



# 抓包原理介绍

Edit

New page

[Jump to bottom](#)

wanghongenpin edited this page 3 hours ago · 6 revisions

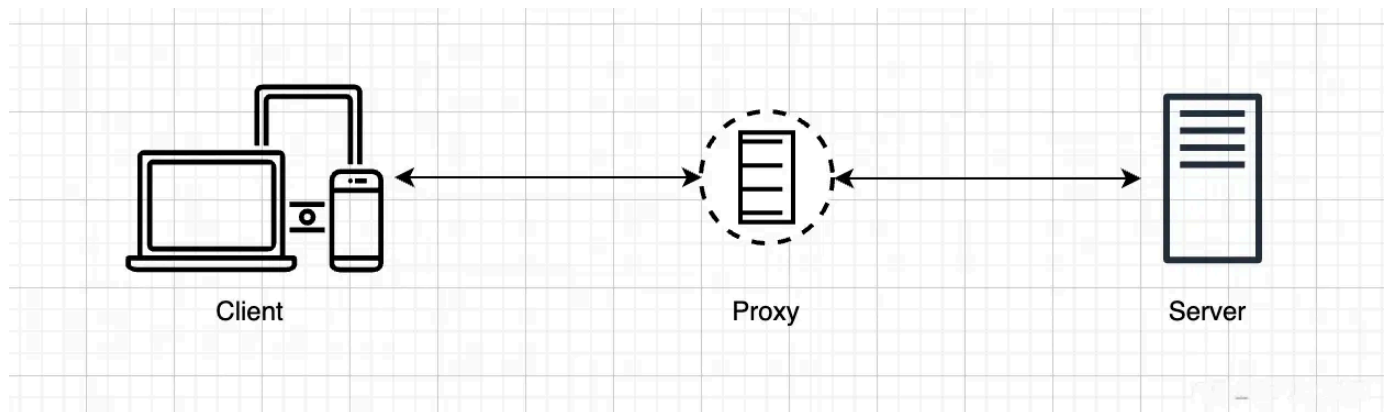
## 开源免费抓包ProxyPin, 支持全平台系统。

Github: <https://github.com/wanghongenpin/proxypin>

### 运行原理

[ProxyPin](#)运行机制是在启动了一个本地代理服务器（默认监听端口9099）来接收网络请求，当客户端与代理服务器进行通信时，ProxyPin会使用自签名SSL证书进行客户端SSL握手通信。为了保证客户端与中间人成功进行SSL握手通信，需要将ProxyPin的根证书安装到系统里。

