Unicornscan

Unicornscan中文翻译过来大概意思是“独角兽”，很明显我联想到了360无线研究院的那群老表们。虽然我望之莫及。

好那么在来写Unicornscan，首先呢，Unicornscan是一款网络扫描工具，具有测量TCP/IP协议响应的高级接口的端口扫描器。

——*Unicornscan*

一，帮助文档

```
unicornscan (version 0.4.7)
使用方法: unicornscan [options
`b:B:cd:De:EFG:hHi:Ij:l:L:m:M:o:p:P:q:Qr:R:s:St:T:u:Uw:W:vVzZ:' ] X.X.X.X/YY:S-
E

-b, --broken-crc
*在[T]ransport层、[N]etwork层或两者上设置破碎的crc和[TN]

-B, --source-port
*设置源端口？或者扫描模块所期望的数字

-c, --proc-duplicates
处理重复回复

-d, --delay-type
*设置延迟类型(数值，有效选项为` 1:tsc 2:gtod 3:sleep ')

-D, --no-defpayload
没有默认有效负载，只探测已知协议

-e, --enable-module
*启用作为参数列出的模块(当前的输出和报告)

-E, --proc-errors
处理“非打开”响应的过程错误（icmp错误、tcp RST…）

-F, --try-frags
试试frags

-G, --payload-group
用于tcp/udp类型有效负载选择的有效负载组*有效负载组（数字）（默认全部）
```

```
-h, --help
帮助

-H, --do-dns
dns是否在报告阶段解析主机名

-i, --interface
接口*接口名称，如eth0或fxp1，通常不需要

-I, --immediate
立即模式，找到东西就显示出来

-j, --ignore-seq
ignore seq*ignore'A'll', R'eset用于tcp头验证的序列号

-l, --logfile
日志文件*写入此文件而不是我的终端

-L, --packet-timeout
数据包超时*等待数据包返回（默认为7秒）

-m, --mode
模式*扫描模式，默认为tcp（syn）扫描，U表示tcp的udp T，sf表示tcp连接扫描，对于-mT，您还可以
在类似于-mTsFpU的-mTsFpU后面指定tcp标志。

它将使用（NO syn | FIN | NO Push | URG）发送tcp syn数据包

-M, --module-dir
module dir*目录模块位于（默认为/usr/lib/unicornscan/modules）

-o, --format
format*回复显示内容的格式，格式说明见手册页

-p, --ports
端口要扫描的全局端口（如果未在目标选项中指定）

-P, --pcap-filter
*接收器的额外pcap过滤器字符串

-q, --covertness
隐蔽性*隐蔽性值从0到255

-Q, --quiet
不要用输出来筛选，它会去别的地方(数据库说...)

-r, --pps
pps*数据包/秒（总计，而不是每台主机，当您走得越高，它就越不准确）

-R, --repeats
重复*重复数据包扫描N次

-s, --source-addr
source addr*随机数据包“r”的源地址

-S, --no-shuffle
不随机播放不随机播放端口

-t, --ip-ttl
ip ttl*按62或6-16或r64-128设置发送数据包的ttl

-T, --ip-tos
```

```
ip tos*设置发送数据包的tos

-u, --debug
调试*调试掩码

-U, --no-openclosed
没有打开关闭不要说打开或关闭

-w, --safefile
safefile*写入接收包的pcap文件

-W, --fingerprint
fingerprint*OS fingerprint 0=cisco（def）1=openbsd 2=WindowsXP 3=p0fsendsyn
4=FreeBSD 5=nmap linux 7:strangetcp

-v, --verbose
-详细详细（每次更详细，所以-vvvv真的很详细）

-V, --version
版本显示版本

-z, --sniff
嗅嗅相似

-Z, --drone-str
drone str*drone字符串

*:带有` * '的选项需要一个参数

所有1的地址范围都像1.2.3.4/8一样是cidr。?。?。?

如果省略cidr掩码，则暗示/32

端口范围类似于1-4096，53只扫描一个端口，a代表所有65k，p代表1-1024

示例:unicorn scan-I et h1-Ir 160-E 192 . 168 . 1 . 0/24:1-4000网关:a
```

