

# AlFuzzing使用说明

---

简单配置，可直接食用

## 一、未授权测试配置

1. 基本场景
2. 配置文件
3. 未授权漏洞确认

## 二、越权检测配置

### 1. 检测流程

状态码检查

敏感数据检测

特殊情况

未检测到敏感数据

兜底机制

## 三、不存在越权漏洞的情况

## 四、查看结果

结果查看建议

## 五、常见问题

## 六、进阶配置

使用小tips

了解到部分师傅们的反馈，工具配置不太会用，很多师傅们可能更想快速进行一键使用。为此，这里详细介绍工具的使用方法，帮助师傅们能够快速上手！

---

## 简单配置，可直接食用

适合入门使用，无需复杂配置，下载后简单配置即可。

\*\*|注：\*\*|目前工具打包的配置文件已满足日常使用需求，师傅们下载后一般无需改动即可使用。

# 一、未授权测试配置

以下以 pikachu 靶场 为例，说明工具如何使用：

## 1. 基本场景

假设网站应用以 `cookie` 为鉴权凭据，如下图所示：



在实战中，可以通过以下方法测试：

- 删除 `cookie` 或 `token` 等鉴权凭据，观察是否返回数据。
- 如果删除后仍返回数据，有以下两种可能：
  - a. 存在未授权漏洞。
  - b. 鉴权 `token` 以参数形式呈现（此场景后续会迭代优化）。

## 2. 配置文件

工具默认配置了常见的鉴权请求头，如下图所示：unauthorizedScan

```
"unauthorizedScan": {  
  "enabled": true,  
  "removeHeaders": [  
    "Authorization",  
    "Cookie",  
    "Token",  
    "Jwt",  
    "X-Auth-Token",  
    "X-Csrf-Token",  
    "X-API-Key"  
  ],  
}
```

- 如果实际测试过程中遇到其他鉴权凭据，可以手动追加到配置文件 `config.json` 中。
- 重新运行程序后，工具会自动移除鉴权凭据（如 `cookie` 或 `token`），根据内置规则，判断是否存在未授权漏洞。

### 3. 未授权漏洞确认

- 如果删除鉴权凭据后，仍能访问敏感数据，并且置信度>60分，则工具会直接返回存在未授权漏洞。
- 如果未授权响应未匹配到敏感信息，但实际存在未授权漏洞（如查询、删除、修改操作），则会交由后续的越权检测处理。

以下是未授权响应的示例：

12693	https://mall.tl	POST /mall/openrest/	✓	200	296	JSON
12690	https://a	POST /saas/merchant/tag/update	✓	200	642	JSON
12689	https://a	POST /saas/merchant/tag/update		200	600	HTML

### Request

Pretty Raw Hex OneScan

```

7RBrS20qMpW5xjrlw2FMqvYxVMIwAvA%3D%3D; noShowTelTip=true; _fid=
2ab5cb13-e28c-4d94-83d7-b57aafd20892; SessionToken=
3bcc248f-00e4-49b7-8820-b2dd6663135e622; cityid=0101;
businessLine=myelong; _tctma=
20377580.1742796556520060.1742796556134.1742867255453.1744596937
262.4
Content-Length: 57
Sec-Ch-Ua-Platform: "macOS"
Sec-Ch-Ua: "Google Chrome";v="135", "Not-A.Brand";v="8",
"Chromium";v="135"
Sec-Ch-Ua-Mobile: ?0
Usertoken: BZiggmZsQNK1-p86DNzm3g
Actionfrom: PC
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0
Safari/537.36
Accept: application/json, text/plain, */*
Content-Type: application/json;charset=UTF-8
Token:
Origin: ht
Sec-Fetch-Site: same site
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer:
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN, zh;q=0.9
Priority: u=1, i
Connection: keep-alive
{
  "name": "1",
  "id": 65,
  "business": "100072",
  "templateId": 875
}

```

### Response

Pretty Raw Hex Render OneScan

```

1 HTTP/1.1 200 OK
2 Access-Control-Allow-Credentials: true
3 Access-Control-Allow-Origin: http
4 Access-Control-Request-Private-Network: true
5 Connection: keep-alive
6 Content-Type: application/json;charset=UTF-8
7 Date: Mon, 21 Apr 2025 08:31:59 GMT
8 Janus-Addr: MTA2LjEyMC4yMTcuNDQ=
9 Janus-Configid: 629db30e574bd7001d8582a4
10 P3p: CP="IDC DSP COR ADM DEVI TAIi PSA PSD IVAi IVDi CONi HIS OUR
IND CNT"
11 Server: openresty/1.15.8.2
12 Vary: Origin
13 Vary: Access-Control-Request-Method
14 Vary: Access-Control-Request-Headers
15 Content-Length: 84
16
17 {
  "code": "0",
  "msg": "成功",
  "data": null,
  "traceId": "e28b0224181d5b50751ec93673fd8461"
}