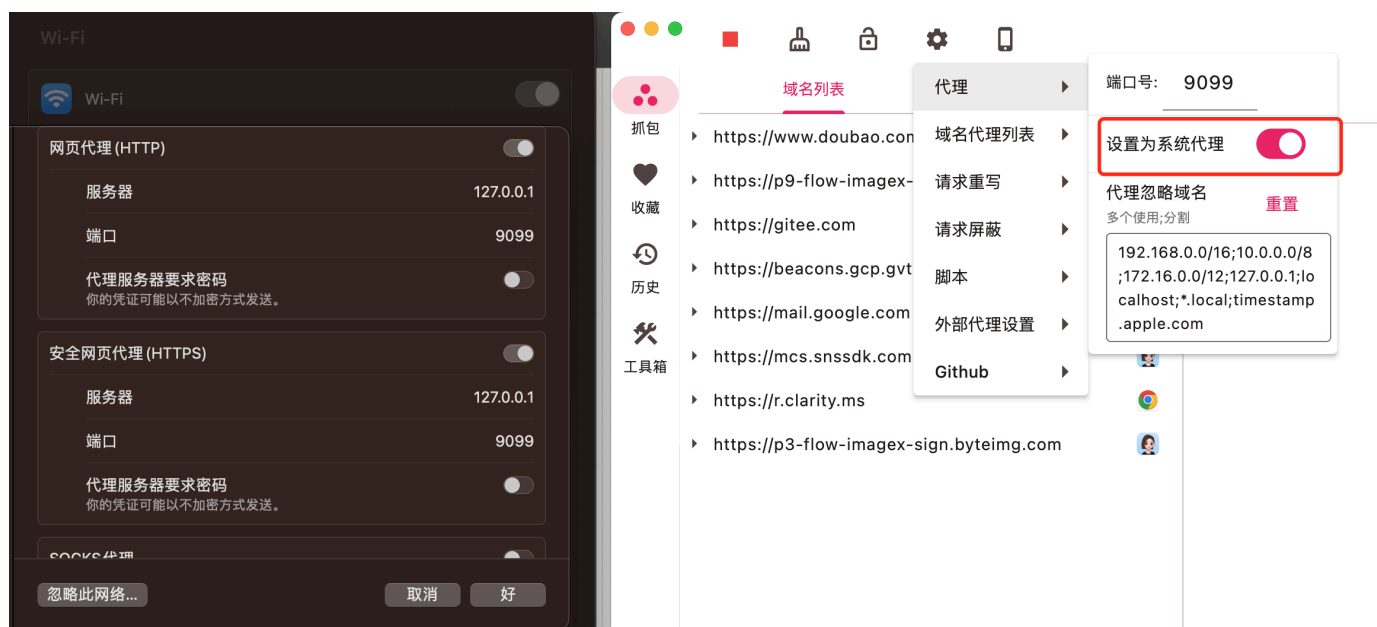
### 🔗 拦截流量



#### 桌面端：

抓包软件会把系统网络代理设置为代理服务器的地址和端口，比如浏览器等其他客户端发送网络请求时，会将流量转发到代理监听的端口上，从而被ProxyPin代理服务器捕获。但是有些软件并不会读取系统代理信息，可以借助于其他软件(如Proxifier)将流量转ProxyPin。

由于系统代理地址只能设置一个，如果你开启了其他VPN软件，那么就会冲突，ProxyPin提供了外部代理配置，可以将外部代理配置成VPN监听的端口，这样ProxyPin会把流量转发到其他VPN软件，从而实现访问全球网站。



## 手机端：

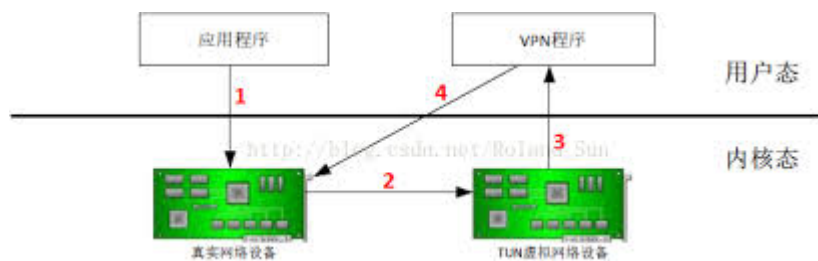
### 第一种配置Wi-Fi代理：

你可以将Wi-Fi代理配置成代理服务器的地址和端口，手机和电脑需要在同一个局域网下，这样系统会将所有流量转发给代理服务器，但是Flutter应用发起的请求不会走代理



### 第二种使用VPN服务：

VpnService会创建一个虚拟网卡，然后通过NAT，将所有数据包转发到TUN虚拟网卡上去。VPN程序读取该虚拟网卡设备上的数据，可以获得所有转发到TUN虚拟网络设备上的IP包。应用程序会解析IP数据包，获取TCP/UDP数据，如果是UDP会原文转发到实际的目标地址，TCP数据会判断是否是HTTP(S)协议包，如果是将会转发到ProxyPin代理服务，从而实现数据拦截处理。



## HTTPS协议

HTTP协议是明文传出, 可以直接解析HTTP报文.



