KCon

构建可审计的公有云安全环境

Royce Lu
Zhanglin He

+1 进阶，
护航未来

KCon

+1

CONTENTS

目录

KCon

+1

云安全基础

Cloud Security Basics

KCon

+1

云模式

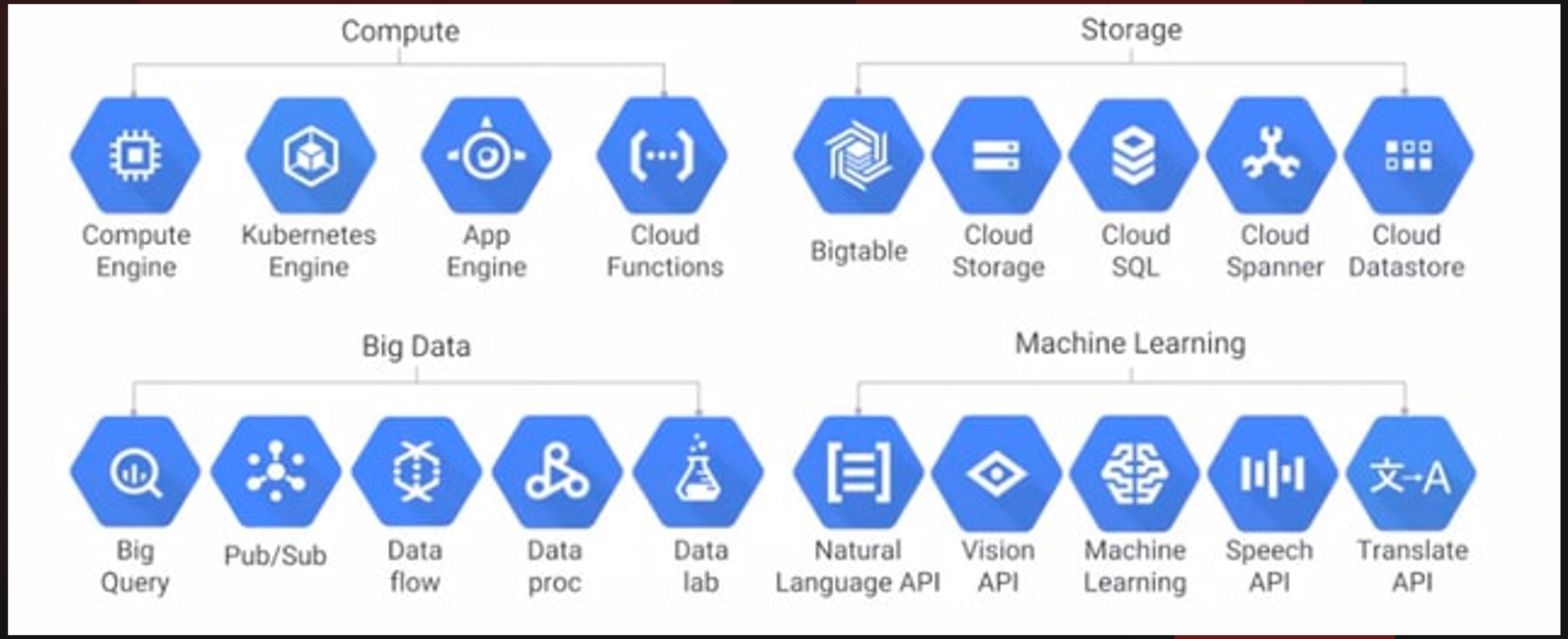

Responsibility
On-premises
IaaS
PaaS
SaaS
FaaS
CIS Controls Cloud Companion Guide
CIS Foundations Benchmarks
Data classification and accountability
Client and end-point protection
Identity and access management
Application-level controls
Network controls
Host infrastructure
Physical security
Cloud Customer
Cloud Provider

KCon

+1

Google Cloud Platform (GCP)

