**IVRE安装**

**使用Docker方式安装**

$ mkdir ivre

$ cd ivre

//拉取Docker镜像

$ mkdir -m 1777 var\_{lib,log}\_{mongodb,neo4j} ivre-share

$ wget -q https://ivre.rocks/Vagrantfile

//启动Docker

$ vagrant up --no-parallel

//进入Docker - ivreclient shell

$ docker exec -it ivreclient bash

root@e809cb41cb9a:/#

//查看ivre命令帮助

root@e809cb41cb9a:/#ivre --help

安装脚本会在ivre目录中创建几个新的文件夹，这些文件夹用于挂载到Docker容器中存放数据库等信息。ivre-share会挂载容器ivreclient的根目录下，之后的扫描操作我们也在这个目录下执行，方便主机和容器间的文件交互。

**数据库初始化**

第一次使用时，对4个数据库进行初始化操作（使用过程中，如需完全重新导入数据，也要对相应数据库做初始化设置）：

$ ivre scancli --init

$ ivre ipinfo --init

$ ivre ipdata --init

$ ivre runscansagentdb --init

**获取IP位置数据**

ipdata数据包含了IP对应的地理位置信息。

$ ivre ipdata --download

$ ivre ipdata --import-all --no-update-passive-db

**安装缺失Python模块**

执行ivre命令后，提示部分Python模块缺失，下面进行这部分模块的安装：

apt-get update

apt-get install python-pip

//Python module PIL: 4.3.0: missing

pip install Pillow

//Python module krbV: missing

apt-get install libkrb5-dev

pip install python-krbv

//Python module MySQLdb: missing

apt-get install python-mysqldb

**IVRE使用**

**Flow analysis**

网络数据流的图形展示，这个比较炫酷，简单介绍下。先在主机上使用Wireshark抓一段时间包，数据包保存为pcap格式，命名为test.pcap，放入IVRE的共享文件夹ivre-share中，执行命令处理数据包：

$ cd /ivre-share

//使用Bro对数据包进行处理，完成后再当前文件夹生成.log文件

$ bro -r test.cap

//初始化flowcli数据库

$ ivre flowcli --init

//将.log文件导入数据库

$ ivre bro2db \*.log

查看导入的流量统计数据

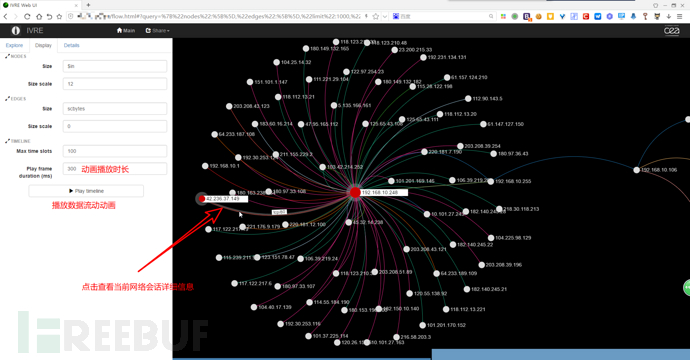
$ ivre flowcli --count

585 clients

1259 servers

3629 flows

打开浏览器，输入<http://your-host-ip/flow.html> 查看网络流量动画。

[](http://image.3001.net/images/20171125/15116099099721.png)

**主动侦查**

主动侦查部分是重点介绍的内容，通常环境下，企业内网环境的网络流量不可能完全经过安全测试服务器，需要了解内网全部服务器的端口、服务情况，需要使用Nmap、Masscan等工具进行主动探测扫描。

**使用Nmap扫描**

随机扫描1000个IP

进入Docker – ivreclient shell后，执行：

$ cd /ivre-share

$ ivre runscans --routable --limit 1000 --country CN --output=XMLFork

这条命令会执行一个随机扫描，扫描1000个中国地区的IP，默认开启30个Nmap并行进程。使用ivre help runscans查看扫描相关的详细帮助内容。扫描后在/ivre-share目录下生成扫描结果文件，包含端口开放、服务信息、网页截图等。

扫描后清理缓存文件：

rm -fr /ivre-share/scans/COUNTRY-CN/ current/

导\*\*入扫描结果\*\*

扫描完成后，将扫描结果导入数据库：

$ cd /ivre-share

$ ivre scan2db -c ROUTABLE-CN-001 -s Parrot -r /ivre-share/scans/COUNTRY-CN/up/

这里 ROUTABLE-CN-001 是这次扫描结果的分类名称，MySource是扫描来源名称，方便指挥对扫描结果的分类搜索。

除了在本地执行扫描工作外，IVRE还可以设置多个扫描在[代理服务器](https://github.com/cea-sec/ivre/blob/master/doc/AGENT.md)，在主服务器上执行ivre {runscansagent|runscansagentdb}命令进行代理服务器的扫描任务下发和结果导入。