## 什么是HTTPS?

超文本传输协议安全(HTTPS)是HTTP的安全版本。HTTPS在HTTP协议的基础上使用TLS/SSL加密进行通信，以提高数据传输的安全性。

## TLS和SSL之间有什么区别?

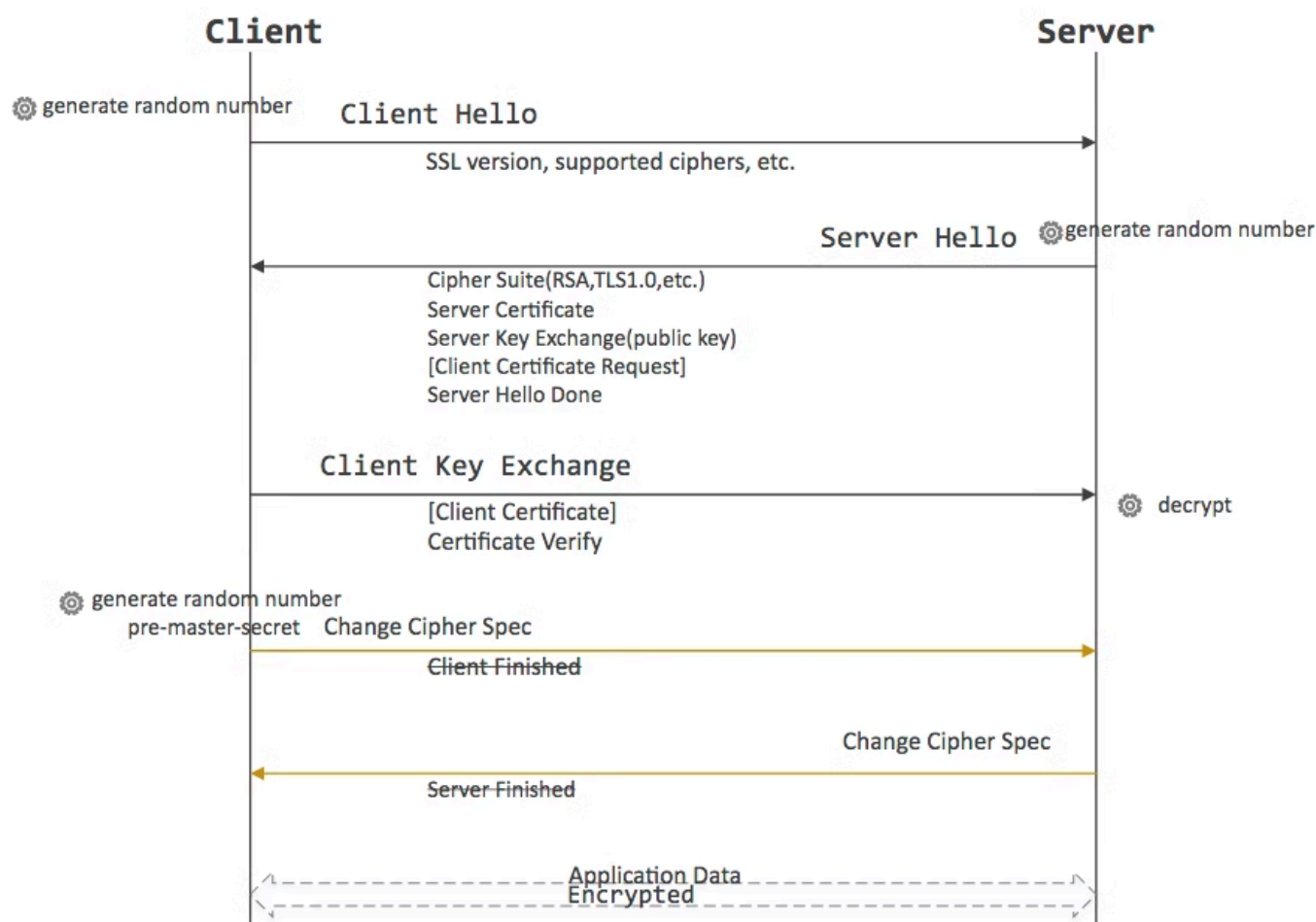
Netscape(网景)开发了名为安全套接字层 (Secure Socket Layer, SSL) 的加密协议，用于在网络上的两个设备之间创建安全连接。

但是，SSL是一种较老的技术，包含一些安全漏洞。传输层安全性协议 (TLS) 是SSL 的升级版，用于修复现有SSL漏洞。TLS1.0版实际上最初作为SSL3.1版开发，但在发布前更改了名称，以表明它不再与Netscape关联。由于这个历史原因，TLS和SSL这两个术语有时会互换使用。

