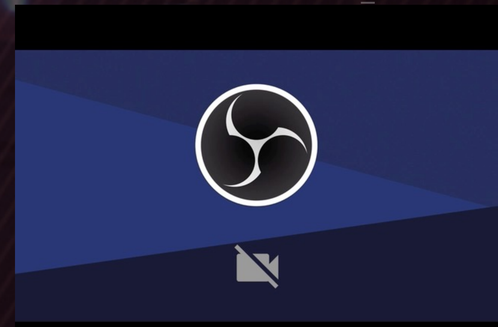
KCon

浅谈零信任环境下攻击场景

薛逸钒 腾讯企业IT安全研究员



KCon

+1

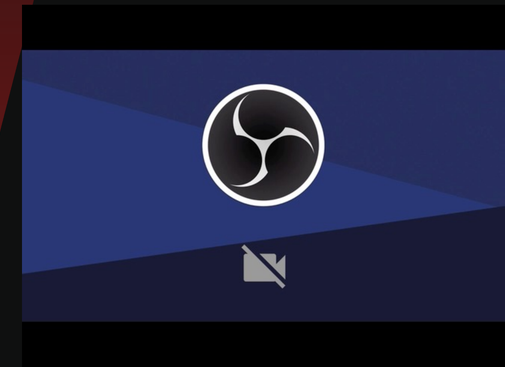
CONTENTS

目录

1 传统网络OR零信任

2 零信任的组成

3 零信任实践过程中一些风险

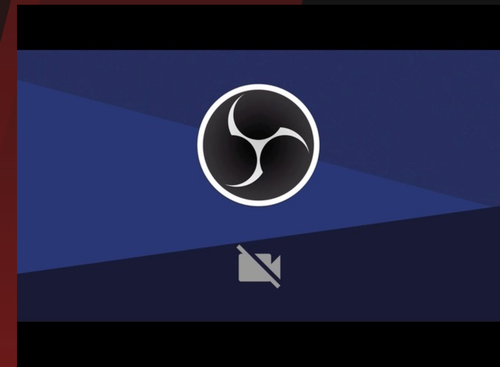


KCon

+1

01

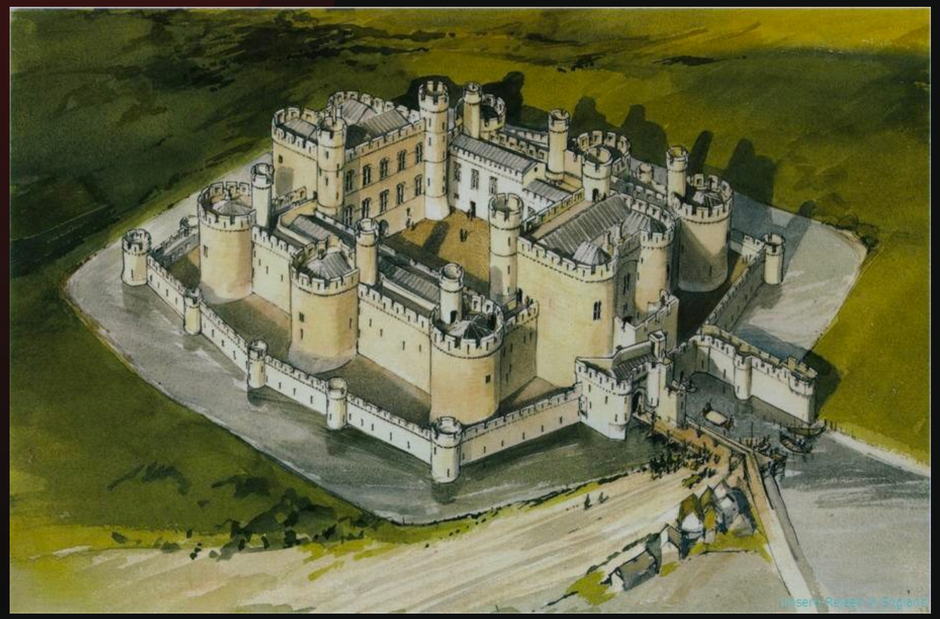
传统网络or零信任



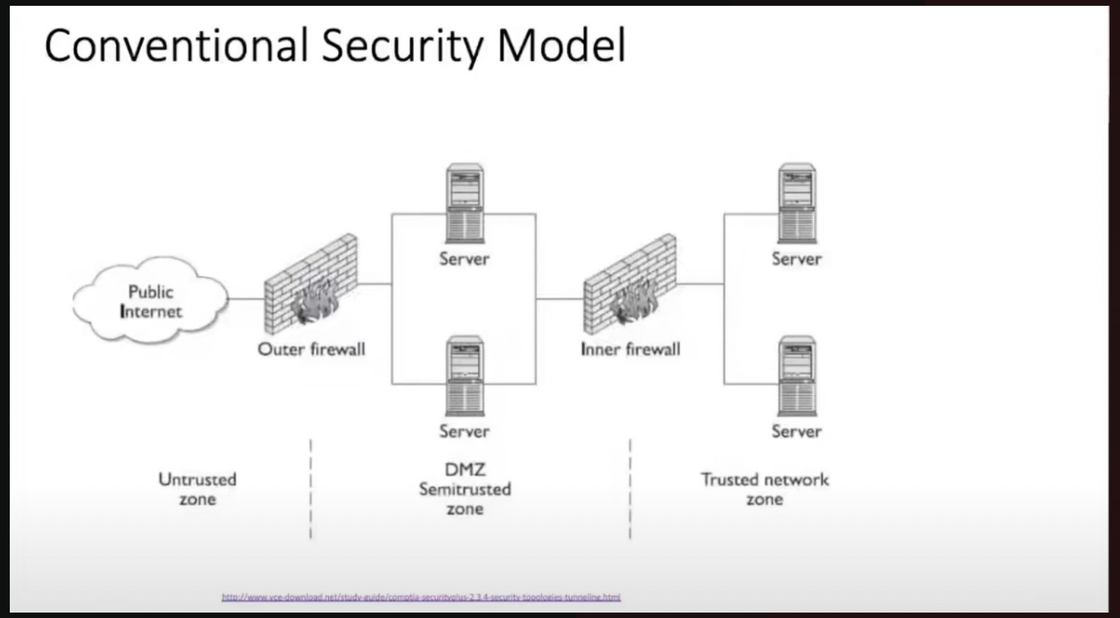
KCon

+1

传统网络架构



传统网络架构攻击路径



边界划分网络区域

突破网络边界之后，内部网络畅通无阻

依赖防火墙