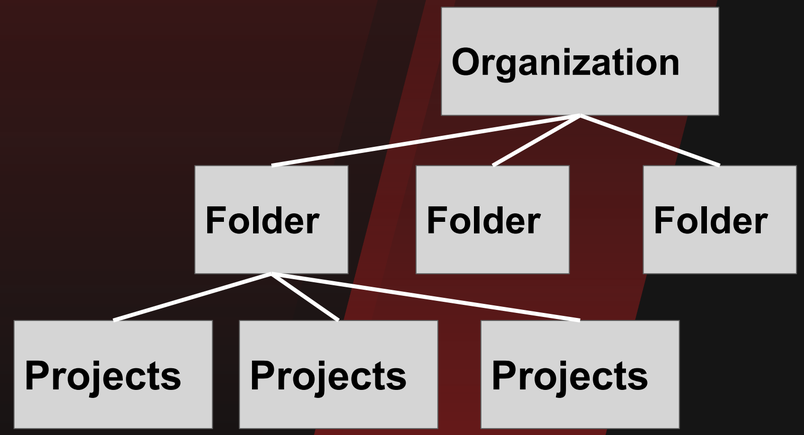
- 云计算，云存储，大数据处理以及机器学习。
- 对于设备与服务器均使用密码学签名(cryptographic signatures)，由硬件层级验证BIOS，内核与镜像的完整性
- 数据中心严格管控，只有相关人员可以进出，并以生物特征，金属探测器，摄像头等方法控制物理安全。



KCon +1

GCP资源管理：阶层架构

- Organization节点
  - 该节点管理员可管理所有子节点。避免秘密专案(shadow project)或者地下管理员(rouge admins)
- Folder节点
  - 提供部門之间的隔离层，比如访问控制策略。
- Projects节点
  - 管理资源的基础节点
- Resources节点
  - 最底层的基本元件，构成了整个google云服务
  - 四大类的各种服务，包括虚拟机，K8S丛集，云存储，BigQuery，机器学习服务，发布订阅服务…等
  - 父节点必须是Project节点

