目录

背景

当下大部分的互联网公司都面临着业务欺诈风险，尤其是是金融、社交、电商等行业。虚假注册、流量欺诈、刷单、薅羊毛、垃圾广告等各类业务欺诈方式层出不穷，导致厂商蒙受直接的经济和口碑损失。

手机号注册由于注册过程方便快捷，并且后续可以与用户及时沟通，已经成为当前最主流的账号注册方式，许多业务的账号安全体系也是依赖手机号建立的。

手机号是当前互联网业务中非常重要的身份验证要素，与此同时，也是黑产与厂商对抗的核心资源。黑产使用的手机卡（手机黑卡）从收购、销售，到使用（活动欺诈）、回收，已经发展为一条完整的产业链，对厂商的业务安全造成了巨大威胁。因此，手机号维度信誉的检测和风险的识别，可以有效帮助厂商降低业务欺诈风险。

产品功能

手机号画像数据来源自第三方的蜜罐网络流量监测及行业数据共享，每日新增数万恶意手机号码，并同步给企业用于业务场景下的风险审计，准确度超过99%，能在用户注册的第一时间判断该号码是否存在过恶意行为，具有极强的实用价值。

产品功能

1）手机号风险等级说明

通过进行手机卡属性查询，服务端将会返回四种风险等级标签【risk】，分别为：

风险等级	分数值	说明
高风险	9分	猫池、接码平台号码，标记为恶意手机号；包括薅羊毛号、恶意注册、刷卡、刷单、刷积分等，可通过风控系统反馈业务系统直接拦截
中风险	5分	曾经有过恶意行为，可疑号码。可以与风控系统事前协商，提供给风控系统的该事件的一定风险分值；跟据风险系统策略的阈值来做最终判定
低风险	2分	虚拟小号；反馈给风控系统，给出一定的风险分值，并记录历史操作，依据企业风控策略做综合判定
无风险	0分	可通过风控系统反馈业务系统放行

2）手机号风险标签说明

字段	名称	说明
user	加密手机号（SHA1）	查询的手机号码
p_name_price	最近黑产项目/价格	该黑卡最近被发现可用于手机验证的黑产项目名称以及项目对应的价格
ctime	首次发现时间	该黑卡第一次被发现的时间
uptime	最近活跃时间	该黑卡最近被发现活跃时间
location	归属地	该卡的归属地。（大陆精确到市，国外卡只展示归属国家）
attribute	卡属性	识别该卡的所属运营商
card_type	卡类型	识别该卡的是普通卡还是虚拟卡拦截卡等。（拦截卡：被黑产控制用于收发验证码的真实用户手机号

价值优势

产品价值

■ 事前数据补充：

购买手机号画像，对接自有风控体系，做底层数据的补充，建立更加全面的用户画像；

■ 事中决策参考：

当业务遭受到黑产攻击时，可在风控系统中，针对手机号做实时的风险识别和阻断；

■ 事后分析工具：

可对无法确认风险的行为，从手机号维度补充风险依据，提高事后审计效率和准确率。

产品优势

■ 精准率高

数据来源自第三方的全球蜜罐网络恶意流量监测及行业数据共享，准确度高过99%。

■ 覆盖率高

每日新增数万恶意手机号码及时性高。

应用场景

■ 互联网金融

互联网金融是离钱最近的行业，多样的充值活动也成了羊毛党最喜欢聚集的行业。一个没有风控的充值活动，极有可能给产品造成损失。通过对羊毛党的手机卡进行识别，直接将羊毛党的行为拒之门外。

■电商

目前电商领域的欺诈风险主要集中在恶意刷单、垃圾注册、活动欺诈，通过接入手机号信誉库能有效识别这些虚假操作的手机号，直接降低平台因网络欺诈造成的资金损失及形象受损。

■社交

社交领域的欺诈风险主要集中在垃圾注册、活动欺诈、垃圾广告。这些欺诈行为都是通过黑卡进行注册，养号，然后再操作的。通过接入手机号信誉库，可以提高手机黑卡识别率，从而提高欺诈风险的拦截率。

快速入门

手动查询黑产手机号码：16573967191

risk值展示：

卡属性展示：

卡类型展示:

手动查询黑产号码：17001700591

手动查询黑产号码：16558606371

统计展示：

交付方式

RESTful API交付方式

客户可通过的RESTful API方式对接交付，通过线上API查询手机号，就能立刻返回手机号的风险信息。之后厂商只需要根据自己的规则，进行相关的逻辑判断即可。
每个注册客户会分配一个或多个accesskey key ID （即AK秘钥），需要通过AK秘钥验证方可对接云端API。查询方式详见下