获取外部人员凭证即可完全进入内网
Web攻击、社工钓鱼vpn均可直通内网

网络内部可信

依赖密码进行验证，喷洒、翻找，通用密
核心区域网络互通，并可通过扫描等手段

KCon

+1

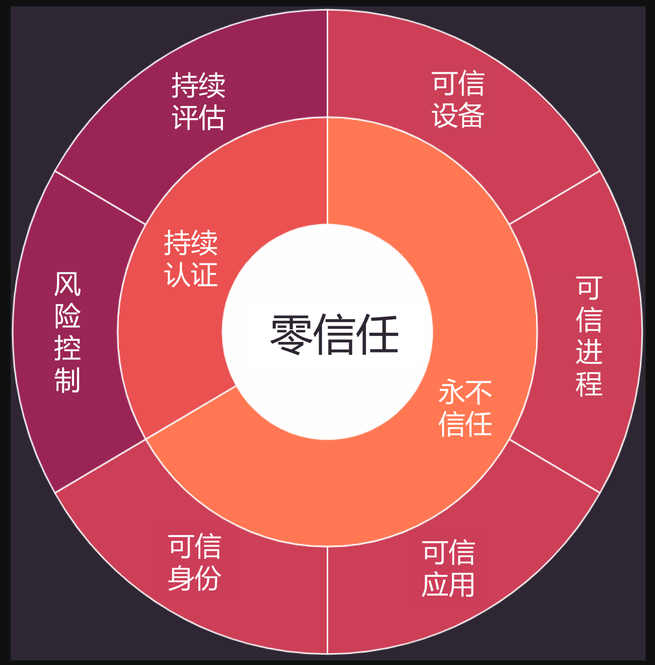
边界划分->无边界办公

异地办公

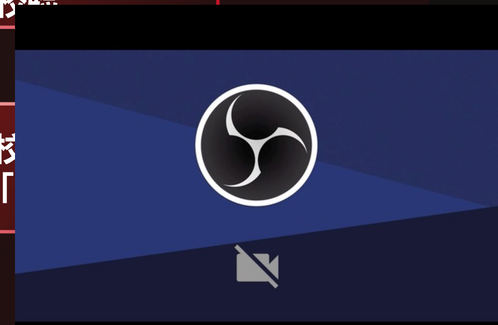
传统网络架构
无法满足需求

零信任应运而生「无边界」- 持续认证，永不信任

- 网络流量监控
- 终端安全检测
- 应急响应
- 漏洞加固



- 安全网关访问凭据校验
- 进程合法校验
- 应用安全票据校验
- 零信任agent校
  『密码」/「MFA」/「

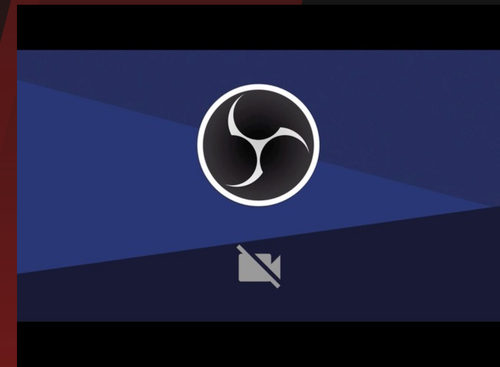


KCon

+1

02

零信任的组成



KCon

