ByteDance 字节跳动 | Security 安全与风控 × 安全范儿 BYTEDANCE SECURITY

浅谈SaaS产品的安全攻防

Lum14n

目录

SaaS介绍与安全挑战

安全挑战

多租户共用

01

SaaS平台的多租户架构使得数据隔离成为一项挑战。不同租户共享同一基础设施，需要确保数据在存储、处理和传输过程中的严格隔离，以防止数据泄露和越权访问。

传统的后端接口鉴权无法直接应用于SaaS平台，因此需要采取额外的措施，如命名规范和命名空间，来实现租户之间的隔离。然而，一旦存在差异，就可能被攻击者利用。

功能复杂迭代快

02

SaaS产品的功能复杂性和快速迭代带来了安全挑战。频繁的功能迭代和更新增加了安全漏洞的风险，可能引入新的安全漏洞或意外的安全风险。

SDLC在SaaS产品中难以全面覆盖，部分功能的引入和组合可能导致未知的安全问题，给产品的安全性带来挑战。

三方组件依赖多

03

SaaS产品常常依赖多个第三方组件，而这些组件的安全性问题难以高效审计。传统的SDLC白盒工具往往无法有效检测和管理这些组件的安全问题。

第三方组件的功能迭代速度快，SaaS提供商难以感知和处理组件的更新，这可能导致潜在的安全风险。

实践思路与技巧

常规思路

获取产品

通过注册一个试用账号购买订阅或与产品提供商协商获得访问权限。内测产品问题往往更多

了解产品

通过浏览产品的官方网站、用户文档、帮助中心或产品演示视频等，了解产品的功能和用途。这可以帮助您理解产品的主要功能点、用户界面和操作流程。

探索功能

使用获得的访问权限，深入了解产品的各个功能点。尝试不同的功能、页面和菜单以及开放平台，了解产品的不同模块和交互方式。

寻找漏洞

页面各个功能点、SDK等帮助工具、OpenApi等，进行攻击渗透测试尝试。

测试技巧:转黑为白

寻找SaaS产品是否存在**私有化部署**的交付方式，从中可以获取到后端组建的整体架构、功能代码、二进制程序，转黑盒测试为(半)白盒测试。

转黑为白:测试思路

整体审视

1、审视整体架构和基础设施，包括网络拓扑、服务器配置和数据存储方式,了解到敏感数据存放方式，存放位置

2、查看docker启动进程，以及相关守护进程，了解docker启动后发了什么，都用到了哪些服务。他们之间的大致关联和用途

3、查看功能代码，了解网站路由，鉴权方式，相关插件等，为后续增加更多的畅想空间

功能审视

1、配合官网文档，产品控制台，了解实际业务功能后，审查后端的具体功能实现。

2、关注大致逻辑，用到的一些组件插件，外接api等，以及对输入参数的校验逻辑，审视其数据来源是否可控(二阶注入)

3、关注功能实现过程中可能涉及的数据处理和存储

流程审视

1、关注功能之间的流转，是否存在流程控制不严格的问题。

2、相关功能组合是否会有新的问题发生

转黑为白：真实案例1

某SaaS产品，存在私有化交付模式，拿到了相关docker镜像并在本地运行。

1、是一个nodejs实现的后端，并使用——inspect方式启动，默认暴露9229端口允许nodejs协议来进行调试

注：saas模式采用slb+nginx反代，是无法访问到该端口的

2、存在相关使用headless chrome截图定时发送邮件的功能

构造一个网页，发起websocket请求连接到localhost的9229端口，发送符合nodejs调试的消息，然后使用headlesschrome对其进行截图，来运行js的代码，达到任意命令执行的效果。

转黑为白：真实案例2

某SaaS产品，存在私有化交付模式，拿到了相关docker镜像并在本地运行功能逻辑比较简单，主要是白盒代码检测，对docker里进行查看后总结如下

1、是一个go实现的后端

2、实际逻辑不多，上传代码压缩包，检测是否存在安全漏洞

3、后端外接了一个程序，最后会将代码压缩包解压到临时目录，然后使用程序进行扫描。

利用ida查看go调用外部程序的逻辑，往上回溯处理http请求的方法，发现隐藏参数会拼接到命令里，注入恶意命令即可任意命令执行

go解压zip的时候，使用原生库处理，存在Zip Slip任意文件写漏洞，加上有程序的整个环境，审查了下配置文件，可以直接替换配置文件中外部程序存在的地址，达到rce。