现在使用的都是TLS协议，目前常用版本TLS1.2或较新TLS1.3。

## TLS/SSL如何工作?

HTTPS为了兼顾安全与效率，同时使用了对称加密 和 非对称加密。在TLS/SSL握手阶段，使用非对称加密来保证对称加密密钥在传输过程中的安全性，在 HTTPS 连接建立后，会使用对称加密算法对实际传输的数据。因为通常对称加密算法速度更快。



1. 客户端发送Client Hello 该消息包含客户端支持的 TLS 版本，支持的加密算法套件，以及一串称为“客户端随机数（client random）”的随机字节。
2. 服务端回应Server Hello 服务器接收到客户端的请求后，会选择一个双方都支持的 TLS/SSL 版本和加密算法套件，并将其包含在响应消息中发送回客户端。同时，服务器还会发送自己的数字证书，这个证书包含了服务器的身份信息、公钥等。
3. 证书校验、密钥交换 客户端收到服务器的响应后，会验证服务器的证书。这包括检查证书的颁发机构是否可信（通常通过预安装在系统里或客户端的受信任的根证书列表进行验证）。如果证书验证不通过，客户端会向用户发出警告，可能会中断连接。如果证书验证通过，客户端会使用证书中的公钥加密随机生成的会话密钥（非对称加密），并将其发送给服务器。
4. 加密通信 服务器使用私钥解密得到会话密钥后，双方就可以使用这个会话密钥进行对称加密通信。对称加密算法速度快，适合加密大量的数据。

## 解密数据

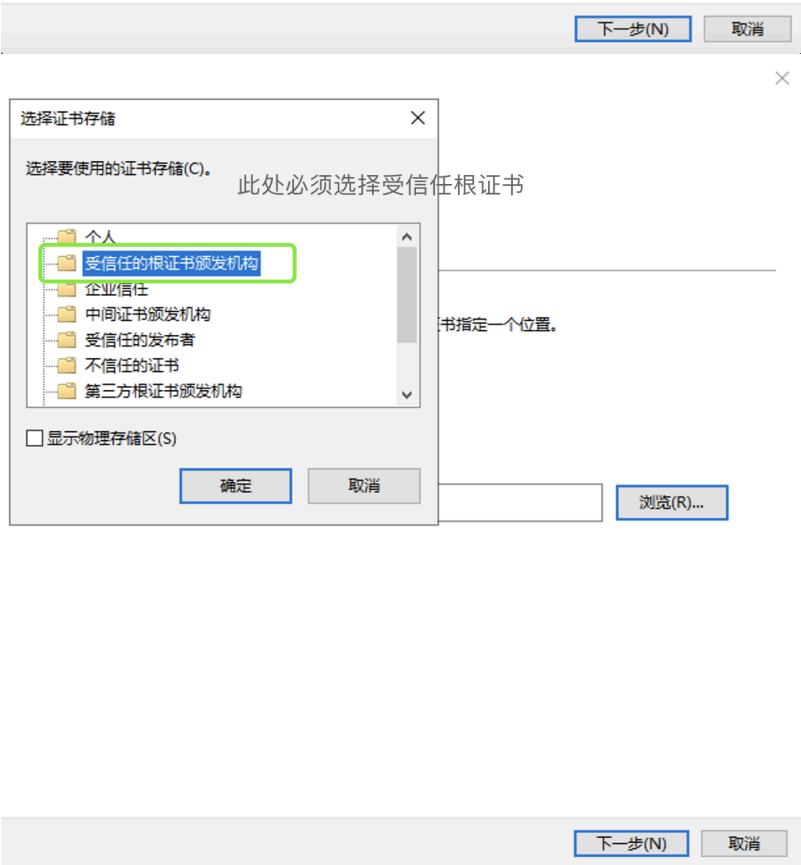
从TLS握手可知，我们如果想解密数据，可以在收到TLS Client Hello包时，响应给客户端自签名的证书，这样后续就可以用自签名证书解密密钥信息。但是由于我们自签名证书不是受信任机构签发的，所以需要把根证书安装到系统受信任根证书里。