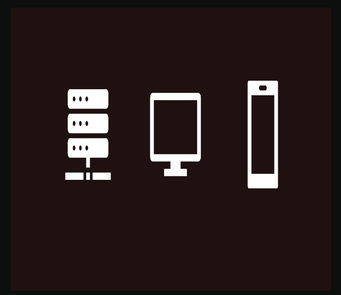
+1

零信任的落地形式-NIST架构

外部平台调用接口

用户PC接入

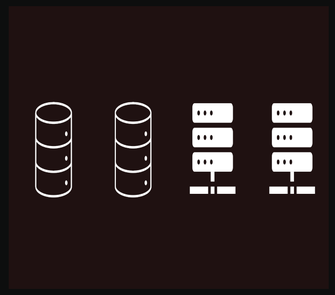
用户移动设备接入



访问主体

零信任管理

零信任安全网关



动态可信数据区

主体

终端安全能力

管理平台

安全管理能力

安全网关

网络安全能力

数据区

数据保护能力

永不信任，始终验证

主体-终端可信

- 零信任agent
- EDR
- 漏洞修复
- 环境标准化

安全网关-请求可信

- 可信进程
- 可信人员
- 可信设备
- 可信应用

管理平台-权限控制

- 控制访问权限
- 控制访问身份
- 控制访问进程
- 控制访问应用

数据区-访问控制

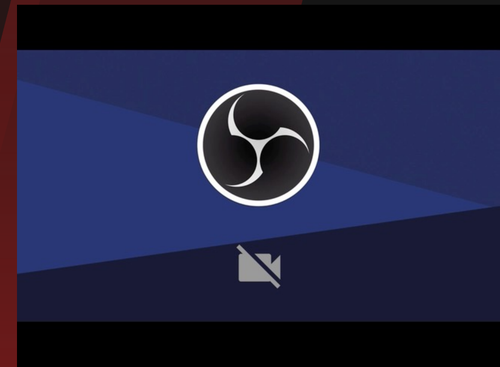
- 身份认证服务
