一、Nginx 反爬（IP 层安全）

1. **IP QPS 限制：**

```
limit_req_zone $binary_remote_addr zone=goods_limit:10m rate=5r/s;

location /api/goods/list {
    limit_req zone=goods_limit burst=10 nodelay;
}
```

效果：

- 每个 IP 每秒最多 5 个商品列表请求
- 爆发可到 10 个，但会被平滑

2. **按 IP 限制并发数：**

```
limit_conn_zone $binary_remote_addr zone=conn_zone:10m;

location /api/goods/list {
    limit_conn conn_zone 5;  # 每个IP 只能 5 个同时进行的请求
}
```

效果：

- 每个 IP 只能 5 个同时进行的请求

3. **UA 黑名单 + 空 UA 拦截：**

```
#  判断 User-Agent 是否为空
if ($http_user_agent = "") {
    return 403;
}
# 屏蔽常见命令行爬虫 / 脚本工具
if ($http_user_agent ~* (curl|wget|httpclient|scrapy)) {
    return 403;
}
# 屏蔽常见开发语言/库 UA
if ($http_user_agent ~* (python|java|Go-http-client|axios|requests)) {
    return 403;
}
# 限制 Referer 必须是你的网站
if ($http_referer !~* "yourdomain\.com") {
    return 403;
}

# 只针对接口检查 Referer（不要对网页检查！）
location /api/ {
    if ($http_referer !~* "yourdomain\.com") {
        return 403;
    }
}
```

效果:

- 判断 User-Agent 是否为空.如果 User-Agent 为空（很多爬虫、脚本、curl 默认不会带 UA），就直接返回 **403 Forbidden**
- 屏蔽常见命令行爬虫 / 脚本工具**curl|wget|httpclient|scrapy**
- 屏蔽常见开发语言/库 UA **(python|java|Go-http-client|axios|requests)**
- 限制 Referer 必须是你的网站

二、后端反爬

1. **接口header签名验证:**

   中间件代码:

```
  public function handle(Request $request, \Closure $next)
    {
        $pathInfo = $request->pathinfo();
        if (in_array($pathInfo, $this->url)) {
            try {
                $service = new
SignatureVerifierService("V@PkQOJlO.CJKu_nb45Vt)y.@LmZBS1B");

//                $params = file_get_contents(''); // 或 GET 参数
                $raw = file_get_contents('php://input');
                $params= json_decode($raw,true);
                @file_put_contents('111.txt','原始数
据: '.json_encode($params,JSON_UNESCAPED_UNICODE+JSON_UNESCAPED_SLASHES).PHP_
EOL,FILE_APPEND);

                $timestamp = $_SERVER['HTTP_X_TIMESTAMP'];
                $signature = $_SERVER['HTTP_CERTIFICATE'];

                if (!$service->verify($params, $timestamp, $signature)) {
                    throw new Exception('签名失败');
                }
            }catch (\Exception $exception){
//                throw new Exception('签名失败');
                throw new Exception($exception->getMessage().'_'.$exception-
>getLine().'_'.$exception->getFile());
            }
        }

        return $next($request);
    }
```

   签名验证类:

```
<?php

namespace app\api\service;

class SignatureVerifierService
{
    private $hmacKey;
    private $timestampKey = 'X-Timestamp';
```

```
    private $expireSeconds = 10; // 10 秒有效期

    public function __construct(string $hmacKey)
    {
        $this->hmacKey = $hmacKey;
    }

    /**
     * 验证签名
     *
     * @param array $params 普通参数，例如 $_POST、$_GET
     * @param string $clientTimestamp Header: X-Timestamp
     * @param string $clientSignature Header: X-Signature
     * @return bool
     */
    public function verify(array $params, string $clientTimestamp, string
$clientSignature): bool
    {
        // 1. 校验时间戳范围（10 秒）
        if (!$this->verifyTimestamp($clientTimestamp)) {
            return false;
        }
        // 2. 参数处理
        $all = $params;

        // 时间戳加入参与签名（跟你的 JS 一致）
        $all[$this->timestampKey] = $clientTimestamp;
        // 3. 排序
        krsort($all);
        @file_put_contents('111.txt','排序后数
据：'.json_encode($all,JSON_UNESCAPED_UNICODE+JSON_UNESCAPED_SLASHES).PHP_EOL
,FILE_APPEND);
        // 4. 拼字符串 key=value&key2=value2
        //
        $signString = http_build_query($all);
        @file_put_contents('111.txt','拼接符
串：'.$signString.PHP_EOL,FILE_APPEND);

        // 5. HMAC-SHA256 + Base64
        $serverSignature = base64_encode(hash_hmac('sha256', $signString,
$this->hmacKey, true));
        @file_put_contents('111.txt','签
名：'.$serverSignature.PHP_EOL,FILE_APPEND);
        // 6. 比较
        return hash_equals($serverSignature, $clientSignature);
    }

    /**
     * 时间戳校验（10 秒内有效）
     */
    private function verifyTimestamp(string $clientTimestamp): bool
    {
        if (!ctype_digit($clientTimestamp)) {
            return false;
        }
```

```
        $client = intval($clientTimestamp);
        $server = time();
        return abs($server - $client) <= $this->expireSeconds;
    }
}
```