测试技巧:第三方组件

，支持 HCL（Terraform）格式语法编写，

请使用Spark SQL语法，详情请参见 Spark SQL Guide

寻找控制台，SDK、OPENAPI使用过程中的关键报错信息，还有使用文档中的一些介绍，猜测使用到的第三方组件。测试组件是否被安全使用

```
at com.facebook.presto.jdbc.internal.okhttp3.i
at com.facebook.presto.jdbc.internal.okhttp3.i
at com.facebook.presto.jdbc.internal.okhttp3.i
at com.facebook.presto.jdbc.PrestoDriverUri.la
at com.facebook.presto.jdbc.internal.okhttp3.i
at com.facebook.presto.jdbc.internal.okhttp3.i
at com.facebook.presto.jdbc.internal.client.Ok
at com.facebook.presto.jdbc.internal.okhttp3.i
at com.facebook.presto.jdbc.internal.okhttp3.i
at com.facebook.presto.jdbc.internal.okhttp3.R
at com.facebook.presto.jdbc.internal.okhttp3.R
```

该工作流使用WDL编写，并通过 Cromwell 工作引擎调度运行。

第三方组件:真实案例(ClickHouse)

```
1 select * fro                                    latest/iam/security_credentials',CSV,'column1 String');
```

column1
ServiceRoleForVolcECS
KubernetesNodeRoleForECS

ClickHouse Product Resou

Docs Cloud SQL Reference Know

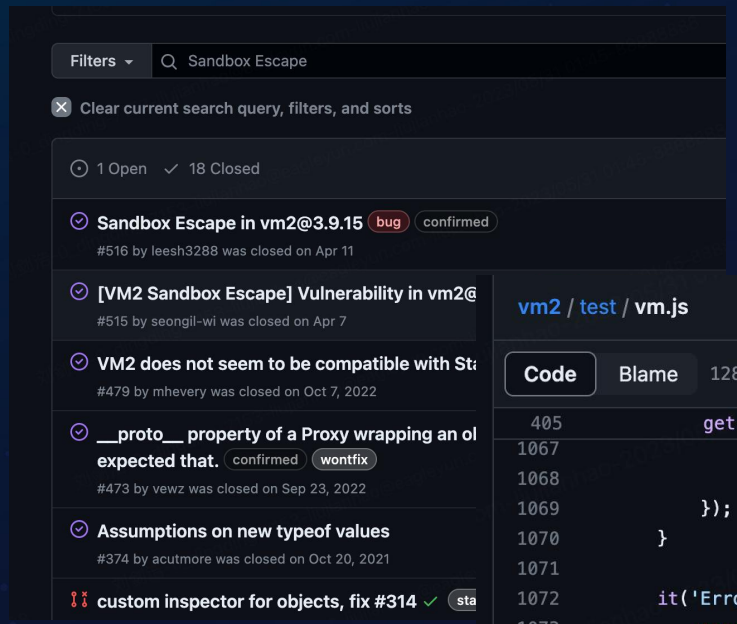
聚合函数

表函数

- file
- merge
- numbers
- url
- postgresql
- jdbc
- odbc
- hdfs
- s3
- input
- generateRandom
- cluster
- view
- null function
- dictionary function
- s3Cluster
- sqlite
- format
- mysql
- remote, remoteSecure

简单
完成
支持
或
返回
与原

第三方组件:真实案例(VM2)



vm2 / test / vm.js

Code | Blame | 1285 lines (1098 loc) · 38.2 KB

```
405             get() {
1067                const process = import('oops!').constructor.constructor('return p
1068            `), /VMError: Dynamic Import not supported/);
1069        });
1070    }
1071
1072    it('Error.prepareStackTrace attack', () => {
1073        const vm2 = new VM();
1074        const sst = vm2.run('Error.prepareStackTrace = (e,sst)=>sst;const sst = n
1075        assert.strictEqual(vm2.run('sst=>Object.getPrototypeOf(sst)')(sst), vm2.r
1076        assert.throws(()=>vm2.run('sst=>sst[0].getThis().constructor.constructor'
1077        assert.throws(()=>vm2.run(`
1078            const { set } = WeakMap.prototype;
1079            WeakMap.prototype.set = function(v) {
1080                return set.call(this, v, v);
1081            };
1082            Error.prepareStackTrace =
1083            Error.prepareStackTrace =
1084            (_, c) => c.map(c => c.getThis()).find(a => a);
1085            const { stack } = new Error();
1086            Error.prepareStackTrace = undefined;
```

第三方组件:真实案例(JDBC)


```
String CLASS_NAME = "com.mysql.jdbc.Driver";
// 连接事先配置好的恶意服务器 cobar
String URL = "jdbc:mysql://localhost:8066/dbtest?detectCustomCollations=true&autoDeserialize=true";

Class.forName(CLASS_NAME);

// getConnection 触发漏洞
Connection connection = ConnectionUtil.getJDBCConnection(CLASS_NAME, URL);
connection.close();
```

第三方组件:真实案例(Spark)

java_method

java_method(class, method[, arg1[, arg2 ..]]) - Calls a method with reflection.