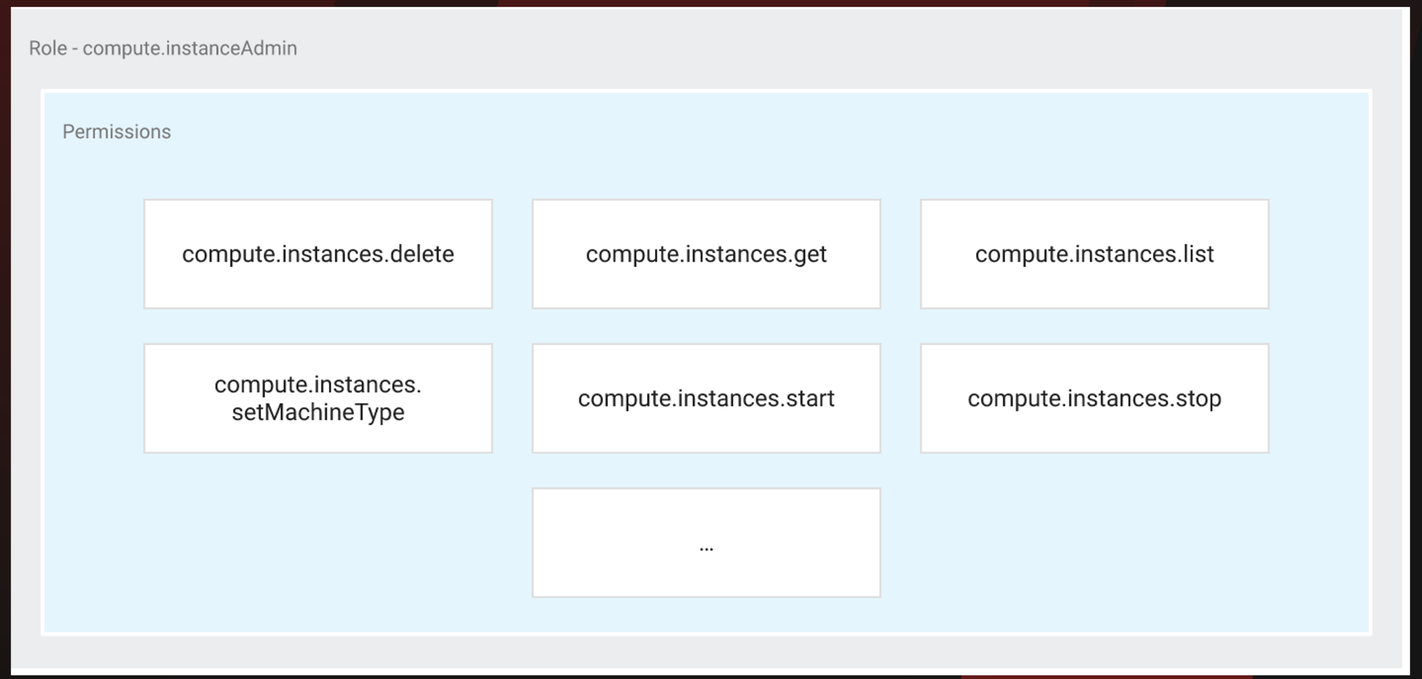


KCon

+1

GCP 权限

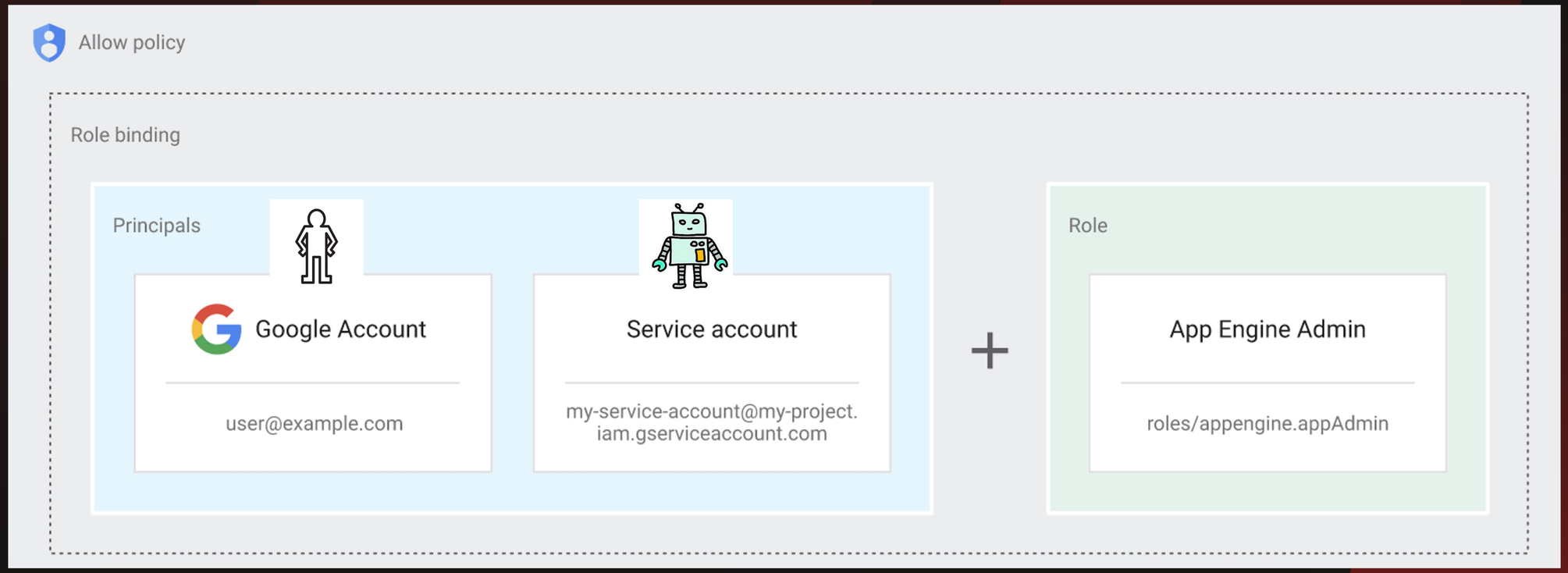
- 权限决定了一个动作能否对资源执行。权限的格式是服务.资源.动词 (service.resource.verb) 举例來说 pubsub.subscriptions.consume
- 如果要使用Pub/Sub的 topics.publish()，那么必须要有 pubsub.topics.publish 这个权限