图：

添加访问白名单

手机号画像服务对查询请求实行IP白名单机制，只有白名单中IP地址才会被允许访问金山云云端黑产库。用户可以在业务风险情报BRI控制台访问控制页面添加自己的服务器IP地址白名单，如下图：


技术指标说明：

默认QPS：1000
延时情况：300ms以内

请求结构

客户调用金山云业务风险情报服务(BRI)的openAPI接口是通过向指定服务地址发送请求，并按照openAPI文档说明在请求中添加相应的公共参数和接口参数来完成的。业务风险情报(BRI)openAPI的请求结构组成如下：

1.服务地址

http://bri.api.ksyun.com

2.通信协议

支持通过 HTTP 或 HTTPS 两种方式进行请求通信，推荐使用安全性更高的 HTTPS方式发送请求。

3.请求方法

业务风险情报(BRI)的openAPI同时支持GET和POST请求，推荐使用GET请求方式。

注意

•不能混合使用两种请求方式。如果使用 GET 方式，参数均从 querystring 取得；如果使用 POST 方式，参数均从 请求Body中取得;
•如果请求方式是GET，需要对所有请求参数做URL编码；如果请求方式是POST，需要使用x-www-form-urlencoded方式进行编码。

4.请求参数

金山云openAPI请求包含两类参数：公共请求参数和接口请求参数。其中，公共请求参数是每个接口都要用到的请求参数，具体可参见公共参数小节；接口请求参数是各个接口所特有的，具体见各个接口的“请求参数”描述。

5. 字符编码

请求及返回结果都使用utf-8进行编码。

公共参数

- **参数说明** 公共请求参数是每个业务风险情报(BRI)都需要使用到的请求参数。

名称	类型	是否必须参数	长度限制(字符)	参数格式	描述
Action	String	是	不确定	[a-zA-Z]+	操作接口名，与调用的具体openAPI相关
Version	String	是	10字符	YYYY-MM-DD	接口版本号，版本号不同接口支持的参数和返回值可能不同，业务风险情报当前只支持一个版本，即2019-12-18
X-Amz-Algorithm	String	是	16字符	AWS4-HMAC-SHA256	签名算法，目前只支持一种，即HMAC-SHA256
X-Amz-Credential	String	是	不确定	AccessKeyId/YYYYMMDD/region/service/AWS4_request	信任状信息，包括访问密钥ID，日期，region名称和服务名称以及结尾字符串AWS4_request

X-Amz-Date	String	否（用于覆盖信任状或者date header中的日期）	16字符	ISO 8601 基本格式 YYYYMMDD'T'HHMMSS'Z'，如20160304T120000Z	签名日期
X-Amz-Signature	String	是	64字符	**16进制编码**表示	请求签名值
X-Amz-SignedHeaders	String	是	不确定	[a-zA-Z0-9-;]+	需要在签名计算中包含的请求header
DryRun	Boolean	否	最长5字符	true(1) or false(0)	检查当前调用者是否有权限执行相关操作，而不是真的调用执行相关操作

- 示例

```
https://bri.api.ksyun.com/?
Action=CheckPhone&Version=2019-12-18
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKLTGo0pHK-EQWiDZWTSBSll2Q%2F20160914%2Fcn-beijing-6%2Fiam%2Faws4_request
&X-Amz-Date=20160914T114902Z
&X-Amz-SignedHeaders=host
&X-Amz-Signature=88f6284257863dedfc350da05d19d07f76cca622e93b829f5ce26c1a75d3da39
&接口请求参数
```

返回结果

调用金山云的openAPI服务，调用成功，返回的HTTP状态码（Status）为200；调用失败，返回4xx 或5xx的HTTP状态码（Status）。

金山云的业务风险情报(BRI)服务的调用返回的数据格式支持xml和json两种，默认返回xml格式，可通过设置HTTP Header Accept=application/json来改变返回数据格式。

调用成功

xml格式示例

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
    <RequestId>0717e32b-e5b9-9c05-08c1-55b795ea2bed</RequestId>
    </ResponseMetadata>
    <!--返回结果数据-->
</response>
```

json格式示例