Examples:

```
> SELECT java_method('java.util.UUID', 'randomUUID');
 c33fb387-8500-4bfa-81d2-6e0e3e930df2
> SELECT java_method('java.util.UUID', 'fromString', 'a5cf6c42-0c85-418f-af6c-3e4e5b1328f2');
 a5cf6c42-0c85-418f-af6c-3e4e5b1328f2
```

Since: 2.0.0

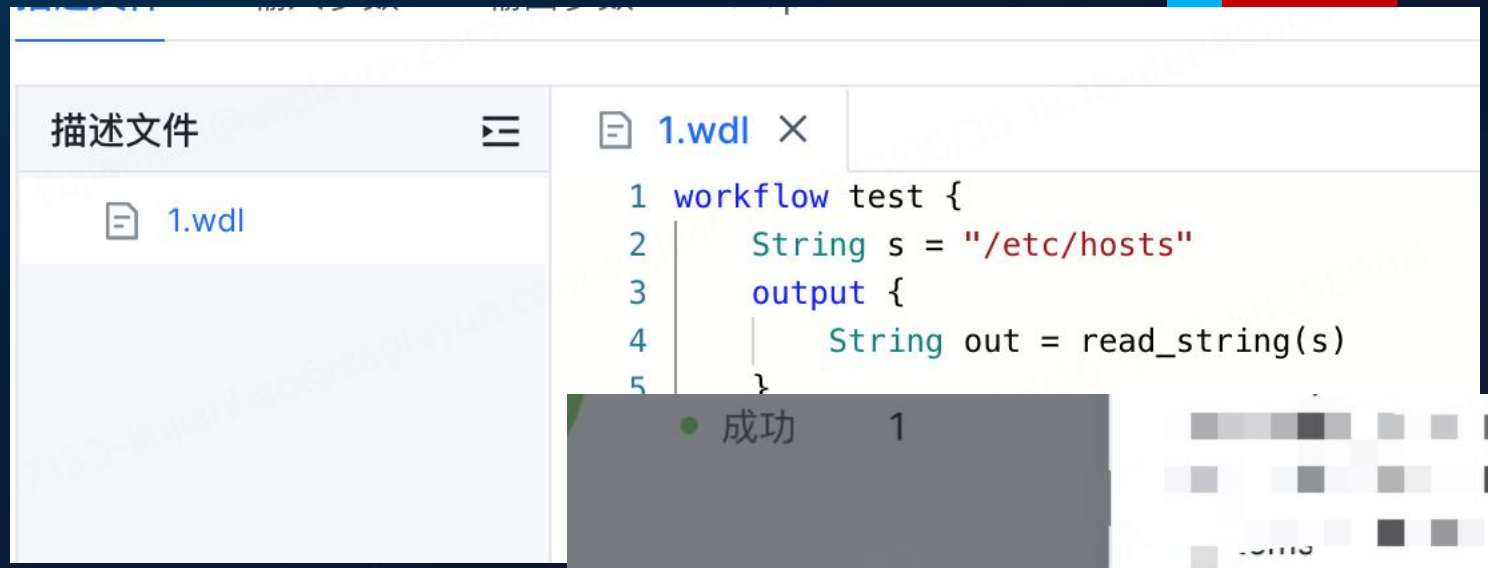
reflect

reflect(class, method[, arg1[, arg2 ..]]) - Calls a method with reflection.

java_method(java.lang.System, getenv)

{virtual_resource=, PATH=/hom

第三方组件:真实案例(Cromwell)



第三方组件:真实案例(npm)

10. 预览完成后，如果您需要将组件发布到线上，有两种方法：
    - 方法一：在组件目录下，执[redacted] publish 命令，这种方式需要用到用户名和开发者识别码。（推荐使用）
    - 方法二：在组件目录下，执行 package 命令，在组件目录外将生成一个以“组件名-版本号”命名的tar.gz压缩包，将此压缩包上传到[redacted]我的组件包中。

    说明：[redacted] publish 命令执行后，打包流程将在服务器中进行，会产生打包排队时间。如提示发布成功但产品中拉取组件报错，很可能是打包正在队列中，请耐心等待一段时间再刷新浏览器重试。