- 二次登陆验证
- 验证访

KCon

+1

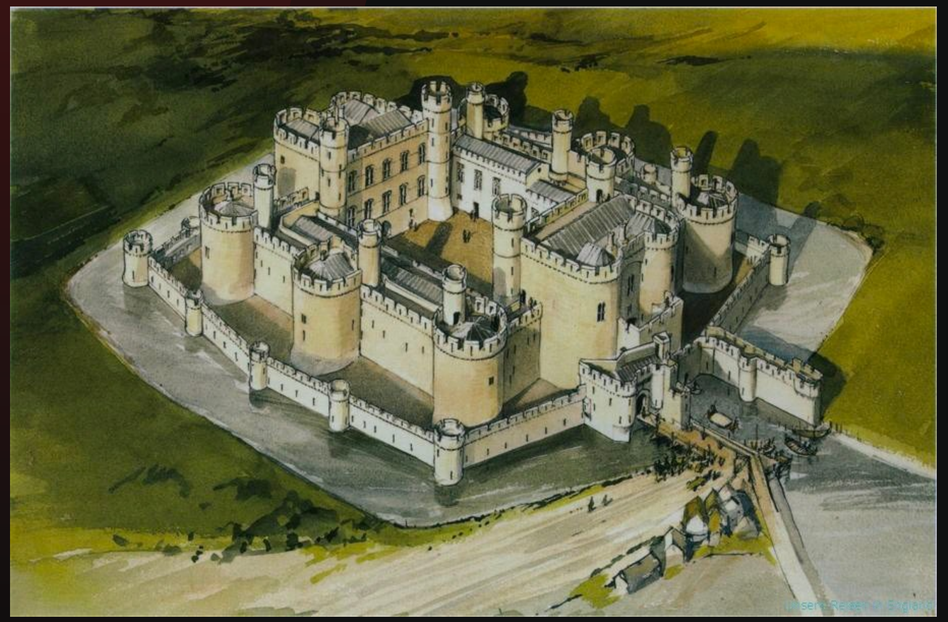
03

全面防御中找破绽

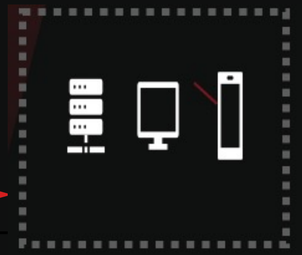


KCon

+1



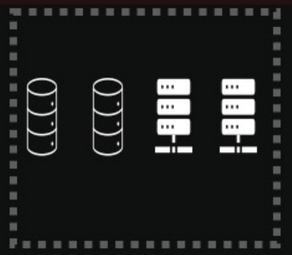
是否安全？



访问主体

零信任管理

零信任安全网关



动态可信数据区

KCon

+1

寻找转变过程中破绽

终端持续验证&监控

设备终端注册采用的方式是否安全？
验证过程中是否存在逻辑问题？
可信进程是否真的可信？
如何从根本上接管MFA认证？

零信任网关的接入

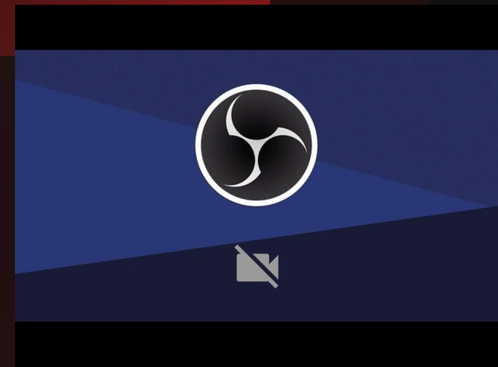
公网代理能否让我们直通内网？
凭据校验逻辑是否完全符合零信任原则？
凭据算法是否可以被伪造？