命令执行后提示xxx results imported，则导入成功。

指定扫描IP目标

通过ivre help runscans命令查看IVRE的帮助文档，找到两个指定扫描IP的参数，--range和--network，--range指定扫描IP的起止范围，--network参数可指定一个网段。例：

$ ivre runscans --network 45.32.14.0/24 --output=XMLFork

使用--output=CommandLine 参数来看下当前执行的完整命令：

root@67d2aa11797f:/ivre-share# ivre runscans --network 45.32.14.0/24 --output CommandLine

Command line to run a scan with template default

nmap -A -PS -PE -sS -vv --host-timeout 15m --script-timeout 2m --script '(default or discovery or auth) and not (broadcast or brute or dos or exploit or external or fuzzer or intrusive)'

使用Nmap模板扫描

--nmap-template参数可指定Nmap扫描模板，在/etc/ivre.conf中添加模板。

Nmap模板参考:

vi /etc/ivre.conf

//创建一个扫描1-65535端口，名称为`full-port`的Nmap扫描模板，在文件中添加以下内容（复制default模板并修改部分参数）

NMAP\_SCAN\_TEMPLATES["full-ports"] = NMAP\_SCAN\_TEMPLATES["default"].copy()

NMAP\_SCAN\_TEMPLATES["full-ports"]["ports"] = "1-65535"

NMAP\_SCAN\_TEMPLATES["full-ports"]["scripts\_categories"] = ['(default or discovery or auth) and not (broadcast or brute or dos or exploit or external or fuzzer or intrusive)']

NMAP\_SCAN\_TEMPLATES["full-ports"]["scripts\_exclude"] = []

添加模板参数后后再看看扫描命令：

root@67d2aa11797f:/ivre-share# ivre runscans --network 45.32.14.0/24 --output CommandLine --nmap-template full-port

Command line to run a scan with template full-port

nmap -A -PS -PE -sS -vv -p 1-65535 --host-timeout 15m --script-timeout 2m --script '(default or discovery or auth) and not (broadcast or brute or dos or exploit or external or fuzzer or intrusive)'

相比之前的命令，增加了-p 1-65535参数，执行全端口扫描。

模板中可以设置的其它参数，我在网上没找到详细的官方介绍文档，可以在IVRE的python文件中查找：/usr/local/lib/python2.7/dist-packages/ivre/nmapopt.py通过模板，你可以定制你的特定扫描需求。

**扫描结果使用**

执行扫描并导入数据库后，有三种方式使用这些内容：

* ivre scancli命令行工具
* Python模块ivre.db
* IVRE Web界面

CLI: ivre scancli

例：使用命令行查看所有开放了22端口的主机:

$ ivre scancli --port 22

See the output of .

使用ivre help scancli查看详细的帮助信息。

Python 模块

例：

$ python

>>> from ivre.db import db

>>> db.nmap.get(db.nmap.flt\_empty)[0]

在Python shell执行help(db.nmap)查看模块的帮助信息。

Web界面

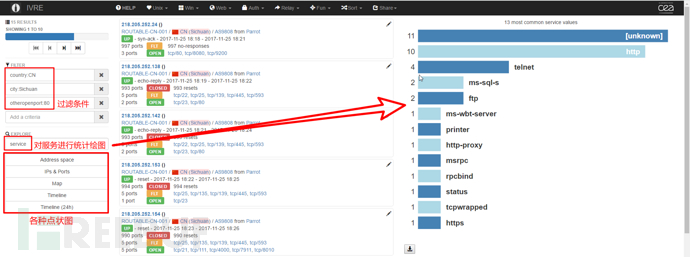
浏览器输入 [http://your-host-ip](http://your-host-ip/) 访问IVRE Web界面，