将组件用开发工具打开，主要涉及package.json和index.js两种文件的配置，其中package.json文件中设置代码请参见：

`npm install`

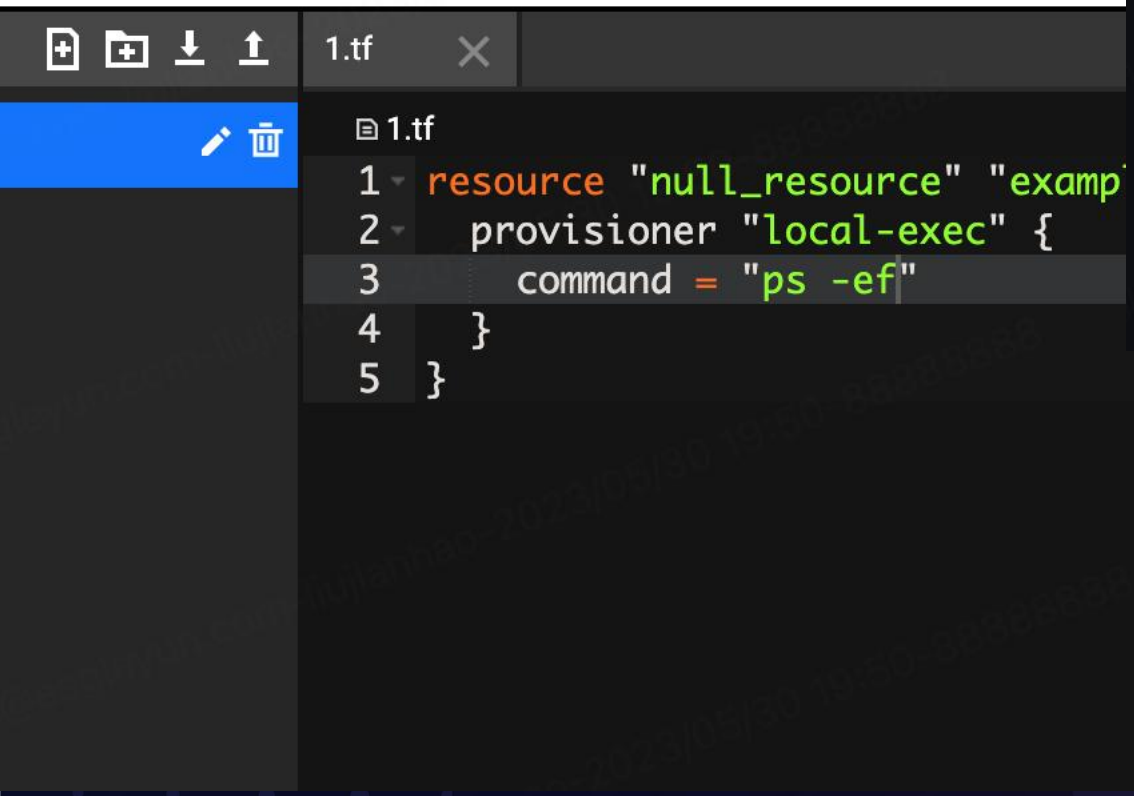
These also run when you run `npm install -g <pkg-name>`

- `preinstall`
- `install`
- `postinstall`
- `prepublish`
- `preprepare`
- `prepare`
- `postprepare`

If there is a `binding.gyp` file in the root of your package and yo

第三方组件:真实案例(Terraform)

，支持 HCL（Terraform）格式语法编写，



local-exec Provisioner

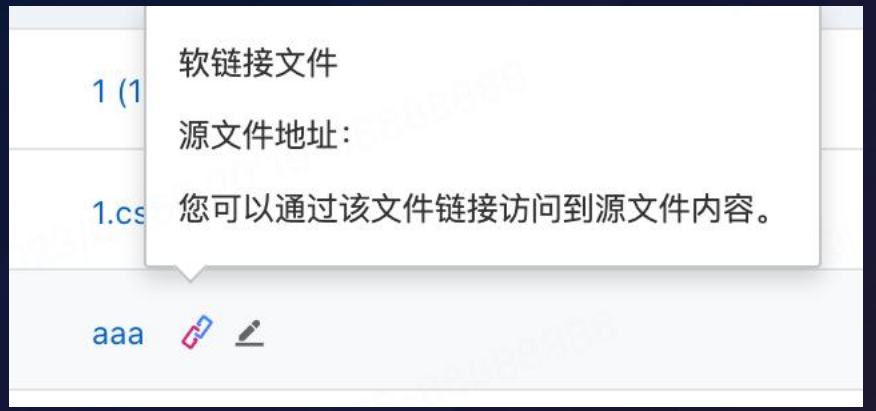
v1.4.x (latest)

The `local-exec` provisioner invokes a local executable after a resource is created. This invokes a process on the machine running Terraform, not on the resource. See the `remote-exec` provisioner to run commands on the resource.