```
{
    "RequestId": "4C467B38-3910-447D-87BC-AC049166F216"
    /*返回结果数据*/
}
```

调用失败

调用接口失败，不会返回结果数据；HTTP请求返回一个4xx或5xx的HTTP状态码，返回的HTTP消息体中包含具体的错误代码(code)及错误信息(message)；与调用成功一样还包含请求ID(RequestId)，在调用方找不到错误原因时，可以联系金山云客服，并提供RequestId，以便我们尽快帮您解决问题。

xml格式示例

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
<Error>
<Code>PermissionDenied</Code>
<InnerCode>permission_denied</InnerCode>
<Message>权限不足</Message>
</Error>
<RequestId>fd229f7c-e65f-1402-1d85-8cdff25c8b59</RequestId>
</response>
```

json格式示例

```
{
"Error": {
"Code": "PermissionDenied",
"InnerCode": "permission_denied",
"Message": "权限不足"
```

},
"RequestId": "0717e32b-e5b9-9c05-08c1-55b795ea2bed"
}

公共错误

错误代码(Code)	错误消息(Message)	HTTP 状态码	中文描述（语义）
MissingAuthenticationToken	Request is missing 'Host' header.	403	请求header中缺少Host
MissingAuthenticationToken	Request is missing Authentication Token.	403	请求header中缺少认证token
MissingAuthenticationToken	%s not in Http Header.	403	%s不在Http header中
SignatureDoesNotMatch	Host' must be a 'SignedHeader' in the Authorization.	403	请求的SignedHeader中必须包含Host
SignatureDoesNotMatch	Credential should be scoped with a valid terminator: 'aws4_request', not: %s.	403	请求Authorization header中的“Credential”末尾必须是“aws4_request”
SignatureDoesNotMatch	Credential should be scoped to a valid region, not:%s.	403	请求Authorization header中的“Credential”中的Region信息无效
SignatureDoesNotMatch	Credential should be scoped to correct service: %s.	403	请求Authorization header中的“Credential”中的Service信息无效
SignatureDoesNotMatch	The request signature we calculated does not match the signature you provided.	403	请求中提供的签名与实际计算结果不匹配
SignatureDoesNotMatch	Signature expired:%s.	403	签名已过期
SignatureDoesNotMatch	Date in Credential scope does not match YYYYMMDD from ISO-8601 version of date from HTTP.	403	请求Authorization header中的“Credential”中的Date应该是ISO8601基本格式，形如”YYYYMMDD“
InvalidClientTokenId	The security token included in the request is invalid.	403	请求中提供的AccessKeyId无效
AccessDenied	User: %s is not authorized to perform: %s.	403	用户%s无权限操作该资源：%s
IncompleteSignature	Date must be in ISO-8601 'basic format'. Got '%s'. See http://en.wikipedia.org/wiki/ISO_8601.	400	Date必须符合ISO_8601基本格式，参考：http://en.wikipedia.org/wiki/ISO_8601
IncompleteSignature	KSC query-string parameters must include %s. Re-examine the query-string parameters.	400	查询条件中缺少签署信息，查询条件中必须包含”X-Amz-Algorithm“、”X-Amz-Credential“、”X-Amz-SignedHeaders“、”X-Amz-Date“信息
IncompleteSignature	Unsupported ksc 'algorithm': %s.	400	只支持如下签名算法：AWS4-HMAC-SHA256
IncompleteSignature	Authorization header requires 'Credential' parameter. Authorization=%s.	400	请求Authorization header中需要包含“Credential”参数
IncompleteSignature	Credential must have exactly 5 slash-delimited elements, e.g. accesskeyid/date/region/service/aws4_request, got: %s.	400	请求Authorization header中中“Credential”至少包含5项以斜杠分隔的元素，如：keyid/date/region/service/aws4_request
IncompleteSignature	Authorization header format error.	400	请求Authorization header的格式错误
IncompleteSignature	Authorization header requires existence of either a 'X-Amz-Date' or a 'Date' header, Authorization=%s	400	请求中缺少“X-Amz-Date”或者“Date” header信息
IncompleteSignature	Authorization header requires 'Signature' parameter. Authorization=%s	400	请求Authorization header中缺少“Signature”信息
IncompleteSignature	Authorization header requires 'SignedHeaders' parameter. Authorization=%s	400	请求Authorization header中缺少“SignedHeaders”信息