效果：

- 前端接口header增加X-Timestamp(当前时间戳)和Certificate(签名)
- 验证header X-Timestamp的时间是否小于当前时间10秒，防止重放
- body转数组增加一个键为X-Timestamp=header的X-Timestamp值；
- 按照数组的键（key）进行降序排序
- 把数组或对象转换成 URL 查询字符串（query string）
- 先用 HMAC-SHA256 加密，再用 Base64 编成可传输的字符串。
- 对比header传递的Certificate和后端的签名

2. **接口返回数据进行加密：**

效果:

- 后端按照上述加密方法对返回的数据进行加密
- 加密的数据按@或者特殊符号拼接密钥再 Base64 返回给前端
- 前端解密数据

3. 代码限制接口请求频率

```
namespace app\middleware;

use think\facade\Cache;
use think\Request;

class RateLimit
{
    public function handle(Request $request, \Closure $next)
    {
        $ip = $request->ip();
        $key = "rate_limit_{$ip}";
        $limit = 10;  // 1分钟内最多10次
        $count = Cache::get($key) ?: 0;

        if ($count >= $limit) {
            return json(['code'=>429, 'msg'=>'请求太频繁']);
        }

        Cache::set($key, $count + 1, 60); // 1分钟过期

        return $next($request);
    }
}
```

效果:

- 使用redis针对每个用户请求商品列表进行限流
- 可以设置每个用户针对某个接口 n次/n秒限制,超过限制加入黑名单。需要前端验证真人解除

三、前端反爬

1. **接口header签名验证:**:

   签名流程:

   - 请求接口header增加X-Timestamp(当前时间戳)和Certificate(签名)
   - body转数组增加一个键为X-Timestamp=header的X-Timestamp值;
   - 按照数组的键（key）进行降序排序
   - 把数组或对象转换成 URL 查询字符串（query string）
   - 先用 HMAC-SHA256 加密，再用 Base64 编成可传输的字符串。
   - 发起请求
2. **接口返回数据进行解密:**

   解码流程:

   - 接口获取返回的response,Base64 decode转成二进制，按照后端约定的分隔符截成密钥和数据原
   - 按照约定的方法进行解密
3. **JSONP 数据请求:**

   描述:使用 **JSONP（JSON with Padding）** 技术，通过动态创建 `<script>` 标签向接口发起请求，并通过全局回调函数获取数据。这种方式可以跨域获取接口数据，并在一定程度上增加爬虫抓取难度。

   方案实现流程:

   - **生成动态回调函数**
     - 每次请求生成随机函数名，例如：`mtopjsonp1234`
     - 回调函数绑定到 `window` 对象，用于接收接口返回的数据
   - **构造接口请求参数**
     - 包含关键字段，如:
       - `appKey`：应用标识
       - `t`：时间戳
       - `sign`：签名（可随机或加密）
       - `api`：接口名称
       - `v`：版本
       - `callback`：回调函数名
       - `data`：请求参数 JSON
   - **动态创建 `<script>` 标签**
     - 将请求 URL 设置为 `script.src`，插入 DOM
     - 接口返回数据时，会执行回调函数
   - **处理返回数据**
     - 在回调函数中，将数据渲染到页面
     - 清理：删除全局函数和 `<script>` 标签

3️ 优点

- **跨域请求简单**：无需服务端 CORS 配置
- **动态回调增加爬虫成本**：每次请求回调函数不同，爬虫抓取难度增加
- **前端完全控制数据展示**：可按需渲染商品列表
- **可模拟接口签名**：给爬虫增加逆向成本