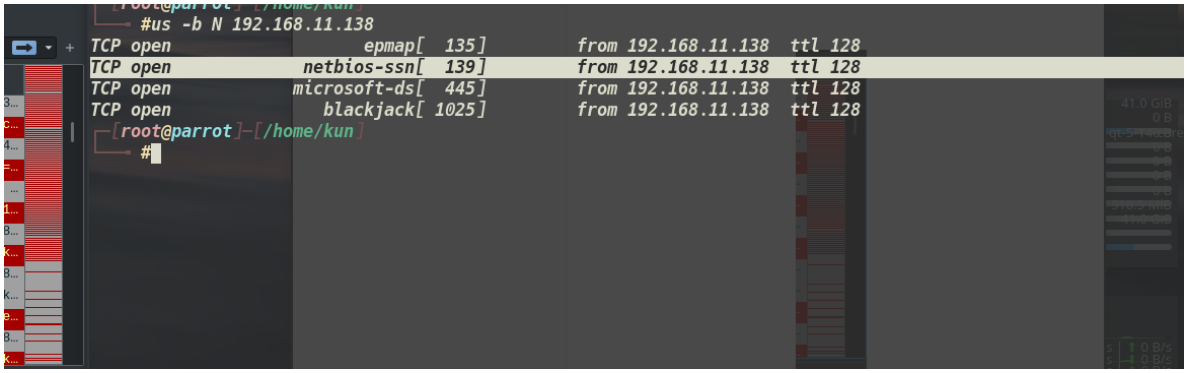
```
usage: unicornscan [options
`b:B:cd:De:EFG:hHi:Ij:l:L:m:M:o:p:P:q:Qr:R:s:St:T:u:Uw:W:vVzZ:' ] X.X.X.X/YY:S-
E
        -b, --broken-crc     *set broken crc sums on [T]ransport layer,
[N]etwork layer, or both[TN]
        -B, --source-port    *set source port? or whatever the scan module
expects as a number
        -c, --proc-duplicates process duplicate replies
        -d, --delay-type     *set delay type (numeric value, valid options are
`1:tsc 2:gtod 3:sleep')
        -D, --no-defpayload   no default Payload, only probe known protocols
        -e, --enable-module  *enable modules listed as arguments (output and
report currently)
        -E, --proc-errors     for processing `non-open' responses (icmp errors,
tcp rsts...)
        -F, --try-frags
        -G, --payload-group     *payload group (numeric) for tcp/udp type
payload selection (default all)
        -h, --help            help
```

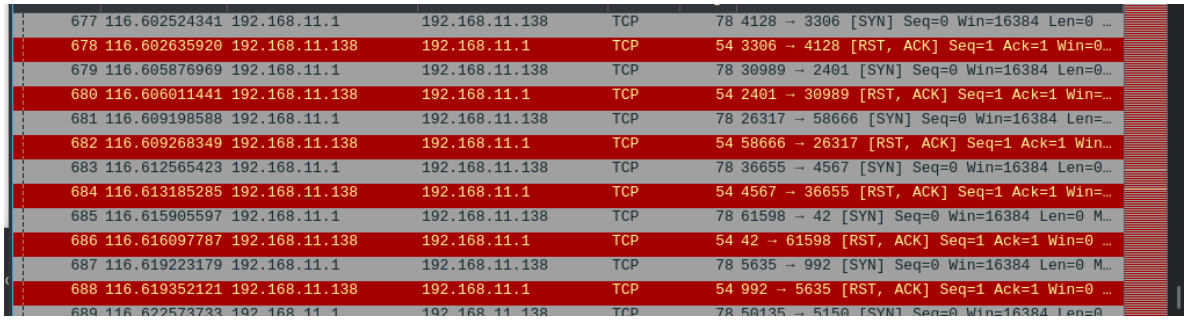
```
        -H, --do-dns          resolve hostnames during the reporting phase
        -i, --interface      *interface name, like eth0 or fxp1, not normally
required
        -I, --immediate       immediate mode, display things as we find them
        -j, --ignore-seq     *ignore `A'll, 'R'eset sequence numbers for tcp
header validation
        -l, --logfile        *write to this file not my terminal
        -L, --packet-timeout *wait this long for packets to come back (default
7 secs)
        -m, --mode           *scan mode, tcp (syn) scan is default, U for udp T
for tcp `sf' for tcp connect scan and A for arp
                               for -mT you can also specify tcp flags following
the T like -mTsFpU for example
                               that would send tcp syn packets with (NO
Syn|FIN|NO Push|URG)
        -M, --module-dir     *directory modules are found at (defaults to
/usr/lib/unicornscan/modules)
        -o, --format         *format of what to display for replies, see man
page for format specification
        -p, --ports           global ports to scan, if not specified in target
options
        -P, --pcap-filter    *extra pcap filter string for reciever
        -q, --covertness     *covertness value from 0 to 255
        -Q, --quiet           dont use output to screen, its going somewhere
else (a database say...)
        -r, --pps            *packets per second (total, not per host, and as
you go higher it gets less accurate)
        -R, --repeats        *repeat packet scan N times
        -s, --source-addr    *source address for packets `r' for random
        -S, --no-shuffle      do not shuffle ports
        -t, --ip-ttl         *set TTL on sent packets as in 62 or 6-16 or r64-
128
        -T, --ip-tos         *set TOS on sent packets
        -u, --debug             *debug mask
        -U, --no-openclosed      dont say open or closed
        -w, --safefile       *write pcap file of recieved packets
        -W, --fingerprint    *OS fingerprint 0=cisco(def) 1=openbsd 2=WindowsXP
3=p0fsendsyn 4=FreeBSD 5=nmap
                              6=linux 7:strangetcp
        -v, --verbose         verbose (each time more verbose so -vvvvv is
really verbose)
        -V, --version         display version
        -z, --sniff           sniff alike
        -Z, --drone-str      *drone String
*:      options with `*' require an argument following them

  address ranges are cidr like 1.2.3.4/8 for all of 1.?.?.?
  if you omit the cidr mask then /32 is implied
  port ranges are like 1-4096 with 53 only scanning one port, a for all 65k and
p for 1-1024
example: unicornscan -i eth1 -Ir 160 -E 192.168.1.0/24:1-4000 gateway:a
```