Note that even though the resource will be fully created when the provisioner is run, there is no guarantee that it will be in an operable state - for example system services such as `sshd` may not be started yet on compute resources.

第三方PaaS服务:真实案例(对象存储)

第三方PaaS服务:真实案例(对象存储)



一些有趣的对抗

修复&&绕过

```
function safedns(host){
return "223.109.74.240" //伪代码，实际做了ssrf校验返回a记录ip
}

target= "https://www.volcengine.com/"
p = new URL(target)
target_ip =  safedns(p.host)
console.log(target.replace(p.host,target_ip))
```

https://223.109.74.240/

```
function safedns(host){
return "223.109.74.240" //伪代码，实际做了ssrf校验返回a记录ip
}

target= "https://www.volcengine.com@www.volcengine.com/"
p = new URL(target)
target_ip =  safedns(p.host)
console.log(target.replace(p.host,target_ip))
```

https://223.109.74.240@www.volcengine.com/

```
function safedns(host){
return "223.109.74.240" //伪代码，实际做了ssrf校验返回a记录ip
}

target= "https://www.你好.com/"
p = new URL(target)
target_ip =  safedns(p.host)
console.log(target.replaceAll(p.host,target_ip))
console.log(p.host)
```

https://www.你好.com/

www.xn--6qq79v.com

修复&&绕过

```
function safedns(host){
return "223.109.74.240" //伪代码，实际做了ssrf校验返回a记录ip
}

target= "https://www.qq.com/aaa"
p = new URL(target)
target_ip =  safedns(p.host)
p.host=target_ip

target = decodeURIComponent(p.toString())
console.log(target)
```

```
https://223.109.74.240/aaa
```

```
function safedns(host){
return "223.109.74.240" //伪代码，实际做了ssrf校验返回a记录ip
}

target= "https://127.0.0.1%23@qq.com/aaa"
p = new URL(target)
target_ip =  safedns(p.host)
p.host=target_ip

target = decodeURIComponent(p.toString())
console.log(target)
```

```
https://127.0.0.1#@223.109.74.240/aaa
```

修复&&绕过

基于时间的DNS缓存

将dns解析记录缓存起来，达到一定时间后过期，防止dnsrebing造成ssrf攻击

基于池的DNS缓存

将dns解析记录缓存起来，达到一定数量后丢弃最开始记录，防止dnsrebing造成ssrf攻击

修复&&绕过

Mysql JDBC场景下
1、URLQuery禁止autoDeserialize、allowLoadLocalInfile。
2、禁止?关键字防止造成Query

安全了么？

修复&&绕过

功能迁移

将危险的第三方组件
执行过程迁移出来

不信任原则

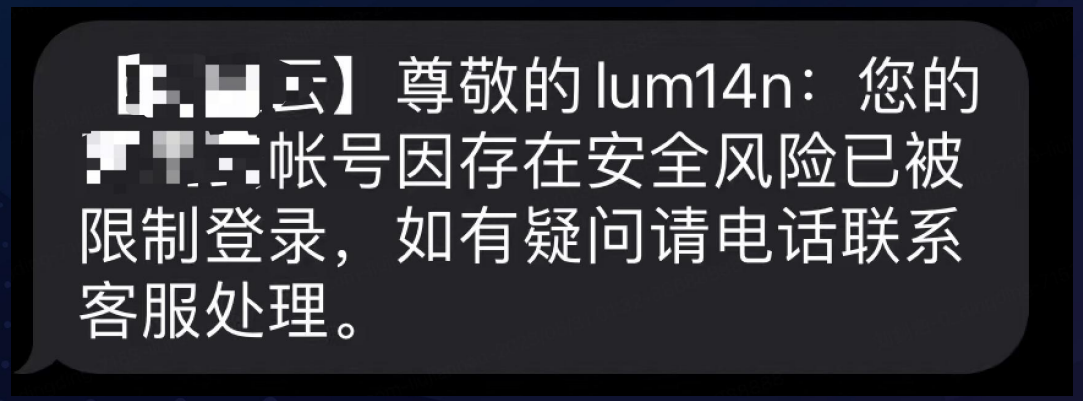
1、将功能迁移出来，利用相关paas自身的隔离，做到租户隔离，保障服务运行安全。

