Zmap 使用文档.....3
一:Zmap 入门.....3
二:最佳扫描指南.....5
三:命令行参数.....5
四:附加信息:.....7
1.TCP SYN Scans.....7
2.ICMP 扫描:.....7
3.UDP 扫描.....7
4.配置文件.....8
5.显示结果.....8
6.结果输出:.....10
7.黑名单和白名单:.....11
8.速率限制和随机采样:.....12
9.发送多个探测包:.....13
五:zmap 扩展部分.....13
1.实例应用.....13
2.编写探针和输出模块.....15

译者水平有限，时间仓促,难免瑕疵，如有错误，敬请指正

翻译：newbie0086#foxmail.com

Zmap 是一个开源网络扫描器，仅需一台有良好的网络上传速度的机器就能让研究人员对互联网进行广泛的研究，在不到 45 分钟内，就能实现对 IPV4 所有 IP 的扫描
ZMap 随着时间的推移，可以用来研究协议，监控服务，和帮我们更好的了解分布在互联网大型系统
Zmap 开发由密歇根大学的计算机科学家团队：

- Zakir Durumeric, 博士 候选人，密歇根大学
- Eric Wustrow,博士 候选人，密歇根大学
- J. Alex Halderman, 助理教授，美国密歇根大学

研究小组联系方式: zmap-team@umich.edu.

Zmap 对研究者是一个强大的工具，在运行 Zmap 的时候，你可能会扫描到某些人不想扫描的IPv4地址，我们鼓励 Zmap 使用者有责任心的扫描，停止和排除对一些网络地址的扫描

我们提倡使用者是有授权的扫描,开发出一套研究者需要的最佳扫描指南，我们不建议研究者对存在漏洞和受保护的资源进行扫描，并遵守相关法律法规

Zmap 使用文档

一:Zmap 入门

Zmap 设计用来扫描 IPV4地址和大部分地址进行全面扫描，Zmap 对研究者是一个强大的扫描工具，所以，在运行 Zmap 的时候，请记住，你扫描的整个 IPV4地址每秒发包可能超过140w个，在执行任何扫描前，我们建议与该管理员联系，并查阅我们列出的最佳扫描指南

默认情况下，Zmap 将会执行在特定端口执行最高效率的 TCP SYN 扫描，在10M 网速下，执行1W 个随机地址80端口的扫描如下：

```
$zmap --bandwidth=10M --target-port=80 --max-target=10000 --output-file=results.txt
```

或更简洁的命令:

```
$zmap -B 10M -p 80 -n 10000 -o results.txt
```

如果扫描成功启动，将会输出每秒的状态如下

```
0% (1h51m left); send: 28777 562 Kp/s (560 Kp/s avg); recv: 1192 248 p/s (231 p/s avg); hits: 0.04%
```

```
0% (1h51m left); send: 34320 554 Kp/s (559 Kp/s avg); recv: 1442 249 p/s (234 p/s avg); hits: 0.04%

0% (1h50m left); send: 39676 535 Kp/s (555 Kp/s avg); recv: 1663 220 p/s (232 p/s avg); hits: 0.04%

0% (1h50m left); send: 45372 570 Kp/s (557 Kp/s avg); recv: 1890 226 p/s (232 p/s avg); hits: 0.04%
```

扫描状态:%-complete(美国东部时间);发送的包和当前发送比例(avg-发送比率);接收到的包和接收比率(avg-接收比率);hits：成功率

如果你不知道你网络能支持的扫描速度，你可以尝试安不同的扫描速度或者调节 bandwidth 限制，比如4M

默认的，Zamp 根据 SYN ACK 的返回打印出 IP 列表，还有一些附加的输出格式，提供不同的输出结果

```
115.237.116.119

23.9.117.80

207.118.204.141

217.120.143.111

50.195.22.82
```

我们强烈建议你使用黑名单文件(主要是些私有地址)排除一些不需要扫描的地址,默认情况下,zamp 将会加载一个简单的黑名单文件，其中包涵保留地址和未分配的地址，位置在/etc/zamp/blacklist.conf 如果你发现每次启动 zmap 时,你自己的需要一些特定的扫描规则，比如你的宽带速度限制或黑名单，可以修改/etc/zmap/zmap.conf 或者使用自定义的配置文件

如果你正尝试解决扫描故障，有几个选项可以帮助调试，你可以执行 dry run scan(预扫描模式)使用--dryrun 选项来看查看网络发送的数据包，同时通过--verbosity=n 改变日志记录