KCon +1

GCP 身份与存取控制(Identity and Access Manamgemt)

- 委托人(Principle). 可以是用户账号(User account)，或者服务账号(Service Account)。以邮箱來代表。
- 角色(Role). 权限并不直接设定在委托人(Principle)之上，而是设定给角色(role)。角色(Role)是权限的集合。权限决定是否可以存取某个资源。当管理员把角色(role)指定给某个委托人，也把角色的权限开给了该委托人。
- 策略(Policy). 包含了委托人(principle)与角色(role)的绑定(binding)。



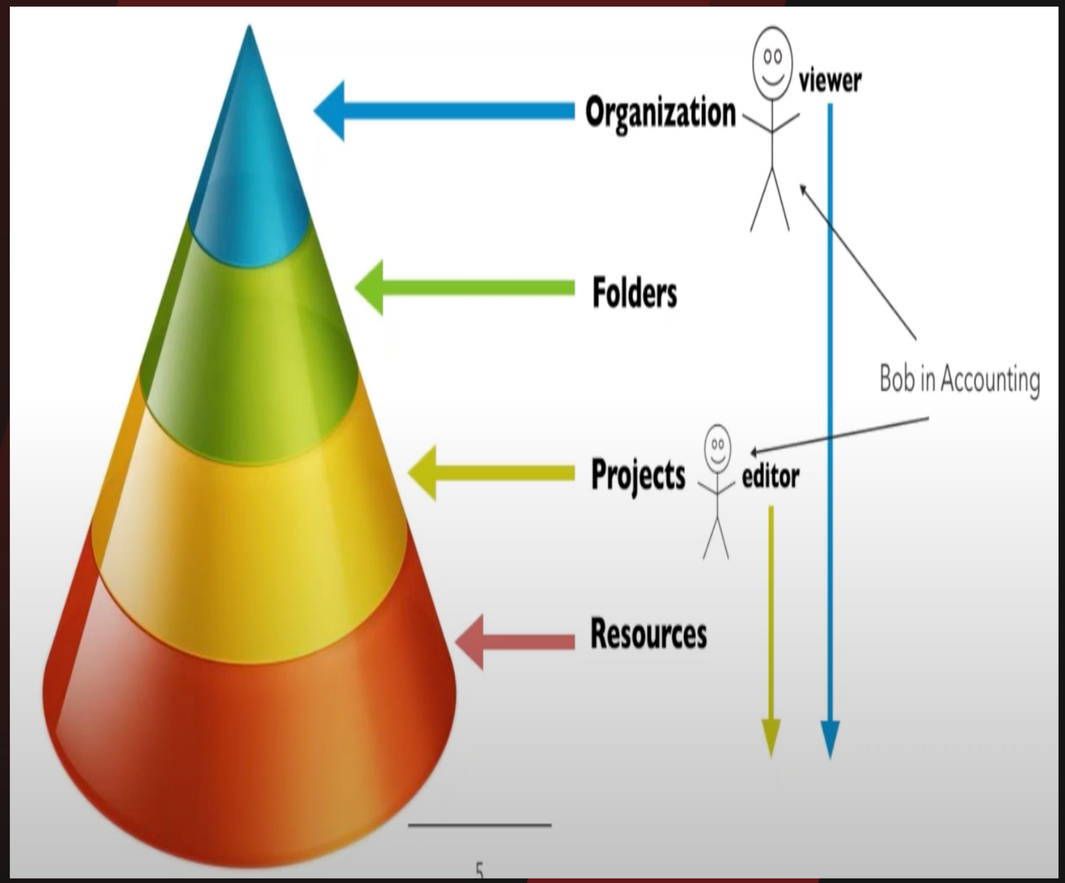
KCon +1

GCP IAM : 角色 (Role)

- 自定义角色(Customized Role)
  - 用户自订角色的权限
- 预定义角色(Predefined Role)
  - GCP预定义的细粒度，资源级别的角色，比如bigquery.admin，spanner.databaseReader，cloudsql.editor…etc
- 基本角色(Basic Role)
  - 把所有角色简化为三种 拥有者(Owner)，编辑者(Editor)与查看者(Viewer)。权限套用至Project里所有资源。
  - 并未遵守最小化权限原则
  - Project默认的服务账户(Service Account)是编辑者(Editor)。

GCP IAM与资源阶层架构

- GCP提供阶层式的资源拥有权
  - 冲突時，以权限大的为基准。Ex: Bob有Project Editor与Resource Owner权限，那么Bob对于该资源有Owner权限。
  - Project级别的权限可以存取现在，未來该Project底下的所有资源。



KCon

+1

红蓝视角

Red Team and Blue Team

KCon

+1

红队视角

红队视角 ： 渗透

KCon

+1

Post-exploitation : Enumeration，Access，Exfiltrate

- 检查scope of VM instance，以及服务账号SA accounts的权限
- 在VM instance上搜索可能的key files，secrets，scripts，keywords
- 对GCP翻箱倒柜 … enumeration all VM instances，stroage buckets，resources…etc.
- IAM提权或者横向移动
- 存取敏感讯息
- 植入后门潜伏

KCon +1

GCP 资源安全关系

资源	安全
Databases: Cloud SQL， Spanner，Bigtable， Firestore …etc	导出并且偷取数据
Storage Buckets	暴力枚举bucket名字，泄漏机密资料
Cloud Key Management Services	利用可存取的密钥解密 (本地脚本，bash history可能有KMS API参数如密钥名)
Custom image	检视是否有漏洞，设定错误或机密资讯
Cloud Git	泄漏或植入代码，找寻安全漏洞
Write/Read stackdriver logging	记录网路流向，syslog，每台机器执行的指令...etc. 修改躲避检测
App Engine， Cloud Functions， Cloud Run (Serverless)	代码存取的环境变量可能指向key file 或者 password

KCon

+1

常见的不安全设定

- 为了方便，给默认的服务账户(Service Account)添加太多权限
- 事实上，默认服务账户的权限是Project level editor，已经可以利用iam.ServiceAccount.ActAs以其他服务账户的权限行动，进而提权
- 服务账户能够存取其他Project的资源，造成横向移动。而这些资源通常是比较重要的资源，才会需要跨专案分享