二，命令实例

> 循环沉余码校验（CRC，Cyclic Redundancy Check）
>
> 是一个检查通信线路中传输错误的方法。发送设备为数据块添加一个16或32位的多项式，他就是传输中附加在块中的循环沉余码校验。

> 接收端为数据进行对比，如果如果相符则接收，如果不相符则发送端重新发送数据块。





us -b N 192.168.11.138

在网络层进行检测，当数据包无问题后输出结果。

us -b T 192.168.11.138

在传输层进行检测，当数据包没有问题后输出结果

us -b NT 192.168.11.138

使用传输层与网络曾进行检测，当数据包没有问题后输出结果

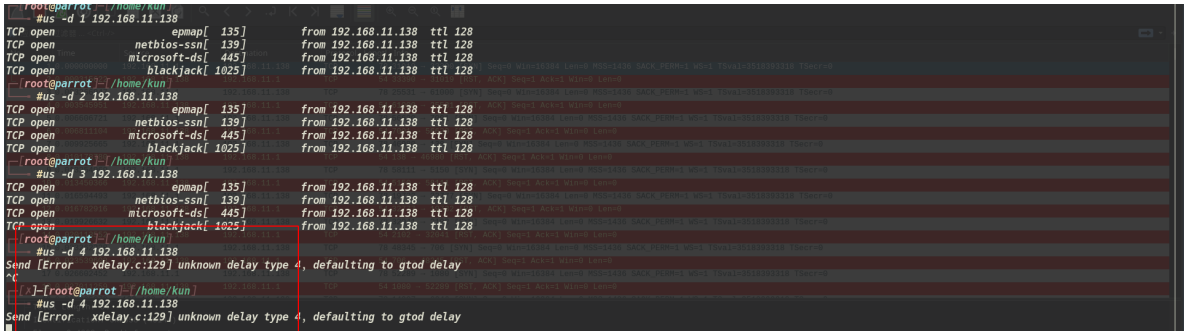
> 你说这样做的目的是什么，那我告诉你使用CRC的话，可以确保目标百分百，接收到数据包