## 根证书安装

如果你想拦截电脑流量需要在电脑上安装根证书，拦截手机需要在手机端安装证书

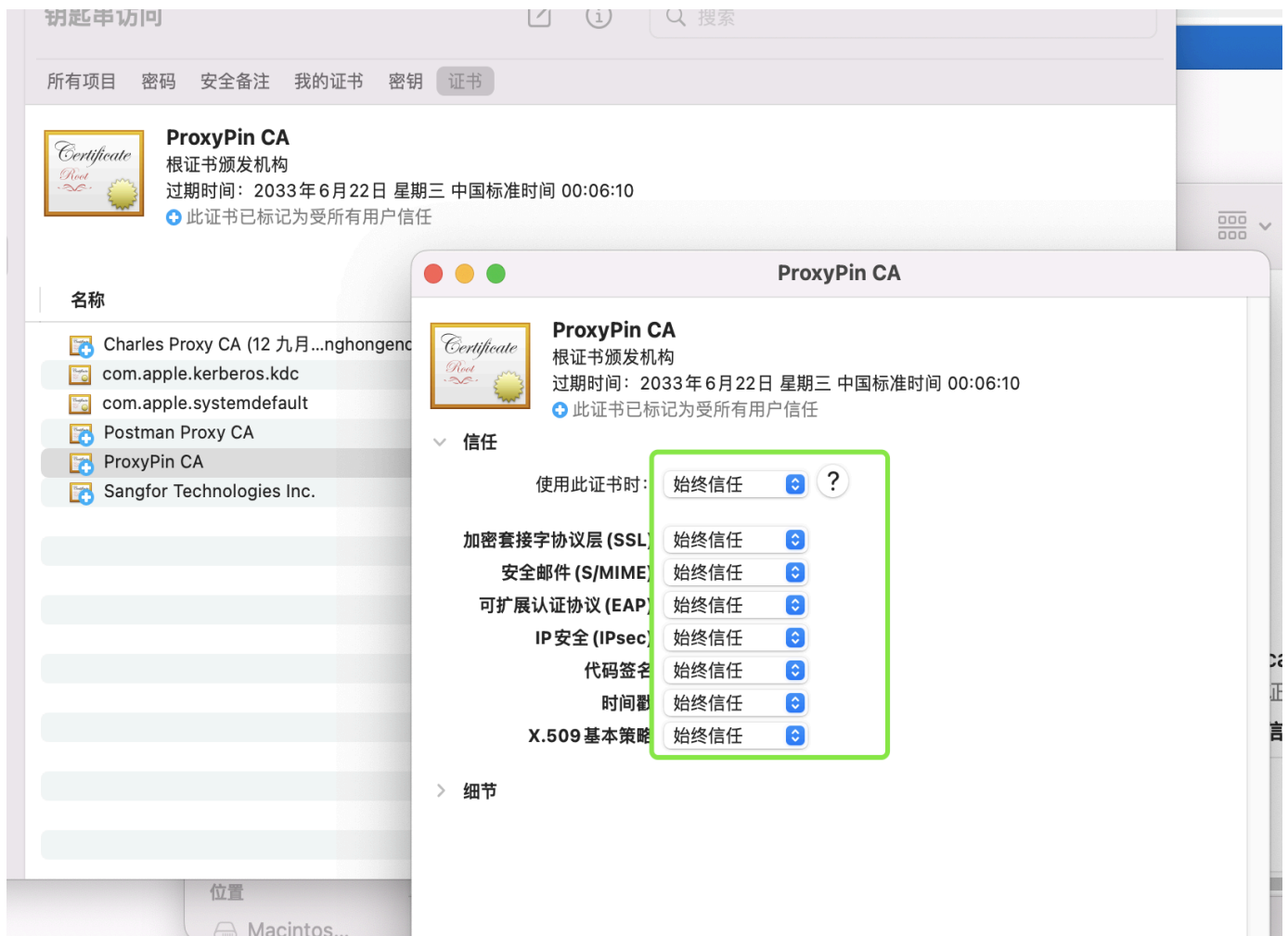
## Windows根证书安装

点击安装证书或者双击证书文件进行安装,选择“受信任的根证书颁发机构”



## Mac根证书安装

点击安装证书或将证书文件拖拽到钥匙串系统证书里, 安装完双击选择“始终信任此证书”