```

## 二、越权检测配置

### 1. 检测流程

越权检测主要分为以下步骤：

#### 状态码检查

- 替换请求返回以下状态码时，直接判定无漏洞：
  - 401（未授权）
  - 403（禁止访问）
  - 404（未找到）
  - 500（服务器错误）

#### 敏感数据检测

- 在原始响应和替换响应中检测敏感数据（如手机号、身份证、邮箱等）。

- 如果两者包含相同敏感数据，继续使用相似度判断：
  - 相似度阈值配置： `"similarityThreshold": 0.5`
  - 如果相似度 > 阈值且包含敏感数据，则直接返回确认漏洞。

```
"similarityThreshold": 0.5,
```

### 特殊情况

- 如果原始响应无敏感数据，但替换响应有，标记为 `unknown`，需人工确认【多数情况下是由于服务端鉴权了，返回的数据是从token里面取出】

### 未检测到敏感数据

对于未检测到敏感数据的场景（如修改、删除、增加操作），工具会将以下内容发给 AI 进行越权检测：

- 原始请求头和响应
- 替换为 B 账号 `token` 后的重放响应和状态码

工具内置了 AI 的 `prompt`：

```

var Prompt = `{
  "role": "你是一个AI，负责通过比较两个HTTP响应数据包来检测潜在的越权行为，并自行做出判断。",
  "inputs": {
    "reqA": "原始请求A",
    "responseA": "账号A请求URL的响应。",
    "responseB": "使用账号B的Cookie（也可能是token等其他参数）重放请求的响应。",
    "statusB": "账号B重放请求的请求状态码。",
    "dynamicFields": ["timestamp", "nonce", "session_id", "uuid", "request_id"]
  },
  "analysisRequirements": {
    "structureAndContentComparison": {
      "urlAnalysis": "结合原始请求A和响应A分析，判断是否可能是无需数据鉴权的公共接口（不作为主要判断依据）",
      "responseComparison": "比较响应A和响应B的结构和内容，忽略动态字段（如时间戳、随机数、会话ID、X-Header等）",
      "httpStatusCode": "对比HTTP状态码：403/401直接判定越权失败（false），500标记为未知（unknown），",
      "similarityAnalysis": "使用字段对比和文本相似度计算（Levenshtein/Jaccard）评估内容相似度。",
      "errorKeywords": "检查responseB是否包含 'Access Denied'、'Permission Denied'、'403 Forbidden'等关键词",
      "emptyResponseHandling": "如果responseB返回null、[]、{}或HTTP 204，且responseA有数据，判定为越权失败",
      "sensitiveDataDetection": "如果responseB包含敏感数据（如手机号、身份证号、邮箱、中文姓名、银行卡号等），判定为越权失败",
      "consistencyCheck": "如果responseB和responseA结构一致但关键数据不同，判定可能是权限控制正确（true）",
    },
    "judgmentCriteria": {
      "authorizationSuccess (true)": "如果不是公共接口，且responseB的结构和非动态字段内容与responseA一致，判定为权限控制正确",
      "authorizationFailure (false)": "如果是公共接口，或者responseB的结构和responseA不相似，或者responseB包含敏感数据，或者responseB返回500，或者responseA和responseB结构不同但没有限权相关信息，或者responseB包含错误关键词，判定为越权失败",
      "unknown": "如果responseB返回500，或者responseA和responseB结构不同但没有限权相关信息，或者responseB包含敏感数据，或者responseB返回500，或者responseA和responseB结构不同但没有限权相关信息，或者responseB包含错误关键词，判定为未知",
    }
  },
  "outputFormat": {
    "json": {
      "res": "\"true\"", "\"false\" 或 \"unknown\"",
      "reason": "清晰的判断原因，总体不超过50字。"
    }
  }
}

```

## 兜底机制

- 如果 AI 配置错误或未开启 AI 功能，会使用相似度判断是否存在漏洞。
- 若相似度 > `similarityThreshold` 阈值，则标记为需人工确认【对于修改、删除、添加的越权场景兜底】。

