.org下载hello-dolly-1.6-en_GB.zip，而不是请求我伪造的插件zip文件。我下载了原始文件，添加了一个shell.p

攻击者既然可以对WordPress网站执行MitM■■或DNS■■，那么就可以针对自动更新功能执行零交互攻击，并将恶意脚本写入服务器。当然这不一定是一次简单的攻击，但

WordPress团队意识到这些问题，但是他们的立场似乎是，如果HTTPS启用失败，为了允许在具有旧或损坏的SSL堆栈系统上运行的WordPress网站进行更新，WordPress将

0x08 注意事项/陷阱

当请求更新详细信息和更新存档时，WordPress会尝试首先通过HTTPS连接到api.wordpress.org和downloads.wordpress.org，但是如果由于任何原因导致HTTPS
如果WordPress的PHP脚本属于不同的用户，那么WordPress将默认无法自动更新（因此不容易受到上述攻击），例如index.php为用户foo拥有，但WordPress是在用户w

上一篇：针对Weblogic测试的一些小总结 下一篇：那些年，我们追过的“蓝”

1. 0 条回复
   - 动动手指，沙发就是你的了！