ServiceUnavailable	Exception %s	500	服务暂不可用
ServiceUnavailable	Auth Service is unavailable because of an unknown error, exception or failure	500	验签或授权服务暂不可用
ServiceUnavailable	Request was rejected because it referenced an 'InnerApi' that does not have an internal service	404	请求被拒绝，因其引用的InnerAPI无内部服务。
ServiceUnavailable	OpenAPI or Service is unavailable because of an unknown error, exception or failure.	500	openAPI或服务暂不可用。
DryRunOperation	Request would have succeeded, but DryRun flag is set	412	请求本可成功，但由于设置DryRun标记未成功
NoSuchEntity	Request was rejected because it referenced an 'InnerApi' that does not exist.	404	请求被拒绝，因其引用的InnerAPI不存在
LimitExceeded	Request was rejected because the request speed of this openAPI is beyond the current flow control limit.	409	请求被拒绝，因该openAPI接口访问速度已达到流控上限
InvalidParameterValue	An invalid or out-of-range value was supplied for the input parameter %s.	400	输入参数%s的值无效、不合法或者超出范围
InvalidMethod	The method %s for is not valid for this web service.	400	Method %s对当前web服务无效
MissingParameter	An value must be supplied for the input parameter %s.	400	输入参数 %s的值不能为空
InvalidQueryParameter	The query parameter %s is malformed or does not adhere to KSC standards.	400	查询参数 %s格式不对、不存在或者不符合金山云标准
ServiceTimeout	Internal Service is unavailable because of time out.	500	内部服务由于超时暂不可用

签名机制

身份与访问管理的openAPI调用采用AWS签名算法版本4，具体可以参考AWS文档，支持GET和POST两种HTTP方法，GET方法所有请求参数包括signature放置在url中，POST方法则将Signature以名为authorization header的形式放置在header中，其主要区别在于GET方式处理的请求url长度不能过长。
签名计算的主要流程如下：

1.创建一个正规化请求

在签名前，首先将请求进行正规化格式化，目的是让签名计算过程无二意，其主要过程伪代码如下：

```
CanonicalRequest = HTTPRequestMethod + '\n' + CanonicalURI + '\n' + CanonicalQueryString + '\n' + CanonicalHeaders + '\n' + SignedHeaders + '\n' + HexEncode(Hash(RequestPayload))
```

Hash指代计算哈希的算法，目前使用SHA-256，HexEncode是对哈希值进行用16进制编码（使用小写字母）。

具体步骤如下

1.抽取HTTP请求方法（如GET、PUT、POST）结尾附加“换行符”

2.URI绝对路径进行URI编码得到正规化URI，如果绝对路径为空，那么使用前斜线"/"，结尾附加“换行符”

3.构建正规化Querystring，结尾附加“换行符”

- URI编码每一个querystring参数名称和参数值（注：GET方式需要包含哈希算法、信任状、签名日期和签名header等全部参数）
- 按照ASCII字节顺序对参数名称严格排序
- 将排序好的参数名称和参数值用=连接，按照排序结果将“参数对”用&连接

4.构建正规化headers，结尾附加“换行符”，伪代码如下：

```
CanonicalHeaders = CanonicalHeadersEntry0 + CanonicalHeadersEntry1 + ... + CanonicalHeadersEntryN
```

其中：

```
CanonicalHeadersEntry = Lowercase(HeaderName) + ':' + Trimall(HeaderValue) + '\n'
```

lowercase表示将header名字转为小写字母，trimall表示去掉header值前和值后的白空格，并将header值里面的连续白空格

变成单空格，但是不去掉双引号中间的任何空格，且最后的正规化headers是按照header名称排序后的结果

5.添加签名headers，结尾附加“换行符”。签名header是包含在正规化headers中名称列表，其目的是指明哪些header参与签名计算，从而忽略请求被proxy添加的额外header，其中host、x-amz-date两个header如果存在则必须添加进来，伪代码如下

```
SignedHeaders = Lowercase(HeaderName0) + ';' + Lowercase(HeaderName1) + ";" + ... + Lowercase(HeaderNameN)
```

然后处理请求body，如下

6.对请求body使用哈希算法（SHA256）计算哈希值，并将二进制哈希值结果用16进制编码表示出来（且不使用大写字符），伪代码

```
HashedPayload = Lowercase(HexEncode(Hash(requestPayload)))
```

此时，

- 将上述i-vi步骤的结果连接成一个字符串，即为正规化请求（Canonical Request）
- 将vii步的正规化请求使用vi步的哈希算法计算哈希值

2.创建签名字符串

签名字符串主要包含请求以及正规化请求的元数据信息，由签名算法、请求日期、信任状和正规化请求哈希值连接组成，伪代码如下：

```
StringToSign = Algorithm + '\n' + RequestDate + '\n' + CredentialScope + '\n' + HashedCanonicalRequest
```