KCon

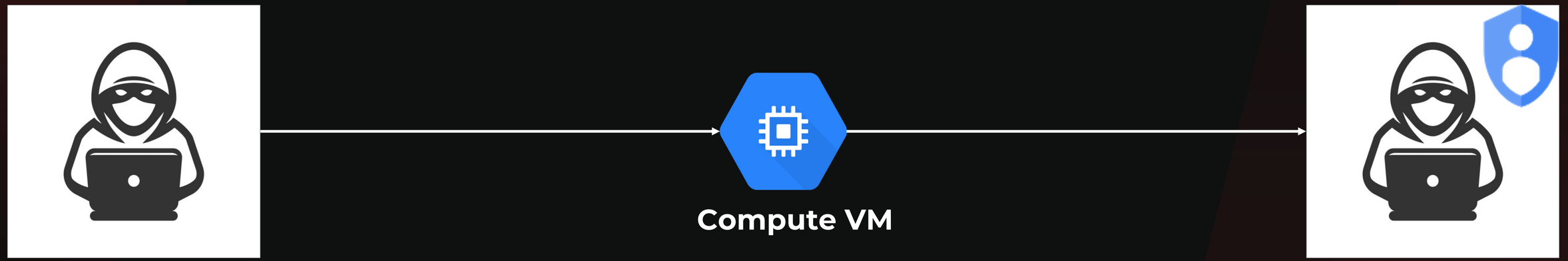
+1

提权方法一：获得token或者使用key file

KCon

+1

提权方法二：开新机器，附加高权限服务账号



Attacker

iam.serviceaccounts.ActAs

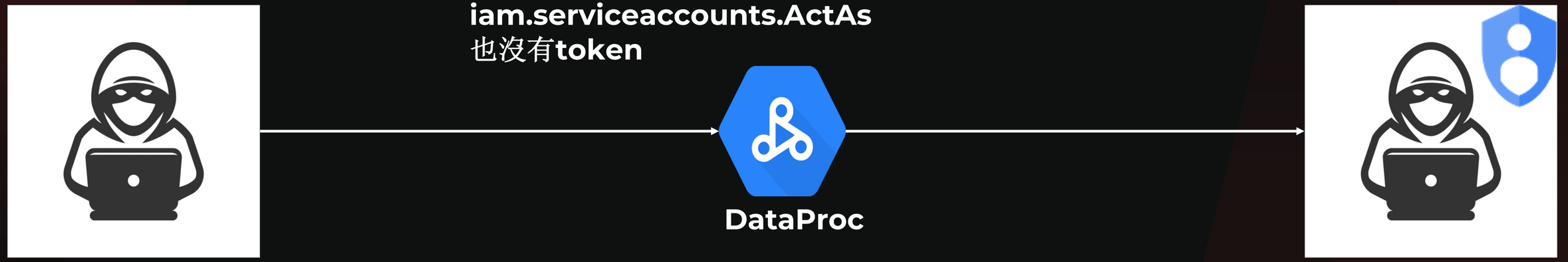
Project Owner

KCon

+1

提权方法三：启动服务(DataProc/Composer…etc)

不需要
iam.serviceaccounts.ActAs
也沒有**token**



DataProc.cluster.create

Default Compute Service Account

KCon

+1

安全与方便的两难

- 很难做到绝对安全的配置
- 公有云庞大且复杂的机制，设计上的不完美。
- 学习的成本，新进人员的训练。
- 为求方便而导致的权限问题与配置问题
- 所以，建议使用自动化系统去监控异常，补足安全上未知的缺漏。

KCon

+1

蓝队视角


蓝队视角
Blue Team
减小攻击面
身份访问控制
资源访问控制
监控和记录
公开接口
权限
API访问范围
网络
存储
实时监控

KCon

+1

安全审计框架

信息收集

- 资源
- IAM
- 事件
- 运行时环境

信息处理

- 数据标准化
- 通用字段
- 基于数据类型分类

威胁分析

- 开源引擎
- 基于Rego的规则
- 审计报告

KCon

+1

信息收集模块

信息收集模块

静态信息收集：
- API
- 主动扫描

- 资源
- IAM

动态信息收集
- 记录
- 动态探针

- 事件
- 运行时环境

KCon

+1

动态探针

Runtime Probe

运行时环境

- 正在访问的数据库和云存储
- 敏感信息
- 进程和服务

授权回溯

利用Terminal回溯实际用户。在GCP的记录里，不会记录使用服务账户进行API交互的实际用户信息。

枚举检测

在GCP的虚拟机中进行API访问时，会先检查本机是否可以访问特定的API，再由API服务端判断用户是否有该API的访问权限。枚举本机可以访问的API不会出现在记录里。