## iOS根证书安装

下载手机端 [ProxyPin](#) 在HTTPS代理下载根证书或通过配置WI-FI代理连接电脑访问进行下载证书。下载完整证书需要信任证书。

安装证书 设置 > 已下载描述文件 > 安装

信任证书 设置 > 通用 > 关于本机 > 证书信任设置



## 安卓根证书安装

Android7（不包括）以下，用户目录证书权限和系统目录证书权限是一样的,直接安装证书即可. Android7之后, Android系统禁用了用户目录证书的权限，用户目录的证书默认不再作为可信任证书，apk需要在network\_security\_config.xml配置中信任用户证书



## 加密与凭据

加密

### 加密手机

已加密

凭据存储

#### 信任的凭据

显示信任的 CA 证书



#### 用户凭据

查看和修改存储的凭据



#### 安装证书

从存储设备安装证书



#### 清除凭据

删除所有证书



#### 证书管理应用

无



所以Android7之后大部分应用需要把证书安装到系统下，才可以进行抓包。但是安装到系统里需要ROOT，您也可以使用MuMu等模拟器。

系统版本 < 14, 安装到系统证书目录,需要安装到/system/etc/security/cacerts.

系统版本 >= 14, 安装到系统证书目录,需要安装到/apex/com.android.conscrypt/cacerts.

### Magisk模块:

安卓ROOT设备可以使用 [Magisk ProxyPinCA](#) 系统证书模块, 安装完重启手机后 在系统证书查看是否有ProxyPinCA证书, 如果有说明证书安装成功。

[您也可以使用frida绕过安卓ssl pinning](#)

无ROOT可以使用 [Xposed模块抓包](#) 只能hook原生请求, flutter等请求无法处理。

## 请求调试



捕获到流量之后，可以使用调试功能进行数据模拟等各种测试。ProxyPin提供了非常强大的调试功能，三 有重写、断点和脚本功能。

## 请求重写

可通过重写规则来修改请求和响应内容。目前重写支持5种类型，分别是替换请求、替换响应、修改请求、改响应和重定向。

### 替换请求

表示整体替换请求数据，支持替换的部分包括：请求方法、请求路径、请求头和请求体。

### 替换响应

此重写行为表示整体替换响应数据，支持替换的部分：状态码方法、响应头、响应体。

请求重写

是否启用请求重写

使用文档

名称	行为
heartbeat	修改响应
	替换请求
	替换响应

请求重写规则

启用: ☒

名称:

URL:

行为: 

替换响应

状态码

响应头

响应体

类型: 文本

启用: ☒

消息体替换为:

```
{
  "data": {
    "app_id": "497858",
    "app_user_info": {
      "bui_audit_info": {
        "audit_status": 2,
        "is_auditing": false,
        "last_modify_time": 1719905647,
        "unpass_reason": ""
      },
      "avatar_url": "https://p26-passport.byteacctimg.com/img/user-avatar/assets/e7b19241fb224cea967dfaea35448102_1080_1080.png~300x300.image",
      "bg_img_url": "https://p3-passport.byteacctimg.com/origin/60e0000116e1e21ff9a",
      "bui_audit_info": {
        "audit_info": {},
        "is_auditing": false,
        "audit_status": 2,
        "last_update_time": 0,
        "unpass_reason": "",
        "details": {}
      },
      "can_be_found_by_phone": 1,
      "connects": [
        {
          "platform": "aweme_v2",
          "profile_image_url": "https://p26.douyinpic.com/aweme/1080x1080/aweme-avatar/tos-cn-avt-0015_6335c157201b3b1f07a257db2cdc9711.jpeg?from=4010531038"
        }
      ]
    }
  }
}
```

关闭

保存

### 修改请求

相比于替换请求行为，修改请求提供了更加细致化的修改策略。例如删除查询参数，正则替换请求体的内容。

举个例子。有下面这样的请求参数，我们希望修改name的值为345，但保留其他的参数不变。  
<https://example.com?name=123&age=32>

请求重写规则

使用文档

启用: ☐

添加

类型

修改参数

匹配

name=123

替换

name=345

测试数据

version\_code=20650&device\_platform=web&aid=497858&use-olympus-account=1&is\_new\_user=0&name=123&age=32

取消

确认

关闭

保存

修改响应

基本操作同上面修改请求。

脚本

ProxyPin提供了一个脚本功能，开发人员可以编写JS代码以灵活的方式操作请求/响应



编辑脚本 ?

