目录

一、题目

1、源码

```php
// composer require "twig/twig"
require 'vendor/autoload.php';
class Template {
    private $twig;

    public function __construct() {
        $indexTemplate = '<img ' .'src="https://loremflickr.com/320/240">' .
            '<a href="{{link|escape}}">Next slide »</a>';
        // Default twig setup, simulate loading
        // index.html file from disk
        $loader = new Twig\Loader\ArrayLoader(['index.html' => $indexTemplate]);
        $this->twig = new Twig\Environment($loader);
    }

    public function getNexSlideUrl() {
        $nextSlide = $_GET['nextSlide'];
        return filter_var($nextSlide, FILTER_VALIDATE_URL);
    }

    public function render() {
        echo $this->twig->render('index.html',['link' => $this->getNexSlideUrl()]);
    }
}

(new Template())->render();
```

2、知识点

知识点	说明
require	引用文件。当出现错误时，生成一个致命错误，在错误发生后脚本停止执行
include	引用文件。当出现错误时，生成一个警告，在错误发生后脚本会继续执行
filter_var	通过指定的过滤器过滤一个变量
=>	定义数组
FILTER_VALIDATE_URL	过滤器把值作为URL验证，判断URL格式是否正确
htmlspecialchars	把预定义字符（& " ' < >）转换为HTML实体

3、解读

1）第25行，将Tmplate类实例化成对象，并调用其render()方法
2）实例化成对象时，会触发6行的__construct构造器，并优先执行其中的代码
3）第7行，使用escape对link变量进行过滤，并将拼接后的HTML代码赋值给变量 $indexTemplate
4）第11行，实例化对象ArrayLoader，并传入二维数组，将$indexTmplate作为index.html的值，最后返回给$loader
5）第12行，实例化对象Enviroment，并传入$loader作为实参
6）第20行，这里开始执行调用 render() 函数了
7）第21行，调用该对象的 getNexSlideUrl()函数，作为link的值，并将整体传给index.html，输出
8）第15行，这里开始执行调用 getNexSlideUrl() 函数
9）第16行，通过 GET方法接收nextSlide变量，并赋值给 $nextSlide
10）第17行，通过 filter_var() 函数，用 FILTER_VALIDATE_URL 判断 $nextSlide是否为一个URL，并返回boolean值

4、分析

1）通过filter_var()和escape对用户的值进行了双层过滤
2）filter_var()中使用的过滤器 FILTER_VALIDATE_URL 用于判断值是否为 URL格式
3）escape是用PHP内置函数 htmlspecialchars 实现的，也就是过滤预定义字符' " & < >
4）此时通过javascript 伪协议传入XSS代码，拼接到HTML代码中，即可实现XSS攻击

5、利用

1）Payload：
?nextSlide=javascript://comment%250aalert(1)

2）说明：
//在javascript中表示单行注释，那么后面的内容都会被注释掉，但是符合URL格式了；
其次使用%250a（经过URL解码后也就是%25，换行符）造成一个换行，那么后面的alert(1)也就属于下一行，不会被//进行注释，并成功在javascript中被执行

二、CMS

1、源码-Anchor 0.9.2

themes\default\404.php

```
<?php theme_include('header'); ?>
    <section class="content wrap">
        <h1>Page not found</h1>
        <p>Unfortunately, the page <code>/<?php echo current_url(); ?></code>
        could not be found. Your best bet is either to try the <a href=
        "<?php echo base_url(); ?>">homepage</a>, try <a href="#search">searching</a>
        ,or go and cry in a corner (although I don't recommend the latter).</p>
    </section>
<?php theme_include('footer'); ?>
```

anchor\functions\helpers.php

```
function current_url() {
        return Uri::current();
}
```

system\uri.php

```
public static function current() {
    if(is_null(static::$current)) static::$current = static::detect();
        return static::$current;
}
```

```
public static function detect() {
    // create a server object from global
    $server = new Server($_SERVER);
    $try = array('REQUEST_URI', 'PATH_INFO', 'ORIG_PATH_INFO');

    foreach($try as $method) {
        // make sure the server var exists and is not empty
        if($server->has($method) and $uri = $server->get($method)) {
            // apply a string filter and make sure we still have somthing left
            if($uri = filter_var($uri, FILTER_SANITIZE_URL)) {

                // make sure the uri is not malformed and return the pathname
                if($uri = parse_url($uri, PHP_URL_PATH)) {
                    return static::format($uri, $server);
                }
                // woah jackie, we found a bad'n
                throw new ErrorException('Malformed URI');
            }
        }
    }
    throw new OverflowException('Uri was not detected. Make sure the
    REQUEST_URI is set.');
}
```

```php
public static function format($uri, $server) {
    // Remove all characters except letters,digits and $-_.+!*'(),{}|\\^~[]`<>#%";/?:@&=.
    $uri = filter_var(rawurldecode($uri), FILTER_SANITIZE_URL);
    // remove script path/name
    $uri = static::remove_script_name($uri, $server);
    // remove the relative uri
    $uri = static::remove_relative_uri($uri);
    // return argument if not empty or return a single slash
    return trim($uri, '/') ?: '/';
}
public static function remove_script_name($uri, $server) {
    return static::remove($server->get('SCRIPT_NAME'), $uri);
}
public static function remove_relative_uri($uri) {
    // remove base url
    if($base = Config::app('url')) {
        $uri = static::remove(rtrim($base, '/'), $uri);
    }
    // remove index
    if($index = Config::app('index')) {
        $uri = static::remove('/' . $index, $uri);
    }
    return $uri;
}
```

2、知识点

知识点	说明
FILTER_SANITIZE_URL	删除字符串中所有非法的URL字符，允许正常数字字母和符号
parse_url()	解析一个URL并返回其组成部分为一个数组
PHP_URL_PATH	获取URL中的URI

3、解读

1）themes\default\404.php 第4行输出了$current_url函数。

2）anchor\functions\helpers.php，可以发现$current_url函数返回了Uri类中的静态函数current()。

3）system\uri.php，函数current()中，如果静态变量$current为空，就将静态方法detect()赋值给$current并作为返回值。

4）函数detect()中，获取$_SERVER中 'REQUEST_URI'、'PATH_INFO'、'ORIG_PATH_INFO'三个键的值，如果存在任意一个值，就赋给$uri变量，并使用 filter_var删除非法URL字符，最后提取出URL中的URI赋值给$uri，通过静态函数format()返回数值。

5）函数format()中，进行了三次过滤，获取到用户访问的文件名。

4、分析

1）通过访问不存在的页面，会自动跳转到404.php，服务器提取URI作为报错信息的一部分进行返回在HTML标签中。

2）由于没有对XSS进行过滤，那么此时就可以构造一个不存在的URI，并在/后面加上Payload，触发XSS攻击。

5、利用

```
    Payload：
    http://localhost/anchor/index.xxx/<script>alert(1)</script>
```

6、修复方案

```
    对XSS漏洞，最好就是过滤关键词，对关键字符进行HTML实体编码替换。
```

7、参考链接

```
    https://github.com/hongriSec/PHP-Audit-
Labs/blob/master/Part1/Day2/files/README.md
```