KCon

+1

03

安全建议

Suggestion and Best Practice

KCon

+1

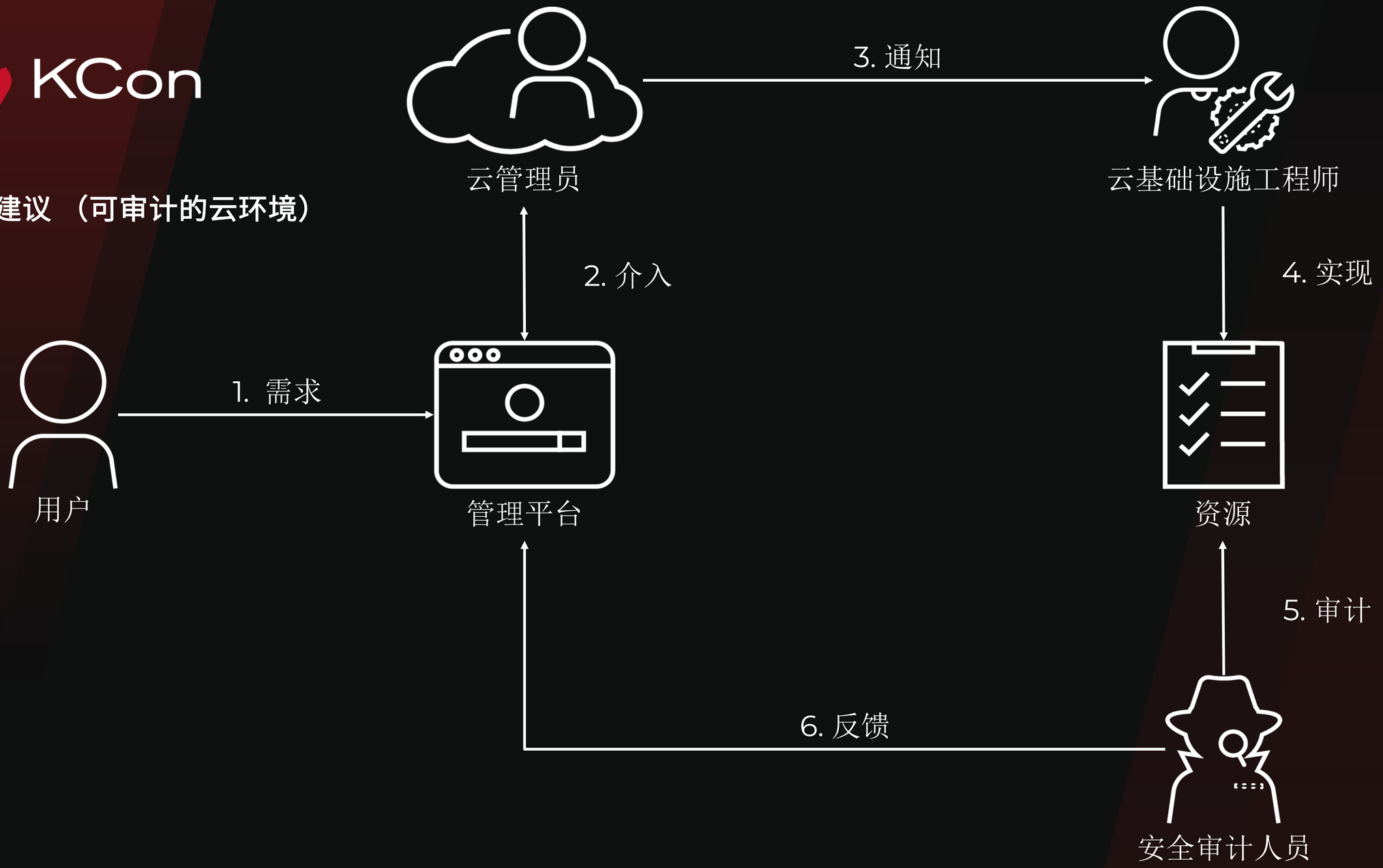
安全建议 （可审计的云环境）

- 一般情况下，安全审计只能找到有明显问题的资源
  - 防火墙规则，允许了太多的IP/端口/协议
  - Service Account 拥有了太高的权限，比如 role/editor
  - 对于特定资源，不同部门的人同时拥有admin权限
- 如果创建资源的时候，能明确的申明资源的用途/需要的权限

KCon

+1

安全建议 （可审计的云环境）



KCon +1

安全建议 （可审计的云环境）

- 用户需求：我需要从API服务器下载数据。
- 云管理员介入，明确需求。
  - 用户有API服务器的访问权限。
  - 用户使用VM27来获取数据。
  - API服务器只接受到443端口的TCP链接。
- 云管理员总结用户需求，建议云基础工程师创建防火墙规则： “Allow-VM27-APIServer-TCP-443”
- 云基础工程师实现这个规则：

```
{
        Action: “Allow”,
        Source: “10.0.0.27”， # IP of VM27
        Destination: “10.0.2.111”， # IP of API Server
        Protocol: “TCP”,
        Port: 443,
        …
}
```

- 安全审计看到这一条防火墙规则时，通过规则名和规则实现，可以判断这个防火墙规则在满足用户需求的同时，没有授予任何多余的权限。

KCon

+1

安全建议: 细分责任

- 避免在单一账号被攻陷的情况下不会有太大的损失。
  - 单一账号不应该同时拥有Cloud KMS管理员权限和Cloud KMS加解密(encrypt, decrypt)权限。