X

名称: 请输入名称

URL: github.com/api/\*

脚本:

```

1  // 在请求到达服务器之前,调用此函数,您可以在此处修改请求数据
2  // 例如Add/Update/Remove: Queries、Headers、Body
3  async function onRequest(context, request) {
4      console.log(request.url);
5      //URL参数
6      //request.queries["name"] = "value";
7      // 更新或添加新标头
8      //request.headers["X-New-Headers"] = "My-Value";
9
10     // Update Body 使用fetch API请求接口, 具体文档可网上搜索fetch API
11     //response.body = await fetch('https://www.baidu.com/').then(response => response.text());
12     return request;
13 }
14
15 // 在将响应数据发送到客户端之前,调用此函数,您可以在此处修改响应数据
16 async function onResponse(context, request, response) {
17     // 更新或添加新标头
18     // response.headers["Name"] = "Value";
19     // response.statusCode = 200;
20
21     // 删除标头
22     // delete response.headers["Key-Need-Delete"];

```

取消

保存

## API说明

### function onRequest(context, request)

在请求到达服务器之前,调用此函数,您可以在此处修改请求数据

具体参数格式见 > 参数定义

```

async function onRequest(context, request) {
    console.log(request.url);
    //URL参数
    request.queries["name"] = "value";
    //更新或添加新标头
    request.headers["X-New-Headers"] = "My-Value";
    delete request.headers["Key-Need-Delete"];

    //Update Body 使用fetch API请求接口, 具体文档可网上搜索fetch API
    //request.body = await fetch('https://www.baidu.com/').then(response => response

    //共享参数 后面onResponse时取出
    context["name"] = "hello";

```



```
    return request;
}
```

## 请求中断

如果需要中断此请求，onRequest函数结果返回null即可！

## function onResponse(context, request, response)

在将响应数据发送到客户端之前,调用此函数,您可以在此处修改响应数据

```
async function onResponse(context, request, response) {
    // 更新或添加新标头
    // response.headers["Name"] = context["name"];

    // Update status Code
    response.statusCode = 500;

    //var body = JSON.parse(response.body);
    //body['key'] = "value";
    //response.body = JSON.stringify(body);
    return response;
}
```



## 参数定义

```
//context
{
    "os": "macos",
    "scriptName": "Your Script Name",
    "deviceId": "设备id",
    "session": {} //运行时会话对象，不同请求可传递参数
}

//request
{
    "method": "<String> HTTP Method. 示例: GET, POST, ...",
    "host": "<String> *只读* 域名. Ex: www.baidu.com, localhost, ...",
    "path": "<String>: URL Path. Ex: /v1/api",
    "queries": "<Map<String, String>> JS字典对象URL参数",
    "headers": "<Map<String, String>> JS字典对象所有Header键值",
    "body": "<String> 请求体字符串类型 json格式转换对象需要调用JSON.parse(request.body)"
}

//response
{
    "statusCode": 200,
    "headers": "<Map<String, String>> JS字典对象包含所有Header键值",
    "body": "<String> 请求体字符串类型 json格式转换对象需要调用JSON.parse(request.body)"
}
```



# JS内置方法

## MD5

```
var hash = md5('value') //output 2063c1608d6e0baf80249c42e2be5804
```



## File 文件操作

API和dart一致，见dart File文档 <https://api.dart.ac.cn/stable/3.4.4/dart-io/File-class.html>

//File(path): 创建一个表示文件的对象，参数 path 是文件的路径。

```
var file = File('file.path');
```

//异步读取文件为字符串

```
var text = await file.readAsString();
```

//异步将字符串内容写入文件。

```
await file.writeAsString('text');
```



## Fetch API使用参考文档

[https://developer.mozilla.org/zh-CN/docs/Web/API/Fetch\\_API/Using\\_Fetch](https://developer.mozilla.org/zh-CN/docs/Web/API/Fetch_API/Using_Fetch)

<https://www.ruanyifeng.com/blog/2020/12/fetch-tutorial.html>

+ Add a custom footer

► Pages 8

+ Add a custom sidebar

## Clone this wiki locally

<https://github.com/wanghongenpin/proxypin.wiki.git>