二:最佳扫描指南

我们给研究者提供了一些扫描建议来指导研究者更好的扫描

1.密切与网络管理员联系，减少风险和查水表

2.扫描时不对网络资源占尽或者对影响上游供应商

3.网络信号优良并配置好 DNS

4.明确指定扫描的范围

5.提供一个简单的退出方法和即时兑现请求

6.除了目标研究范围内，不要进行更多的频繁的，大量的请求

7.随着时间的推移进行扫描，或者源地址有流量传播时进行扫描

研究人员应该禁止利用漏洞或者对受保护的文件进行扫描，并需遵守相关的法律法规

三:命令行参数

常见的选项

这些选项是执行常规简单扫描时，最常用的，我们注意到一些选项是基于 probe module 和 ooutput module 的(比如目标端执行 icmp 扫描时不可用)

-p,--target-port=port TCP 端口号(比如443)

-o,--output-file=name 我们要保存的扫描结果，用- 表示屏幕输出

-b,--blacklist-file=path 子网排除使用 CIDR 表示方法，参考 conf/blacklist.example

扫描选项:

-n,--max-target=n 最大目标数量，可以是一个数字比如-n 1000或者-n 0.1%（可扫描的地址空间），不包含黑名单中的地址

-N,--max-results=n 接受到多少结果后，推退出

-t,--max-runtime=secs 发送数据包最长时间

-B，--bandwidth=bps 设置发送速率 数据包/秒

-c,--cooldown-time=secs 在发送数据包后多少时间收返回数据(默认8s)

-e,--seed=n 用于选择地址排序，如果你想运行多个 zmap 对地址进行扫描时使用

-T,--sender-threads=n 发包线程（默认1）

-P，--probes=n 对每个 IP 进行探测的次数（默认每个 IP 一次探测）

-d,--dryrun 调试时使用，在屏幕显示每个数据包，但不发送

网络选项:

-s,--source-port=port|range 发送数据包的源端口

-S,--source-ip=ip|range 发送数据包的 IP 地址，可以是单个 ip 也可以是返回(e.g 10.0.1-10.0.0.9)

-G --gateway-mac-addr 发送数据包的网管 MAC 地址(不起作用下会自动检测)

模块选项:

Zmap 允许用户使用指定的或者书写自己的模块，zmap 支持两个类型的模块，探头模块和输出模块，探头模块负责生成探测包和分类处理 hosts 的返回，输出模块负责显示或者存储扫描中的结果

-M,probe-module=name 选择探头模块(默认=tcp_synscan)

-O,--output-module=name 选择输出模块(默认=simple_file)

--probe-args=args 传递给探头模块的参数

--output-args=args 传递给输出模块的参数

--list-output-modules 列出所有可用的输出模块

--list-probe-modules 列出所有可用的探头模块

附加选项:

-C,--config=filename 加载配置文件

-q,--quit 不再打印每秒更新的状态

-g,--summary 在扫描完成后打印出结构和总结结果

-v,--verbosity=n 详细级别(0-5，默认3)

-h,--help 帮助信息

-V，--version 打印出版本信息

四:附加信息:

1.TCP SYN Scans

当执行 TCP SYN scan 时，Zmap 需要指定目标的单个端口和源端口范围

-p,target-port=port 要扫描的 TCP 端口

-s,--source-port=port|range 用来扫描的源端口(e.g.40000-50000)

警告！Zmap 依赖 Linux 内核来处理 SYN/ACK 包(关闭扫描器已建立的连接)

Zmap 在以太网层发送数据包(只做交换，不做路由)，以减少开销，否则会在内核中跟踪打开 Tcp 连接，执行路由查找，如果你开启了防火墙规则比如-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT,会导致 SYN/ACK 无法到达内核，虽然不会影响 ZMAP 记录，但是会阻止发送 RST 包(断开连接)，最终导致超时,我们强烈推荐你在扫描主机时，使用未使用的端口,并且是你的防火墙不过滤的端口，使用-s 标记(-s ‘50000-60000’)

2.ICMP 扫描:

Zmap 默认是用 tcp syn 扫描，它也支持 ICMP 扫描(ping 扫描就是 ICMP 类型的)，可使用下面的方法执行 ICMP 扫描

$zamp --probe--module=icmp_echoscan

3.UDP 扫描

zmap 还支持 UDP 扫描,你可以发送任意 UDP 数据包到每个主机，并接收 UDP 或者 ICMP 不可到达的回应，在 probe_modules/module_udp.c 设置数据包载荷，必须以空字符-结尾,将来我们会在命令行提供 UDP 的数据包载荷

```
$ zmap --probe-module=udp -p 53 -N 100 -o -
```