零信任agent

零信任agent管理端是否存在漏洞？
Agent是否存在漏洞？

微隔离实践

微隔离是否成了接管整个内网的突破口？



KCon

+1

终端持续验证-网络钓鱼实现构造信任设备账户接管MFA认证

⊠⊠某厂商零信任客⊠端⊠行分析，逆向出登⊠算法，通⊠⊠⊠⊠取到登⊠凭据，得到合法身份权限，⊠而接管内网⊠⊠。

账号钓鱼

- ⊠号登⊠存在二次⊠⊠：手机⊠⊠⊠/⊠⊠登⊠，凭据存在⊠⊠限制
- ⊠号登⊠存在⊠⊠校⊠，非可信⊠⊠

- 登⊠触⊠告警
- ⊠号⊠⊠信息⊠期
- 无法使用⊠号密⊠登⊠⊠⊠站点

投递木马

- ⊠⊠主体：EDR安全防⊠⊠格
- 安全网关校⊠可信⊠程，可信⊠⊠
- 内部系⊠需要MFA⊠⊠

- 木⊠被⊠⊠
- 无法⊠入内网
- ⊠入内网之后无法⊠⊠⊠⊠

KCon

+1

终端持续验证-网络钓鱼实现构造信任设备账户以及MFA认证

针对某厂商零信任客户端进行分析，逆向出登陆算法，通过钓鱼获取到登陆凭据，得到合法身份权限，进而接管内网业务。

逆向协议

注册设备

添加验证逻辑

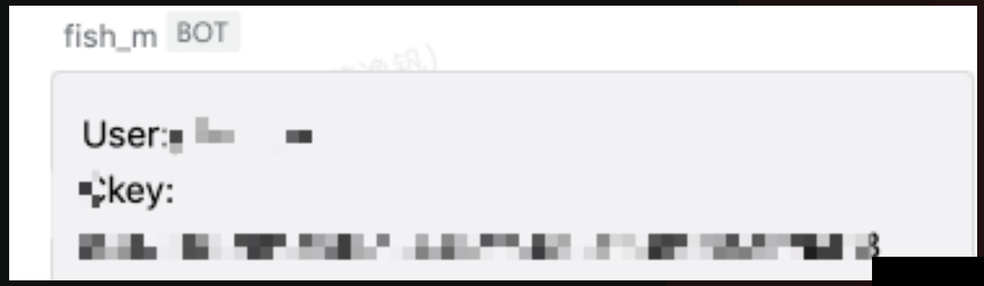
设备注册

- 注册逻辑
- 认证凭据
- 校验凭证
- 合法设备账户

二进制协议逆向

```
sub_454340.c
522   v49 = this;
523   isChecked = QAbstractButton::isChecked(*(QAbstractButton **)(this + 96));
524   sub_45E240(&isChecked);
525   v2 = *(_DWORD *)(this + 144);
526   if ( (!v2 || !*(_DWORD *)(v2 + 4) || !*(_DWORD *)(this + 148)) && QWidget::i
527   {
528     sub_4536D0();
529     v3 = sub_494D50(v40);
530     v52 = 0;
531     v4 = sub_41B5E0(v41, v3, "/v1/user/login");
532     LOBYTE(v52) = 1;
533     v5 = (const struct QUrl *)QUrl::QUrl(v42, v4, 0);
534     LOBYTE(v52) = 2;
535     QNetworkRequest::QNetworkRequest((QNetworkRequest *)v50, v5);
536     QUrl::~QUrl((QUrl *)v42);
537     QString::~QString(v41);
538     LOBYTE(v52) = 6;
539     QString::~QString(v40);
540     v6 = (const struct QString *)sub_494EB0(v39);
541     LOBYTE(v52) = 7;
542     QVariant::QVariant((QVariant *)v34, v6);
543     LOBYTE(v52) = 8;
544     QNetworkRequest::setHeader(v50, 7, v34);
545     QVariant::~QVariant((QVariant *)v34);
546     LOBYTE(v52) = 6;
547     QString::~QString(v39);
548     QVariant::QVariant((QVariant *)v33, "application/json");
549     LOBYTE(v52) = 9;
```