## 三、不存在越权漏洞的情况

以下场景会过滤掉越权漏洞：

### 1. 未开启 AI 或 AI 配置错误：

示例：

- 响应体命中过滤关键字。

```
pretty Raw Hex OneScan
Brs20qMpW5xjrrLw2FMqvYxVMIwAvA#3D#3D; noShowTelTip=true; _fid=
2ab5cb13-e28c-4d94-83d7-b57aafd20892; SessionToken=
3bcc248f-00e4-49b7-8820-b2dd6663135e622; cityid=0101;
businessLine=myelong; _tctma=
20377580.1742796556520060.1742796556134.1742867255453.17445969372
62.4
Content-Length: 56
Sec-Ch-Ua-Platform: "macOS"
Sec-Ch-Ua: "Google Chrome";v="135", "Not-A.Brand";v="8",
"Chromium";v="135"
Sec-Ch-Ua-Mobile: ?0
Ustoken: BZiggmZsQNK1-p86DNzm3g
Actionfrom: PC
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0
Safari/537.36
Accept: application/json, text/plain, */*
Content-Type: application/json; charset=UTF-8
Token: 6
Origin: https://xiaodian.elong.com
Sec-Fetch-Site: same-site
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer:
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN, zh;q=0.9
Priority: u=1, i
Connection: keep-alive

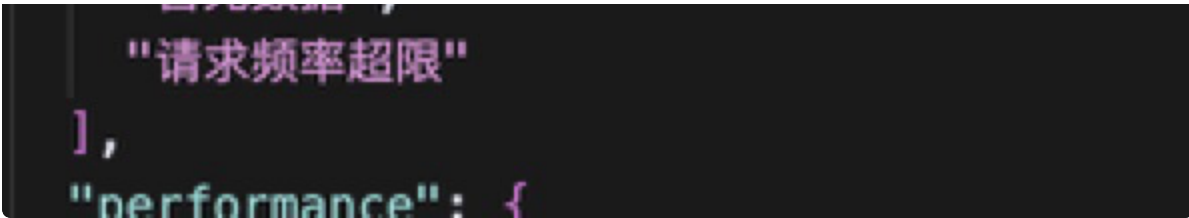
{
  "name": "1",
  "id": 1,
  "business": "100072",
  "templateId": 875
}
```

```
pretty Raw Hex Render OneScan
1 HTTP/1.1 200 OK
2 Access-Control-Allow-Credentials: true
3 Access-Control-Allow-Origin: https://xiaodian.elong.com
4 Access-Control-Request-Private-Network: true
5 Connection: keep-alive
6 Content-Type: application/json; charset=UTF-8
7 Date: Tue, 22 Apr 2025 09:13:11 GMT
8 Janus-Addr: MTA2LjEyMC4yMTcuNDQ=
9 Janus-Configid: 629db30e574bd7001d8582a4
10 P3p: CP="IDC DSP COR ADM DEVI TAIi PSA PSD IVAi IVDi CONi HIS OUR
IND CNT"
11 Server: openresty/1.15.8.2
12 Vary: Origin
13 Vary: Access-Control-Request-Method
14 Vary: Access-Control-Request-Headers
15 Content-Length: 101
16
17 {
  "code": "1000_0003",
  "msg": "无权限操作",
  "data": null,
  "traceId": "d3582cade1f2d9c20a413f098c8f79f6"
}
```

"respBodyBWhiteList": [

"参数错误",  
"数据页数不正确",  
"文件不存在",  
"系统繁忙, 请稍后再试",  
"请求参数格式不正确",  
"权限不足",  
"Access Denied",  
"Permission Denied",  
"Unauthorized Access",  
"请先登录",  
"无权限操作",  
"没有权限",  
"Token不可为空",  
"会话已过期",  
"暂无数据",





#### 检测结果摘要

URL https://...merchant/tag/update  
方法 POST  
状态 安全请求

置信度 未知  
漏洞类型 越权访问  
原因 [相似度计算] 替换请求响应包含错误信息: 无权限操作

#### 请求对比

##### 原始请求

```
Referer: https://...  
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) A  
Cookie: _ga=GA1.1.581066523.1742796589; _ga_3448JMBL38=GS1.1.  
Content-Length: 56  
Sec-Fetch-Site: same-site  
Sec-Fetch-Mode: cors  
Sec-Ch-Ua-Platform: "macOS"  
Priority: u=1, i  
Ustoken: BZiggmZsQNK1-p86DNzm3g  
Actionfrom: PC  
Sec-Fetch-Dest: empty  
Sec-Ch-Ua: "Google Chrome";v="135", "Not-A.Brand";v="8", "Chr  
Sec-Ch-Ua-Mobile: ?0  
{ "name": "1", "id": "1", "business": "100072", "templateId": "875" }
```