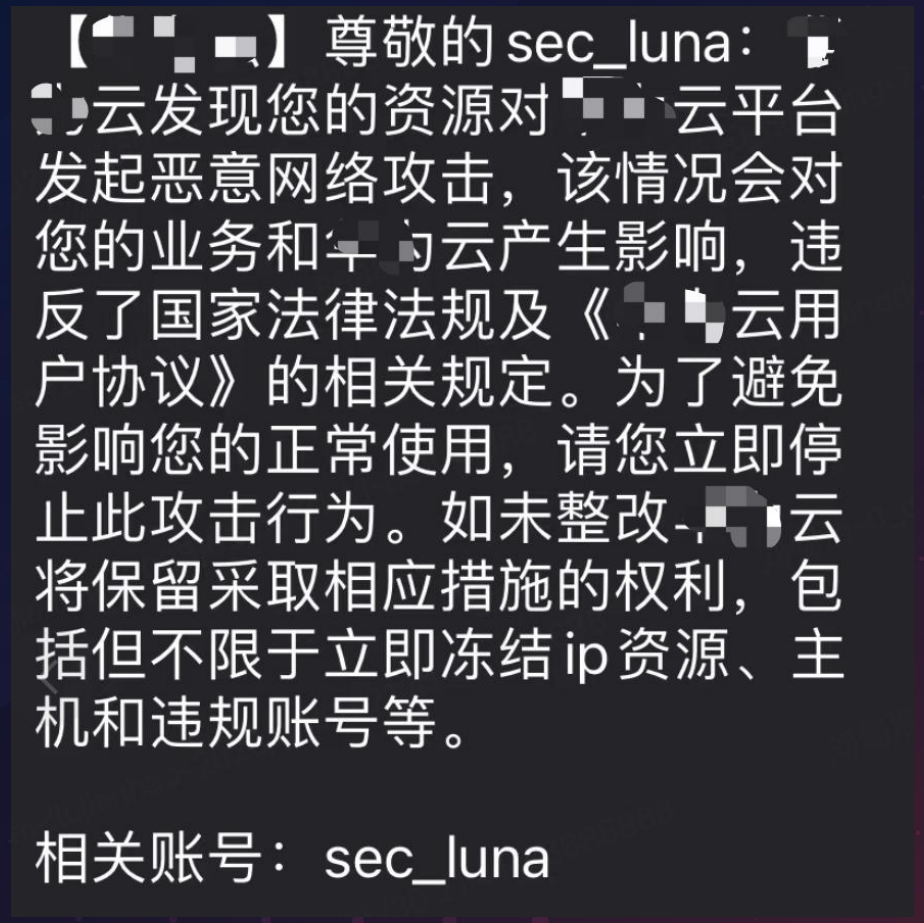
2、敏感数据审查，相关权限最小化原则，下发的相关权限密钥仅仅有服务运行必须的权限

3、网络隔离，服务运行能够依赖相关技术达到隔离效果，但是不能让租户网络能够互通，甚至是内网，metadata等。

反入侵检测


您的[redacted]帐号冻结
您的帐号存在安全风险，请联系客服处理，电话：4008013260
requestId=bc9ba43f-b20a-4ff9-98f3-e154a145ce6f


【[redacted]云】尊敬的lum14n：您的[redacted]帐号因存在安全风险已被限制登录，如有疑问请电话联系客服处理。


【[redacted]】尊敬的sec_luna：[redacted]云发现您的资源对[redacted]云平台发起恶意网络攻击，该情况会对您的业务和[redacted]云产生影响，违反了国家法律法规及《[redacted]云用户协议》的相关规定。为了避免影响您的正常使用，请您立即停止此攻击行为。如未整改-[redacted]云将保留采取相应措施的权利，包括但不限于立即冻结ip资源、主机和违规账号等。
相关账号：sec_luna