钓鱼提交凭证模拟用户登陆注册设备

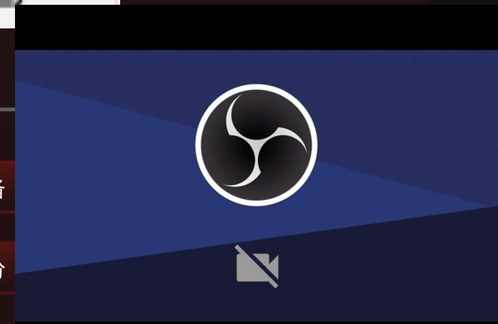


MFA

个人管理-增加双因子验证方式

完全掌控一个合法账户

- 合法设备
- 合法身份



KCon

+1

零信任网关的接入-外网获取代理凭据，伪造代理凭据进入内网

所有访问业务的通信必须经由网关验证，网关隔开了外网和内网，所有安全策略都由网关执行，如果接管网关相当于拿下整个内网的突破口。

安全网关

网关管理

http配置

日志功能

权限校验

信息数据
Mysql

网关用户信息
网关进程信息
网关接口信息

事件信息日志
Elasticsearch

用户访问日志
票据验证信息

配置数据
Redis

网关数据配置
查询用户权限

网络通信

凭据可被嗅探

网络通信未加密

日志存储

公网开放

鉴权问题

业务实现

Ticket实现逻辑

Ticket校验逻辑

路径一：获取ticket->算法可逆->伪造ticket访问

路径二：Ticket无过期限制->日志存

KCon

+1

零信任网关的接入-外网获取代理凭据，伪造代理凭据进入内网

攻击路径一：
获取ticket->加密算法可逆->伪造ticket访问

- **AES**
  - CBC padding orale
- **JWT**
  - 未做签名校验
  - 弱密钥破解
- **散列算法**
  - Hash拓展攻击
  - Sig secret破解


```
{"UserName":"测试账号","DeviceID":"xxx-xxx-TEST-xxxx-
xxxxx","AuthToken":"1D79DC81C6577791CB5E58A48394834D","pid":"xxxx","AppName":"Chrome.exe","AppVersion":
"v1","Url":"password.internal.io","ProcessName":"Google Chrome Helper","Source":""}
```


Token=Hash(Username+DeviceID+Url+ProcessName+Sig)

Sig:P@ssW0rd


```
{"UserName":"测试账号","DeviceID":"xxx-xxx-TEST-xxxx-
xxxxx","AuthToken":"DB0B7E7A64892988DEB7675DB858C5B3","pid":"xxxx","AppName":"Chrome
"v1","Url":"console.internal.io","ProcessName":"Google Chrome Helper","Source":""}
```

KCon

+1

零信任网关的接入-外网获取代理凭据，伪造代理凭据进入内网

攻击路径二：
网络空间测绘发现公网开放的零信任存储数据库，获取ticket，使用ticket公网访问内网。

KCon

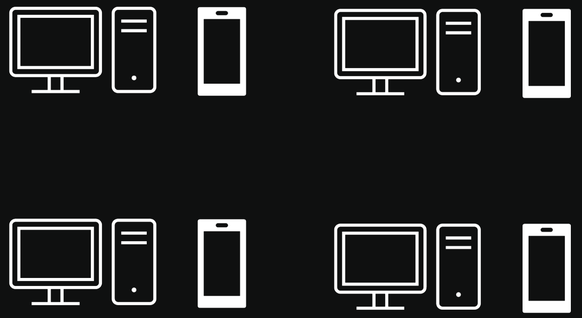
+1

零信任agent实现-复用agent服务通道，「进入」「控制」内网

为了方便监管agent客户端的行为，文件，agent管理后台一般都具有下发文件，执行命令等管理功能，开放到公网上管理端通过代码审计获取前台rce，接管内网。
在agent上，同样部署了多种服务，以供机器使用零信任网络环境。