us -B 1314 192.168.11.138

设置源端口为1314

us -c 192.168.11.138

处理所有重复的回复数据。

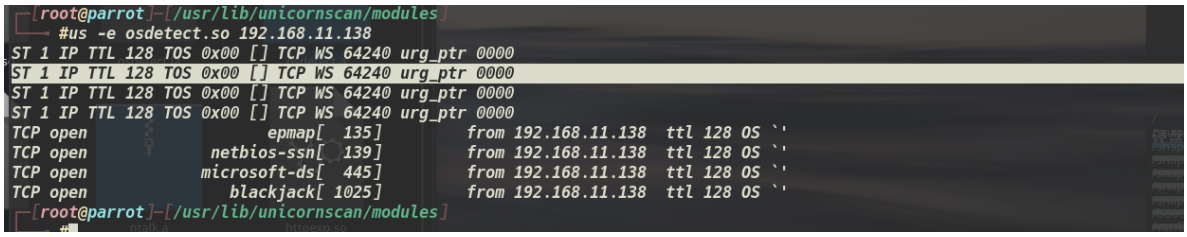


us -d 1 192.168.11.138

设置延迟为（默认提供三个参数，可自行进行调整 1 ~ 3 ）1,对数据包进行发送

us -D 192.168.11.138

探测已知协议，如TCP/IP（假如Uniconscan没有ICMP协议，则不使用icmp协议进行发送数据包）



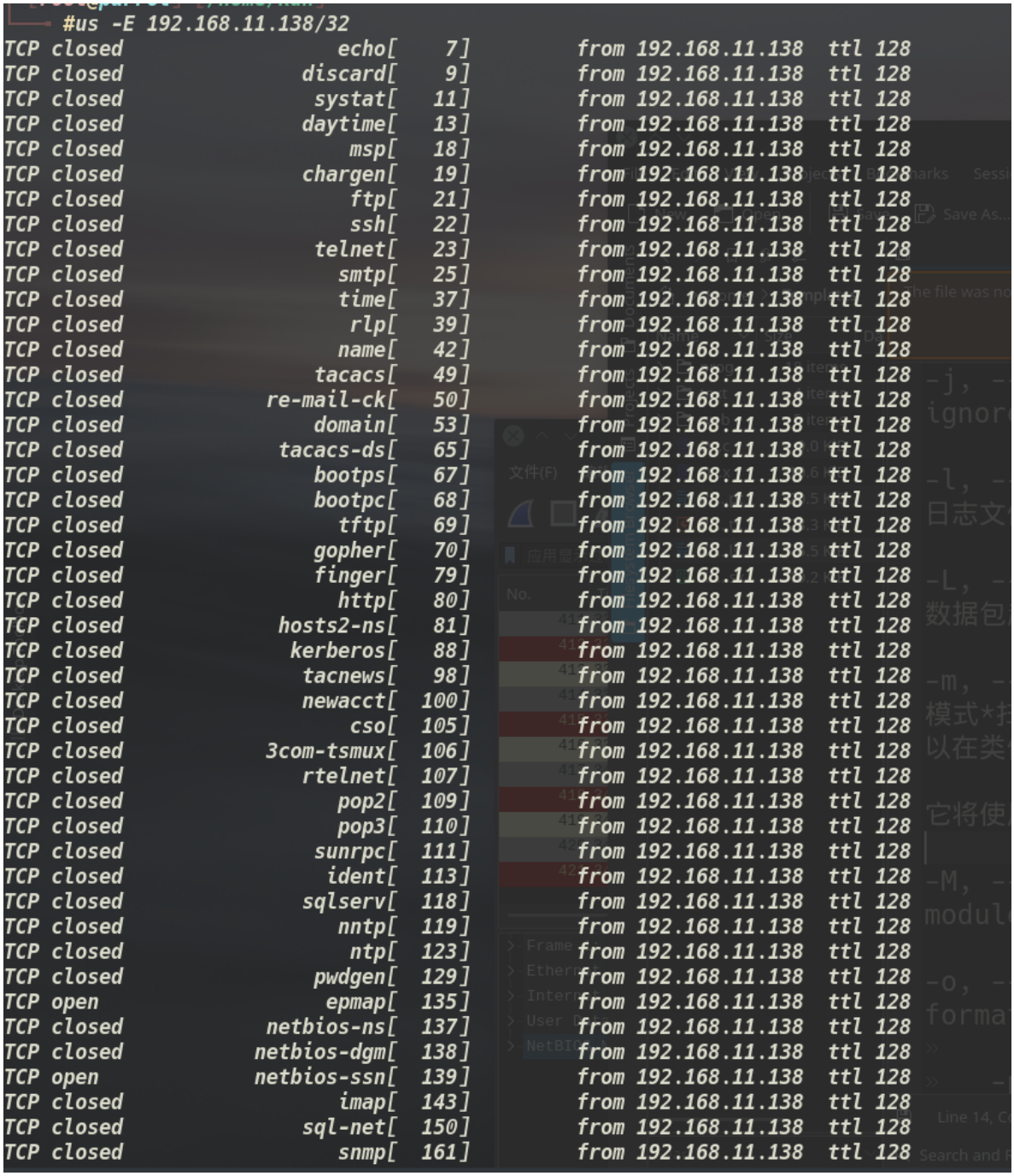
us -e odetect.so 192.168.11.138

使用odetect.so模块对目标进行检索

> 模块处于位置为：/usr/lib/unicornscan/modules/

模块名	模块作用
odetect.so	单独显示ttl值和TCP服务
sip.so	以一秒一个数据包的速度发送数据包

*部分模块介绍



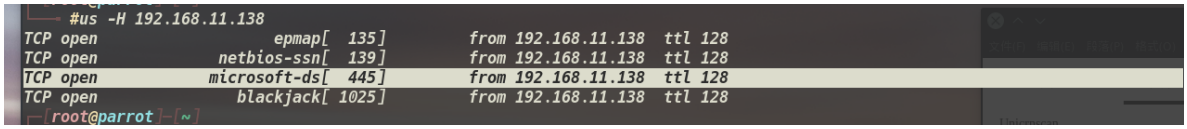
us -E 192.168.11.138/32

处理非打开响应过程错误（如ICMP错误等）

