目录

背景

IP作为互联网空间中的地址标识，一直以来都是甲方厂商最重要的风控指标之一。

近年来互联网的线路资源、IP资源，已成为黑产链条精细化的一个环节，并已形成专门提供IP资源网络服务的“秒拨”动态IP黑产。此类黑产针对互联网行业帐号体系中的注册、登录等操作的具体环节，使用“秒拨”动态IP技术欺骗厂商的IP识别判定策略，从而实施撞库批量注册、养号、爬虫、刷单、诈骗、薅羊毛等恶意行为，给企业造成了巨大的损失。

产品功能

IP画像是一款针对业务场景IP风险实时判定的产品，其通过IP被黑产持有时间与IP业务访问时间比对的方式来判断IP在访问业务时被黑产持有的可能性，同时再结合IP属性类型历史行为等来实时判断IP在业务访问时刻的风险程度。产品采用秒级更新的机制，保证了风险判定的时效性问题，同时摒弃对IP进行复杂的标签标注行为，采用风险分值的方式来量化风险，解决了传统IP黑白名单场景不匹配及易用性差的问题；其不仅能识别传统的代理等黑IP等欺诈行为，更能精准的识别黑产使用秒拨IP的作恶行为，帮助厂商更好的对抗黑产的作恶行为，减少经济损失。

实现原理：

黑产IP资源平台下面会有多个提供如“秒拨”服务的IP资源池节点，虽然单个节点的IP资源数量过百万，但是在一段时间内，单个节点拥有的IP数量是有限的，而且在这段时间内，这些IP会被这个节点一直持有，直到触发某些更新机制（类似缓存）。因此我们可以通过蜜罐、探针等技术手段，实时获取每个节点当前所持有的IP，并对这个IP打上被黑产持有的时间标签，同时结合已有的IP信誉库进行风险分析。基于此，用户可以根据IP对业务的访问时间与被黑产的持有时间及IP历史恶意行为比对分析，以此判断IP的实时风险程度。

产品功能：

产品采用打分制，用户通过上传IP进行查询，系统将返回IP、危险分值，IP类型、及地理位置（精确到地市），分值范围0-100。由于产品时效性高，为实时更新，故在查询的时候，同一个IP前后两次查询结果可能会不同。
API实时查询示例：
返回数据（明文）


```
{
    "ip": "119.7.78.100",
    "type": "家庭宽带",
    "location": "中国 四川省 德阳市 旌阳区 中国联通 31.126972 104.393298 510600 CN 亚洲",
        "risk_tag": "代理:2020-03-01 19:23:27",
    "risk_score":99,
    "risk_level":"高"
}
```


字段说明

Data内容字段说明如下：

字段	说明
ip	所查询的IP
type	IP类型，包括ADSL、家庭宽带、数据中心、移动网络、企业专业、校园单位、未知
location	国家 省份 城市 区县 运营商 纬度 经度 行政区划代码 国家编码 大洲（空格分割）
risk_score	风险分数，范围0-100，得分越高被黑产持有的概率也就越高
risk_tag	风险标签，包括代理、秒拨、机房流量等，并附捕获时间精确至秒
risk_level	风险等级，包括高、中、低、无

风险标签[risk-tag]说明

标签内容	说明
代理	该IP在该时间点被网络黑产作为代理IP使用
秒拨	该IP在该时间点是网络黑产提供的秒拨IP资源（秒拨IP指，黑产利用家庭宽带拨号在一定时间内持有作恶的IP）
机房流量	该IP为IDC所属，通常正常用户流量不会从IDC发出
多开分身	在该时间使用该IP的设备，处于“多开分身”恶意设备环境（多开分身是一种作弊软件，实现在同一设备中，打开多个相同APP的，常见于薅羊毛、恶意引流等场景）
真人作弊	在该时间使用该IP的设备环境，多次命中真人作弊标签，有较大概率存在真人作弊风险
环境伪造	在该时间使用该IP的设备具有明显的改机特征（改机工具是一种提供伪造设备指纹功能的恶意软件，一些云控、云手机等群控类设备中也自带改机功能，同样会被识别标记为此标签）
伪造定位	在该时间使用该IP的设备具有以下特征，通过改机等手段实现虚拟定位或虚拟行驶

自动化脚本	在该时间使用该IP的设备具有启用模拟点击脚本（一种自动化模拟点击的工具，可以实现自动重复攻击，常见于薅羊毛、引流、恶意注册、刷量等场景）的特征

风险分值说明：

针对产品对IP给出的风险分数，用户可以根据以下说明进行处置：

风险等级	分数段	说明	处置建议
高危	100~94分	该IP当前被黑产持有可能性极高	建议采取较强限制策略或直接拦截
中危	94分~79分	该IP当前被黑产持有可能性为中	使用中等限制策略 如短信验证码，人工标记复审
低危	79分~10分	该IP当前被黑产持有可能性低	使用较低限制策略，如图形验证码
正常	10分~0分	该IP近期未发现有明显风险	建议直接放行