反入侵检测

WAF

1、架构层、分片延时等，考验重组能力

2、解码层，考验畸形包兼容能力

application/x-www-form-urlencoded

{"a":"payload","aaa":"a=a"}

=> Value = a"}

Key = {"a":"payload","aaa":"a

3、规则层，考验对应场景的理解对抗

RASP

1、相关RASP绕过技术

2、使用JAVA代码进行URL访问、文件读取，不涉及明显异常

动作轻，验证问题存在为主

EDR

1、不要fork进程，代码层进行探查

2、尽量行为符合业务特点

危害分析与提升

危害分析

我是不是在内网里？

我是不是在容器里？

我是不是在沙箱里？

我是不是在用户隔离的容器里？

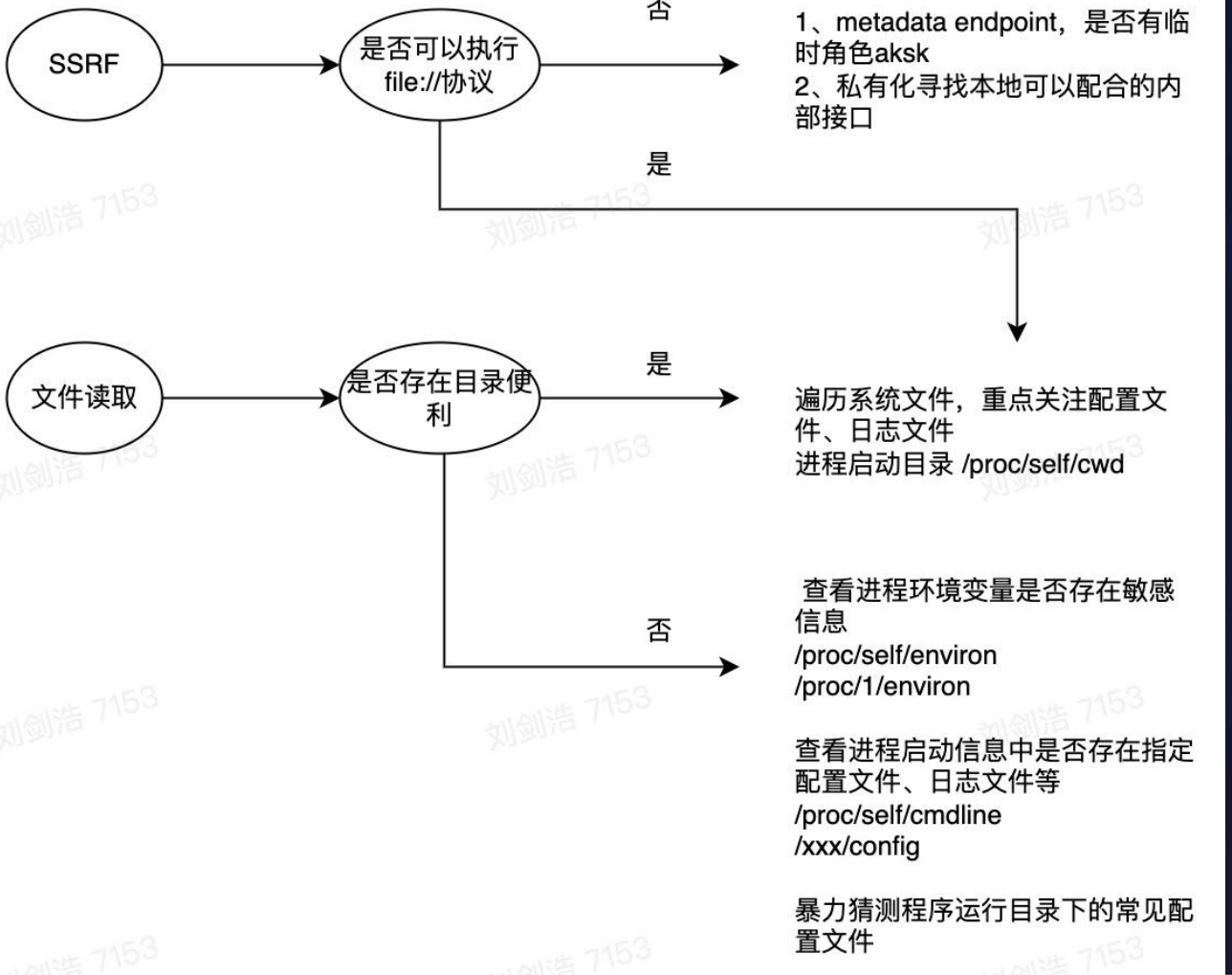
无危害忽略

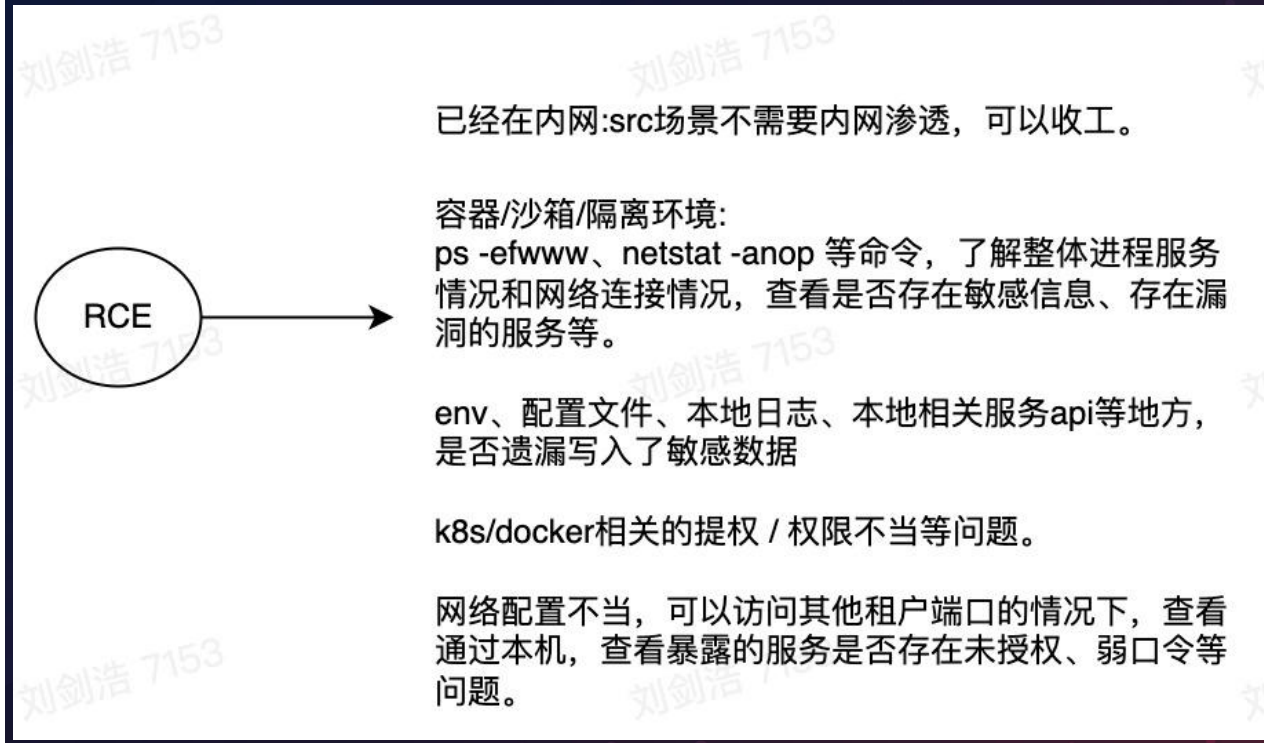
netstat －anop，查看通信的IP都是什么网段的
curl 内网的一些域名
curl metadataurl，查看instance_id和hostname
cat /etc/hosts
cat /proc/self/mountinfo
.dockerenv文件
命令是否被阉割

env命令是否有k8s相关endpoint配置，是否带有相关任务id
ps －efwww查看进程数量，是否带有相关任务id，查看进程的启动时间
cat /proc/uptime

查看系统内的一些日志文件、临时文件是否有其他租户使用的痕迹
拿一个新的账号，做同样的事情，查看是否在机器上能看到