##### 未授权请求

```
Origin: https://...  
Referer: https://...  
User-Agent: AIFuzzing  
Cookie: test  
Sec-Fetch-Mode: cors  
Custom-Header: CustomValue  
Sec-Ch-Ua-Mobile: ?0  
Sec-Fetch-Site: same-site  
Priority: u=1, i  
Actionfrom: PC  
Token: xxx  
Sec-Fetch-Dest: empty  
Ustoken: BZiggmZsQNK1-p86DNzm3g  
Sec-Ch-Ua: "Google Chrome";v="135", "Not-A.Brand";v="8", "Chr  
Sec-Ch-Ua-Platform: "macOS"
```

## 2. 响应体相似度小于阈值:

- 工具会直接过滤掉该响应。

## 四、查看结果

了解工具基本原理后，只需简单配置即可使用：

- 访问功能点，进行操作（点点点）。
- 返回结果页面：`http://127.0.0.1:8222`。

方法	漏洞类型	状态	发现时间	详情
GET	未授权访问	存在漏洞	2025/4/22 17:5...	包含敏感数据的未授权访问: 高置信度未... <a href="#">查看详情</a>

## 结果查看建议

- 优先查看存在漏洞的结果。
- 查看完成后，再检查需人工确认的结果。

## 五、常见问题

### 1. 误报太高

- 可能是部分过滤后缀不在默认配置文件中，根据个人场景添加一下。

图片里面的内容都是黑名单过滤，不要误解了。

```
"suffixes": [  
    ".js",  
    ".css",  
    ".ico",  
    ".jpg",  
    ".jpeg",  
    ".png",  
    ".gif",  
    ".svg",  
    ".woff",  
    ".woff2",  
    ".ttf",  
    ".eot",  
    ".map",  
    ".webp"  
],  
"allowedRespHeaders": [  
    "image/png",  
    "image/jpeg",
```

```
"image/gif",
"text/css",
"application/javascript",
"application/x-javascript",
"application/pdf",
"font/woff",
"font/woff2",
"application/octet-stream"
],
"respBodyBWhiteList": [
    "参数错误",
    "数据页数不正确",
    "文件不存在",
    "系统繁忙，请稍后再试",
    "请求参数格式不正确",
    "权限不足",
```

- 公共接口误报太多，目前内置部分公共接口过滤规则如下，可以根据个人所需补充，过滤掉。

```

],
"similarityThreshold": 0.5,
"excludePatterns": [
    "/static/",
    "/login",
    "/logout",
    "/register",
    "/signin",
    "/signup",
    "/assets/",
    "/js/",
    "/css/",
    "/img/",
    "/fonts/",
    "/upload/",
    "/download/",
    "/upload",
    "/download"
],
"sensitiveDataPatterns": {

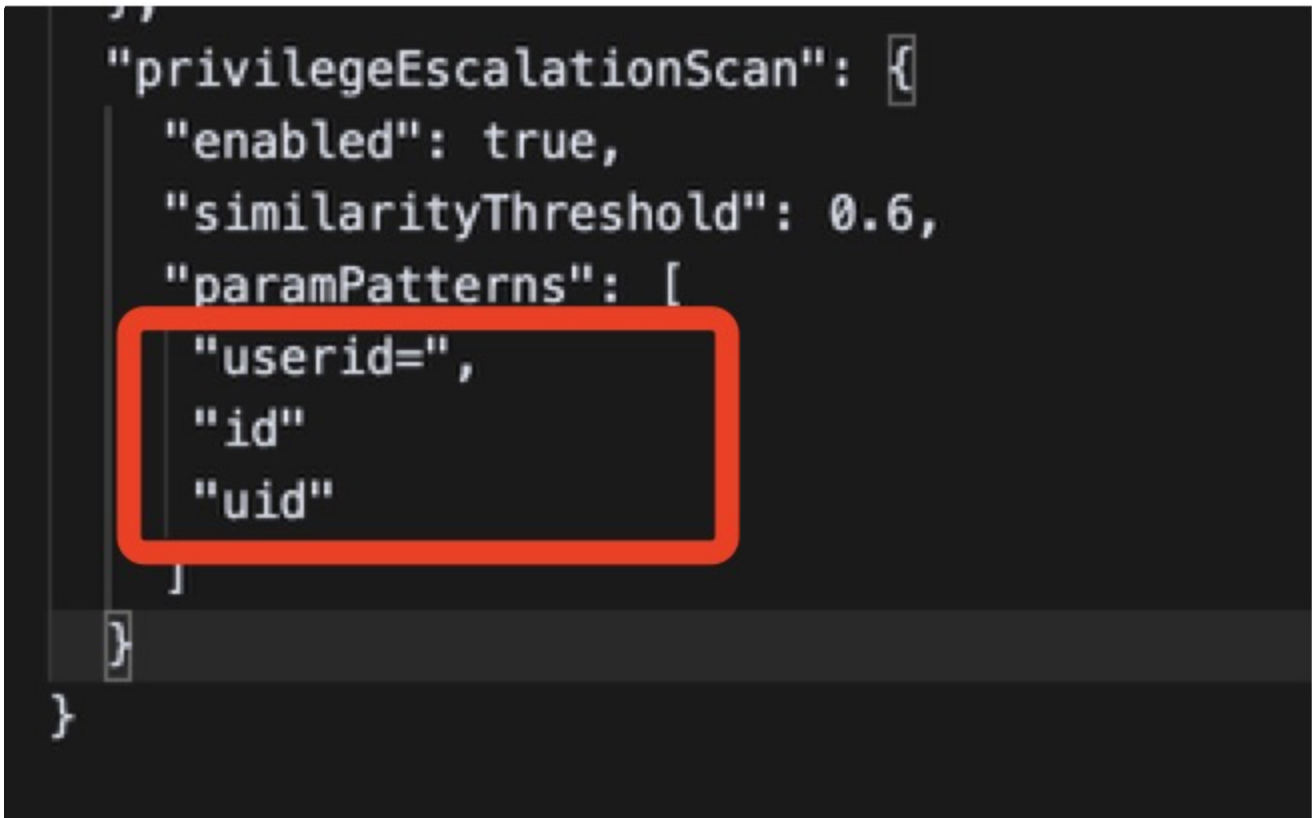
```