> 有效负载（FibreChannel）
>
> 通信帧或保温中数字字段的内容，在FibreChannel中，如果这些数据帧或者报文中包含帧头以及填充数字等可选信息，那么这些信息将不包括在有效负载之内。

us -G 负载值 192.168.11.138

用于TCP/UDP类型的有效负载组。



us -H 192.168.11.138

在报告阶段中解析主机名

us -i vmnet8 192.168.11.138

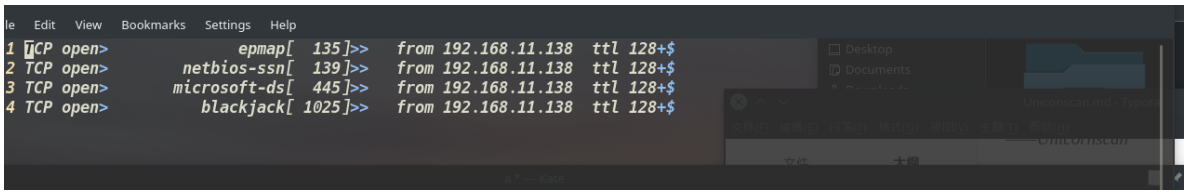
设置接口名称

us -I 192.168.11.138

即时模式，显示Unicornscan找到的东西

us -j all 192.168.11.138

主要用于TCP头验证序列号[^你可以选择任何一个TCP头验证序列，all是所有，当然你也可以使用ACK]



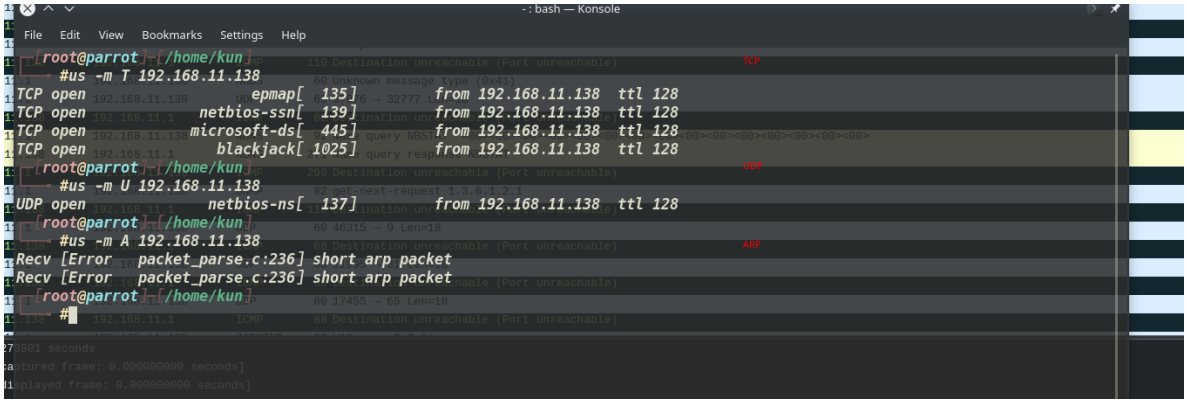
us -l 220 -b N 192.168.11.138

以网络层传输数据包，并使用CRC进行模式，如果双方判断数据包无误之后继续传输数据包。然后将扫描结果导出至220文件中。[^如果不使用-b参数，Unicornscan将会发送的是ARP包。]