零信任管理端

- 下发命令
- 下发补丁
- 下发软件



- WEB服务
- 票据服务
- 代理服务
- 数据库存储
- 检测服务
- Agent管理通道

Agent

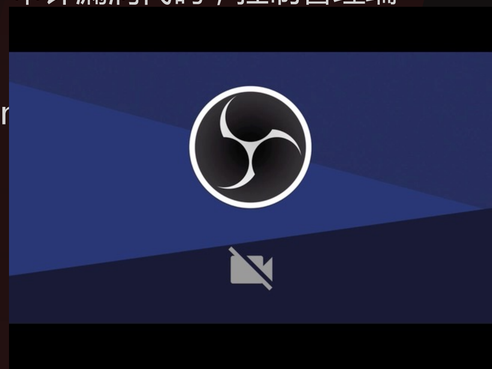
- 重要服务全网卡开放
- Agent管理通道复用

零信任管理端

- Web漏洞
- 数据库弱口令

路径一：外网测绘零信任管理平台，审计漏洞代码，控制管理端

路径二：触及Agent网络，冒用Ager



KCon

+1

零信任agent实现-复用agent服务通道，「进入」「控制」内网

路径1：零信任管理平台，审计代码漏洞，控制管理端

零信任管理端

No 0day

- 下发命令
- 下发补丁
- 下发软件
- 日志存储
- 安全策略
- 安全告警
- 网络策略

文件操作

任意文件读取

数据库操作

RCE

Shell操作

SQL注入

路径二：触及Agent网络，冒用Agent服务，进入内网，冒用身份

内网代理端口：用于应用访问内网

票据端口：用于应用申请票据

代理服务全网卡监听

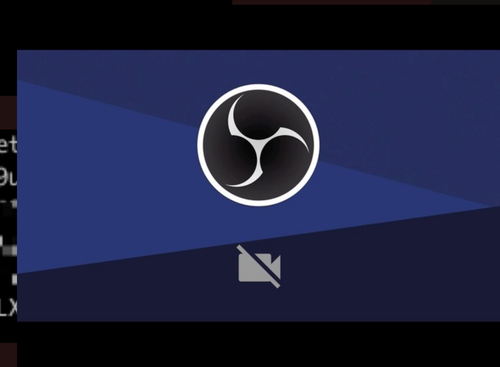
通过访问代理服务进入内网

```
➜ ~ curl -x http://172.17.6.5:8080 http://10.99.[redacted]/ -k -v
*   Trying 172.17.6.5...
* TCP_NODELAY set
* Connected to 172.17.6.5 (172.17.6.5) port 8080 (#0)
> GET http://[redacted]/ HTTP/1.1
> Host: [redacted]
> User-Agent: curl/7.54.0
> Accept: */*
> Proxy-Connection: Keep-Alive
>
< HTTP/1.1 302 Found
< x-proxy-by: [redacted]
< Set-Cookie: [redacted]/; HttpOnly
< Set-Cookie: [redacted] path=/; HttpOnly
< Content-Type: text/html; charset=utf-8
< Location: /page/query
< Date: Sat, 16 Jul 2022 03:03:26 GMT
< Content-Length: 34
< connection: close
< x-forwarded-for: [redacted]
<
<a href="/page/query">Found</a>.

* Closing connection 0
```

票据服务全网卡监听

```
➜ ~ curl http://172.20.10.2:38821/getticket
{"retcode":0,"retmsg":"SH5yJCIx7lJg8ggOku+9u
[redacted]
AwyIRJKi14ZgDGFF5uiSBAZk8iJeT1fNUd0+hUh9noLX
```