- whitelist白名单未配置，默认为空，对所有流量都会fuzz，导致很多无效目标公共接口命中，可以添加whitelist顶级域名白名单，只对指定目标测试【记得是配置顶级域名，例如baidu.com、qq.com】
1. 证书还是有问题,拦截不到https流量

- 1 前提：本地已经安装了go环境，没有安装的自行百度安装下go环境。
- 2
- 3 1. 下载 go-mitmproxy
- 4 `go install github.com/zt2/go-mitmproxy@latest`
- 5 安装完成后，go-mitmproxy 的可执行文件会位于 `$GOPATH/bin` 目录下。
- 6 2. 验证安装
- 7 运行以下命令，检查 go-mitmproxy 是否成功安装：
- 8 `go-mitmproxy --help`
- 9 如果显示帮助信息，说明安装成功。
- 10 3. go-mitmproxy 运行后，会默认生成证书到`~/.mitmproxy/`下，安装下生成的证书【怎么安装证书就不多说了】
- 11
- 12 如果`~/.mitmproxy/`目录下有证书，安装了还是拦截不到
- 13 1. `go-mitmproxy -cert_path ~/mitmproxy` #生成证书到指定目录`~/mitmproxy`下，然后再去这个目录下重新安装一遍证书

## 六、进阶配置

1. 只对包含了指定参数的请求进行fuzz，可以在`excludePatterns`里面配置：



```
    "privilegeEscalationScan": {  
      "enabled": true,  
      "similarityThreshold": 0.6,  
      "paramPatterns": [  
        "userid=",  
        "id"  
        "uid"  
      ]  
    }  
  }  
}
```

配置后，只有包含了这些参数的请求才会被fuzz。

2. 如果实际场景中，默认未授权打分规则、越权置信度阈值不符合个人使用需求，动态调整即可。

```
"useConfidenceScore": true,
"highConfidenceScore": 60,
"mediumConfidenceScore": 40,
"lowConfidenceScore": 20,
"confidenceRules": [
  {
    "name": "contains_sensitive_data",
    "description": "响应中包含敏感数据（如手机号、身份证等）",
    "weight": 60
  },
  {
    "name": "successful_status_code",
    "description": "响应状态码为2xx",
    "weight": 5
  },
  {
    "name": "json_response",
    "description": "响应为有效的JSON格式",
    "weight": 5
  },
  {
    "name": "similar_content_length",
    "description": "与原始响应长度相似",
    "weight": 20
  },
  {
    "name": "api_endpoint",
    "description": "请求URL是典型的API端点",
    "weight": 10
  }
]
```

3. 可以根据个人需求添加敏感信息匹配正则,目前仅内置了用户名、身份证、手机号。

## 使用小tips

- 当我们手工对某个应用测试出逻辑漏洞后，我们可以通过快速配置，对该应用进行批量fuzz【更推荐！】

以上是 AIFuzzing 工具的使用指南，希望对师傅们有所帮助，若您看完后还有使用上的困惑，可以进群私聊！