危害提升


SSRF
是否可以执行file://协议
否
1、metadata endpoint，是否有临时角色aksk
2、私有化寻找本地可以配合的内部接口
是
文件读取
是否存在目录便利
是
遍历系统文件，重点关注配置文件、日志文件
进程启动目录 /proc/self/cwd
否
查看进程环境变量是否存在敏感信息
/proc/self/environ
/proc/1/environ
查看进程启动信息中是否存在指定配置文件、日志文件等
/proc/self/cmdline
/xxx/config
暴力猜测程序运行目录下的常见配置文件


RCE
已经在内网:src场景不需要内网渗透，可以收工。
容器/沙箱/隔离环境:
ps -efwww、netstat -anop 等命令，了解整体进程服务情况和网络连接情况，查看是否存在敏感信息、存在漏洞的服务等。
env、配置文件、本地日志、本地相关服务api等地方，是否遗漏写入了敏感数据
k8s/docker相关的提权 / 权限不当等问题。
网络配置不当，可以访问其他租户端口的情况下，查看通过本机，查看暴露的服务是否存在未授权、弱口令等问题。

危害提升:真实案例

发现任意文件读取

发现服务AKSK

提权管理员权限

存在权限控制问题

全站大量openapi服务可读/写

大事记二

文字说明文字说明

ByteDance 字节跳动 | Security 安全与风控 × 安全范儿 BYTEDANCE SECURITY

THANK YOU FOR READING

00-000000000000
XXXXXXXX@XXXX.com

白帽技术暨 BYTESRC
成长体系发布会