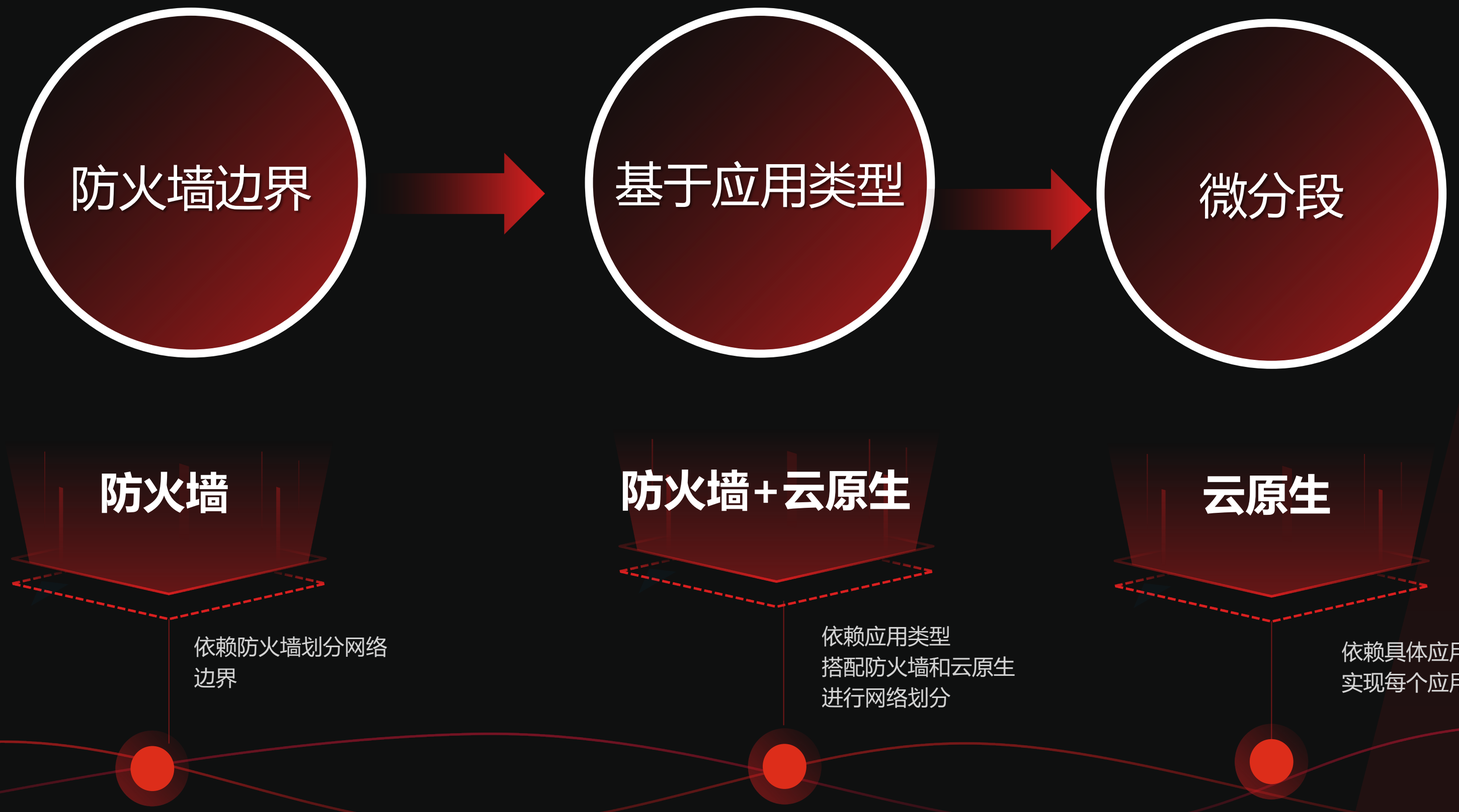


KCon

+1

微隔离实践中的粗心大意-pod接管集群

云原生架构下，逃逸风险除了漏洞攻击之外，更有管理员的错误配置导致从pods中逃逸，进而接管整个集群。

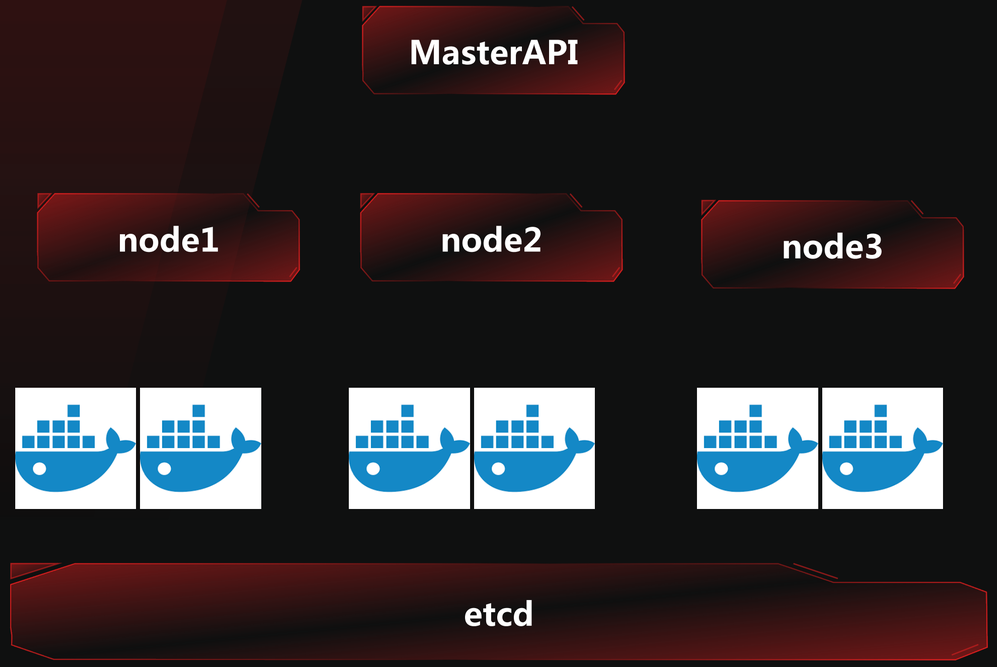


KCon

+1

微隔离实践中的粗心大意-pod接管集群

云原生架构下，逃逸风险除了漏洞攻击之外，更有管理员的错误配置导致从pods中逃逸，进而接管整个集群。



信息收集

容器基本情况
敏感配置、服务

网络探测

API server信息
Service account信息

容器逃逸

漏洞逃逸
配置错误逃逸

远程控制

反弹shell

权限提升

RBAC绕过

持久化

后门pod
影子apiserver等

https://github.com/cdk-team/CDK

KCon

+1

感谢您的观看

THANK YOU FOR YOUR WATCHING

KCon 2022 黑客大会