us -L 1 192.168.11.138

设置数据包最大的返回值，如果超过指定时间则放弃该数据包[^加大扫描速度，默认为7]

> 扫描模式，默认为TCP扫描，U表示UDP扫描，T表示TCP扫描，A表示ARP扫描



us -m T 192.168.11.138

使用TCP进行扫描

us -m U 192.168.11.138

使用UDP进行扫描

us -m A 192.168.11.138

使用ARP进行扫描

us -p 445 192.168.11.138

只扫描445端口

> 过程特征分析软件包（PCAP）
>
> 这个抓包库给抓包系统提供了一个高层次的接口，所有网络上的数据包，甚至是那些发送给其他主机的，通过这种机制都可以进行捕获

us -q 255 192.168.11.138

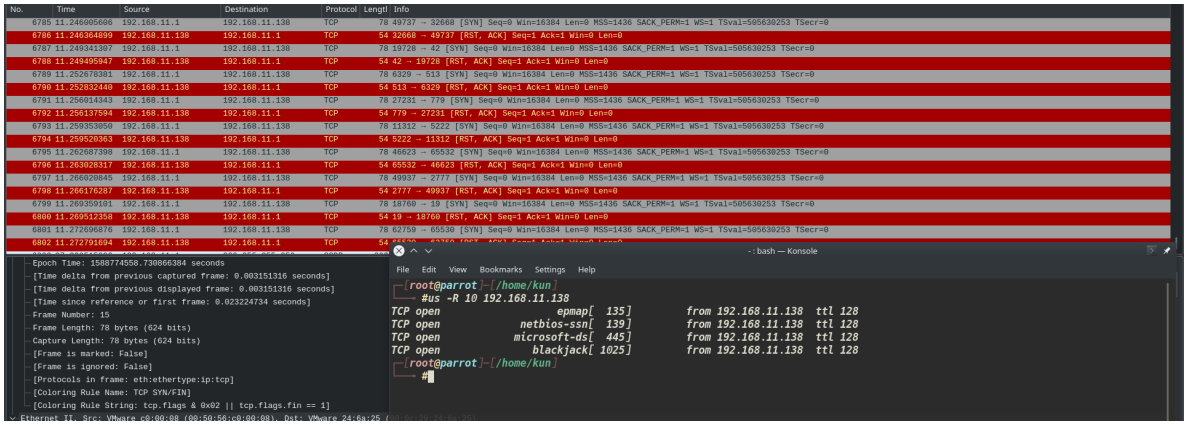
设置隐蔽性 1 ~ 255

us -Q 192.168.11.138

不输出任何信息。

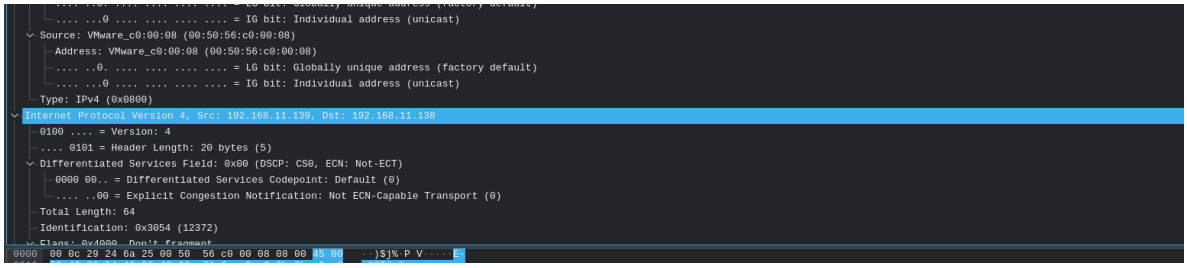
us -r 100 192.168.11.138

设置数据包发送速度，单位为s/秒



us -R 10 192.168.11.138

设置数据包发送重复次数为10次。