- Owner权限和editor权限都同时拥有可以产生问题的权限，必须限制在很少的用户上。

KCon

+1

安全建议：最小化攻击面

- 推荐对每个虚拟机，实用特定的SSH key，而不是在整个Project里共享SSH key。
- 虚拟机允许连接串口(Serial Port)是一个非常危险的选项。如果虚拟机允许通过串口连接，那在攻击者可以绕开防火墙限制，直接连接这台虚拟机。
- 比起给予每个虚拟机一个公网IP，使用负载均衡可以有效的减少攻击面。
- 在防火墙的准许清单规则中，使用准确的IP地址，可以有效的减少攻击面。

KCon

+1

安全建议：关于API key的管理

- API Key通常可以在客户端，或者存放API key的设备中被访问，可能更容易被发现和盗用。
- API Key在使用的过程中，往往不能辨识使用者身份或者请求来源。
- 为了防止API key被用来攻击，可以限制只能调用指定的API。
- 推荐定期更新API key
- 推荐使用标准认证方式(比如OAuth)，取代API key

KCon

+1

安全建议(用戶)

- 使用云厂商推荐的安全机制
  - 使用IAM为主要的授权方式
  - 使用 osLogin，2FA
- 最小权限原则
- 细分责任，使用特定的账户完成特定的任务
- 详细的Log
- 最小化攻击面
- Key的使用规范
- 构建可审计的云环境

KCon

+1

安全建议（服务商）

- 降低权限控制/访问控制的复杂程度(学习曲线, GCS的两套安全机制, VM的Scope …etc)
- 提供可以追溯到真实用户的API请求
  - 在GCP上，目前并不能提供到使用服务账户的进行API请求的真实用户。
- 记录失败的API请求
  - 用户提权的过程中，往往会枚举当前服务账户的权限
- 自动检测服务账户的权限
  - 拥有的权限 VS 实际使用到的权限
  - 高权限用户
  - 宽松的访问控制 (开放太多的ports)

KCon

+1

感谢您的观看

THANK YOU FOR YOUR WATCHING

KCon 2022 黑客大会