价值优势

产品优势

■ 易用性
去除对IP的复杂字段定义说明，量化风险值，不需要复杂的策略规则，直接根据风险值进行判定，风险分值实时更新，解决传统方案更新不及时带来的误判问题。

■ 适用性
产品的使用可忽略场景不匹配问题，去除了对IP的复杂字段定义说明，量化为风险值，不需要复杂的策略规则，直接根据风险值进行判定。

■ 唯一性
全国唯一能提供针对秒拨的防护能力，目前市面的IP风险标签产品无法对秒拨进行有效的识别。

应用场景

由于传统的IP风险标签数据产品无法对ADSL及基站等IP进行有效标签标注，黑产可使用“秒播”实现IP秒级更新从而绕过IP风控规则，实现批量账号登录，注册、撞库、领优惠券、数据爬取等操作，IP画像能有效解决这一问题。

■批量注册
描述：使用虚假号码、通信小号、小号邮箱进行批量注册的行为；
策略：结合业务逻辑进行判断，建议采取一些较强的限制策略；
举例：
某个业务接口短时间内出现大量的请求，虽然不是同一个IP，但这些IP绝大多数都在10分以及10分以上，那可以对这些IP进行限制。

■ 薅羊毛
描述：通过脚本、自动化程序及群控等，控制大量垃圾账号或进行领券、交易的行为；
策略：结合业务逻辑进行判断，建议采取一些较弱的限制策略； 举例：
某个时间段内大量用户进行登录、领取优惠券等操作行为，虽不是同一个IP，但这些IP绝大多数都在10分以及10分以上，限制其领券操作或返回登录验证操作等。

■爬虫
描述：通过脚本或程序自动地抓取网站信息的行为。
策略：结合业务逻辑进行判断，建议采取一些较弱的限制策略,限制IP的访问请求频率
举例：
某个时间段内，存在持续的大量访问操作，并且访问的IP都在10分及以上，限制IP的访问频次，如每秒只允许访问一个连接。

快速入门

黑产IP查询示例：120.9.132.181

risk值说明：

黑产IP查询示例：182.85.18.24

统计展示：

交付方式

RESTful API方式交付

客户可通过的RESTful API方式对接交付，通过线上API查询手机号，就能立刻返回对应IP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IP白名单

IP画像服务对查询请求实行IP白名单机制，只有白名单中IP地址才会被允许查询。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IP地址白名单，如下图：

技术指标说明：

默认QPS：1000（支持横向扩容）
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=CheckIp&Version=2019-12-18
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKLTGoOpHK-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。

IP画像

IpPortrait (Ip画像)

Contents (内容)

•ip画像返回json

•类型：string

•是否可缺省：否

•json内容：

字段	说明
ip	所查询的IP
type	ip类型：包括家庭宽带、数据中心、移动网络、企业专业、校园单位、未知
location	国家 省份 城市 区县 运营商 纬度 经度 行政区划代码 国家编码 大洲（’空格‘分割）
risk_tag	风险标签：包括代理、秒拨、机房流量、无，并附捕获时间精确至秒
risk_score	风险分数：0-100 得分越高黑产持有概率越高
risk_level	风险等级：高、中、低、无风险
user	用户id

代码示例

```
import requests
from requests_aws4auth import AWS4Auth
import time
import json
#ak = 'your ak'
#sk = 'your sk'
url = 'http://bri.api.ksyun.com'
region = 'cn-shanghai-3'
service = 'bri'
ips = ['61.145.48.124',"61.145.49.125"]
data = []
for ip in ips:
    data.append({"ip":ip,"t":"{}".format(int(time.time()))})
data = str(json.dumps(data))
params={"Action":"CheckIp","Version":"2019-12-18","Data":data}
print('请求参数:')
print(params)
auth = AWS4Auth(ak, sk, region, service)
response = requests.get(url,params=params, auth=auth)
print('响应内容:')
print(response.text)
```

API文档

CheckIp (ip画像api)

Request Paremeters (请求参数)

Data

• 所要查询的ip地址及其时间信息
• 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；
要序列化的json:

```
[{
 "ip": "61.145.48.124",
 "t": "1575010666"
}, {
 "ip": "61.145.49.125",
 "t": "1575010685"
}]
```

请求字符串:

'[{"ip":"61.145.48.124","t":"1575010666"},{"ip":"61.145.49.125","t":"1575010685"}]'

注： t: 时间戳，精确到秒，目前支持查询两周内IP风险分数；查询时请上传当前时间戳参数；（如不上传t则默认当前时间）

Response Eliments (响应内容)

Data

- Ip画像返回内容
- 类型：IpPortrait

RequestId
- 请求ID
- 类型string
- 是否可缺省：否