**IVRE WEB界面**

IVRE提供了WEB界面方便直观的展示主动扫描（Nmap）结果，通过关键字过滤搜索你需要的内容。

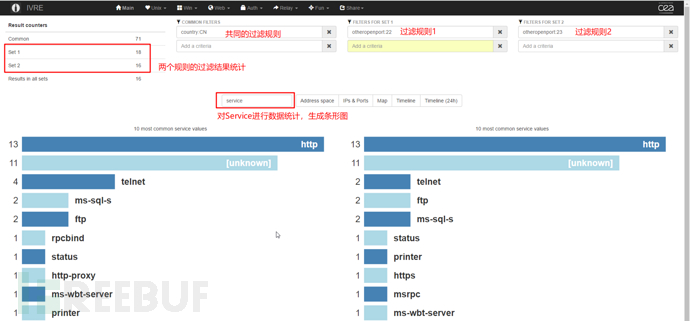
网页菜单栏点击HTLP，可以查看搜索过滤的帮助信息，其它菜单项则是在过滤器中快速添加过滤条件。左侧显示RESULTS、FILTER、EXPLORE三部分内容、

* RESULTS：当前过滤规则的结果总数、显示页面控制
* FILTER：过滤规则，可以填写服务过滤、结果排序、页面展示内容三类规则，可以多个规则组合使用
  + 一些常用的过滤示例：
  + country:CN       //过滤国家为中国
  + city:Chengdu    //过滤城市为成都
  + openport           //列出至少开放1个端口的IP
  + otheropenport:22     //开放22端口
  + service:telnet           //开放Telnet服务
  + !host:218.205.252.70       //不显示218.205.252.70，部分参数前面加！表示否定
  + display:screenshot           //结果中显示截图
* EXPLORE：将当前的过滤结果生成图形化展示

[](http://image.3001.net/images/20171125/15116102053251.png)

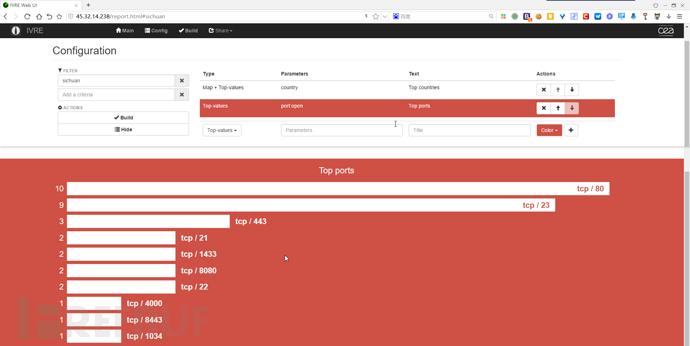
Compare Graphs

菜单栏Share→Compare Graphs，图形比较页面，可以设定两个不同的过滤规则，统计过滤后的数据，指定需统计的内容如服务（service）、端口（port）或产品（product）等，生成两组作比较的条形图。

[](http://image.3001.net/images/20171125/15116102461359.png)

Report

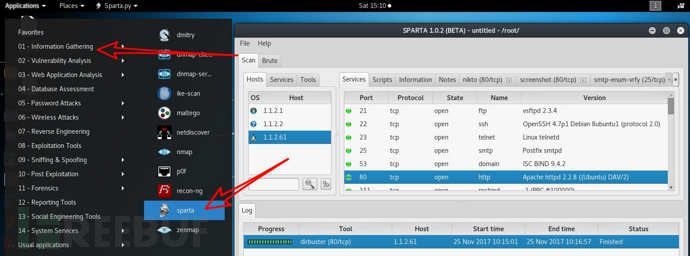
菜单栏Share→Report，进入Report页面，对开发端口、服务、IP地区等进行Top 10统计，生成条形图，需要写统计报告的同学，这个功能比较实用。

[](http://image.3001.net/images/20171125/15116102569194.png)

**结语**

在复杂的企业内网，定期使用IVRE对内网的端口扫描探测并保存结果，便于安全工作人员及时掌握内网的服务开放、是否存在弱口令等情况。

日常的渗透测试工作中，小规模的IP扫描，建议使用另一款工具——斯巴达（SPARTA，Kali自带，Github[链接](https://github.com/secforce/sparta)），输入IP，自动进行Nmap扫描，根据服务探测结果调用相关工具（如Hydra、Dibuster等）进行下一步的测试。

[](http://image.3001.net/images/20171125/15116102697136.png)

在安装过程中遇到：

Command: ["docker", "run", "--name", "ivreclient", "-d", "--link", "ivredb:ivredb", "-v", "/home/os/Downloads/ivre/ivre-share:/ivre-share", "-i", "-t", "ivre/client", {:notify=>[:stdout, :stderr]}]

Stderr: Unable to find image 'ivre/client:latest' locally

替换docker和linux的下载源，网络上的部分方法使用后尚未解决，截图是网上找的