其中，签名算法(Algorithm)为AWS4-HMAC-SHA256，请求日期(RequestDate)格式YYYYMMDD'T'HHMMSS'Z'，信任状(CredentialScope)格式为YYYYMMDD/region/service/aws4_request（包括请求日期（ISO 8601 基本格式）），正规化请求哈希值为上述1中第viii步的结果（注意结尾不附加“换行符”）；

3.计算签名信息

在计算签名前，首先从私有访问密钥（secret AccessKey）派生出签名密钥（signing key），而不是直接使用私有访问密钥；之后使用签名密钥和2中计算的签名字符串来计算签名值，具体计算过程如下

1. 生成签名密钥，伪代码如下

```
kSecret = Your KSC Secret Access Key
kDate = HMAC("AWS4" + kSecret, Date)
kRegion = HMAC(kDate, Region)
kService = HMAC(kRegion, Service)
kSigning = HMAC(kService, "aws4_request")
```

其方式是通过HMAC算法依次生成下一个HMAC的key值（第一个为私有访问密钥字符串），而data值则依次为信任状中的各项内容（日期、region、服务、结尾字符串）；HMAC算法采用HMAC-SHA256，返回值为哈希值二进制形式（256bit，32字节），不需要做8/16进制编码显示。

2.计算签名,伪代码如下：

```
signature = HexEncode(HMAC(derived-signing-key, string-to-sign))
```

使用HMAC-SHA256算法，以签名密钥作为key，签名字符串作为data计算签名，签名后的二进制哈希值结果以16进制编码输出。

手机号画像

PhonePortrait（手机号画像）

Contents(内容)

•手机号画像返回json

•类型：string

•是否可缺省：否

•json内容：

参数	参数名称	参数类型	参数说明

phone_number	加密手机号SHA1	str	sha1加密后的手机号码
ctime	首次发现时间	datetime	该黑卡首次被发现的时间
uptime	最近活跃时间	datetime	该黑卡最近被发现的时间
risk	风险类型	int	9：【高风险】黑卡：猫池号码；5：【中风险】黑产曾经使用过的号码；2：【低风险】虚拟小号；0：【无风险】 正常号码
location	归属地	str	大陆精确到市；非大陆精确到国家/自治区
attribute	卡属性	int	0：实体卡：移动、联通、电信运营商卡；1：虚拟卡：虚拟运营商卡；-1：阿里小号和国外号码卡属性均为-1；（当前版本，查询的正常号码此字段的返回值均为-1）
card_type	卡类型	int	0：普通卡 1：虚拟小号 2：VoIP网络电话 3：拦截卡
p_name_price	xxx注册/价格	str	在xxx平台使用过的
user	用户id	string	用户id

请注意：uptime、attribute、p_name_price字段仅在风险等级risk:9、5、2状态下有效。

代码示例

```
import requests
from requests_aws4auth import AWS4Auth
import time
import json
import hashlib
ak = 'your ak'
sk = 'your sk'
url = 'http://bri.api.ksyun.com'
region = 'cn-shanghai-3'
service = 'bri'
phones = ["15118376562","16573967191", "13470564531"]
usha1 = []
for phone in phones:
    phone = str(phone)
    phone = hashlib.sha1(phone.encode()).hexdigest()
    usha1.append(phone)
data = str(json.dumps(usha1))
params={"Action":"CheckPhone","Version":"2019-12-18","Data":data}
print('请求参数:')
print(params)
auth = AWS4Auth(ak, sk, 'cn-shanghai-3', 'bri')
response = requests.get(url,params=params, auth=auth)
print('响应内容:')
print(response.text)
```

API文档

CheckPhone（手机号画像api）

Request Paremeters（请求参数）

Data

- 所要查询的手机号的哈希值
- 类型：string
- 是否可缺省：否
- 说明：请求字符串为json序列化之后的字符串；支持境外手机号码

请求字符串示例：

```
'["ebe16d1826e6095c36d4c2ec325b5b178c5d3968", "4413d42b546156c7f100a95180a2bc0844c7b8fd", "716efa8e88fce982645f3104b7c37aef3679a0f5"]'
```

Response Eliments（响应内容）

Data

- 手机号画像返回内容

- 类型：PhonePortrait

RequestId

- 请求ID
- 类型string
- 是否可缺省：否