us -s 192.168.11.139 192.168.11.138

随机发送数据包，随机源地址为192.168.11.139

us -S 192.168.11.138

不随机发送数据包，不随机播放端口

us -t 134 192.168.11.138

设置生存时间（TTL）为134

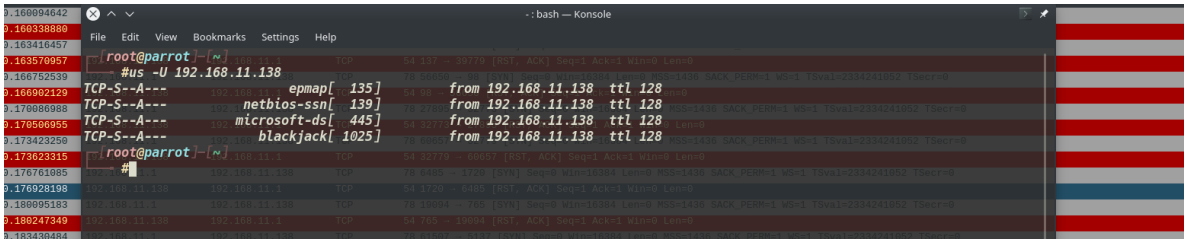
> 服务类型（TOS）
>
> TOS经常用于做（服务质量Service Quality）用于在传输过程中保证服务质量。
>
> 你可以理解为你去泡澡，提供有两个套餐，一个是10块钱，另一个15块钱，一个提供毛巾一个提供纸巾。

> 当你洗完之后你会发现都是一个洗。

us -T 10 192.168.11.138

设置服务类型为10.

us -u 192.168.11.138



us -U 192.168.11.138

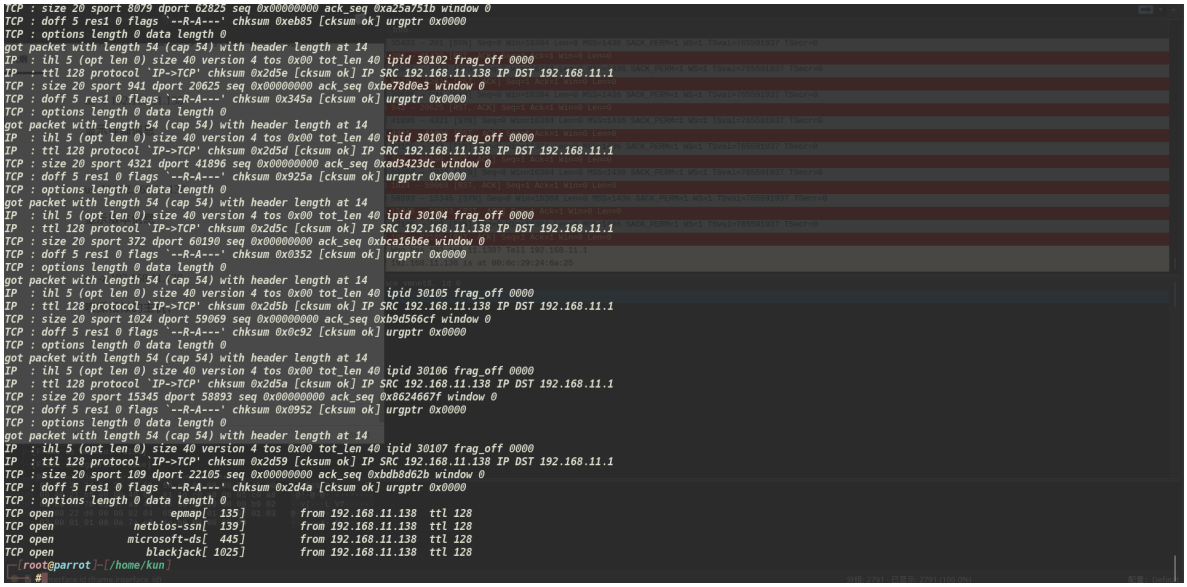
没有打开，不要说打开，打开了，不要说打开

us -v 192.168.11.138

显示详细信息

us -V 192.168.11.138

显示版本信息



us -z 192.168.11.138

嗅探相似的主机，