4.配置文件

Zmap 支持配置文件，从而不用在命令行进行繁琐设置，配置中可以使用长名选项，每行的值像下面这样

```
interface "eth1"

source-ip 1.1.1.4-1.1.1.8

gateway-mac b4:23:f9:28:fa:2d # upstream gateway

cooldown-time 300 # seconds

blacklist-file /etc/zmap/blacklist.conf

output-file ~/zmap-output

quiet

summary
```

由此，zmap 可以使用执行配置文件和指定任何额外所需参数来执行扫描

```
$ zmap --config=~/.zmap.conf --target-port=443
```

5.显示结果

Zmap 有几种类型的非屏幕显示结果，默认会显示每秒更新状态，这个可以用`--quiet` flag 取消

```
0:01 12%; send: 10000 done (15.1 Kp/s avg); recv: 144 143 p/s (141 p/s avg); hits: 1.44
%
```

Zmap 还提供了如下的扫描信息，可以通过`--verbosity` 调节

```
Aug 11 16:16:12.813 [INFO] zmap: started

Aug 11 16:16:12.817 [DEBUG] zmap: no interface provided. will use eth0

Aug 11 16:17:03.971 [DEBUG] cyclic: primitive root: 3489180582

Aug 11 16:17:03.971 [DEBUG] cyclic: starting point: 46588

Aug 11 16:17:03.975 [DEBUG] blacklist: 3717595507 addresses allowed to be scanned

Aug 11 16:17:03.975 [DEBUG] send: will send from 1 address on 28233 source ports

Aug 11 16:17:03.975 [DEBUG] send: using bandwidth 10000000 bits/s, rate set to 14880 pk
t/s

Aug 11 16:17:03.985 [DEBUG] recv: thread started
```

Zmap 还支持类似于如下的扫描结果总结，你可以调用`--summary`来显示

```
cnf   target-port             443
cnf source-port-range-begin         32768
cnf source-port-range-end           61000
cnf source-addr-range-begin         1.1.1.4
cnf source-addr-range-end           1.1.1.8
cnf maximum-packets             4294967295
cnf maximum-runtime             0
cnf permutation-seed            0
cnf cooldown-period             300
cnf send-interface              eth1
cnf rate                    45000
env nprocessors             16
exc send-start-time             Fri Jan 18 01:47:35 2013
exc send-end-time               Sat Jan 19 00:47:07 2013
exc recv-start-time             Fri Jan 18 01:47:35 2013
exc recv-end-time               Sat Jan 19 00:52:07 2013
exc sent                    3722335150
exc blacklisted             572632145
exc first-scanned               1318129262
exc hit-rate                0.874102
exc synack-received-unique          32537000
exc synack-received-total           36689941
exc synack-cooldown-received-unique     193
exc synack-cooldown-received-total      1543
exc rst-received-unique         141901021
exc rst-received-total          166779002
adv source-port-secret          37952
adv permutation-gen             4215763218
```

6.结果输出:

Zmap 可以通过使用输出模块提供不同显示结果，zmap 目前支持三种格式:

Simple-file extended-file 和 redis,.默认情况下 zmap 返回结果是简单文件格式，如果没有指定输出文件,zmap 不会产生结果，你可以编写自己的输出模块，参考:here

-o,--output-file=p 将扫描到的 Ip 写到此文件

-O，--output-module=p 调用自定义输出模块

--list-output-modules 列出可以用的输出模块

Simple File 简单文件格式

简单文件格式模式只提供成功扫描的 ip（(比如 syn 扫描结果)

```
115.237.116.119

23.9.117.80

207.118.204.141

217.120.143.111

50.195.22.82
```

Extend File 扩展文件

扩展文件将会提供每条记录的详细信息

```
response, saddr, daddr, sport, dport, seq, ack, in_cooldown, is_repeat, timestamp

synack, 159.174.153.144, 10.0.0.9, 80, 40555, 3050964427, 3515084203, 0, 0,2013-08-15 18:55:47.681

rst, 141.209.175.1, 10.0.0.9, 80, 40136, 0, 3272553764, 0, 0,2013-08-15 18:55:47.683

rst, 72.36.213.231, 10.0.0.9, 80, 56642, 0, 2037447916, 0, 0,2013-08-15 18:55:47.691

rst, 148.8.49.150, 10.0.0.9, 80, 41672, 0, 1135824975, 0, 0,2013-08-15 18:55:47.692

rst, 50.165.166.206, 10.0.0.9, 80, 38858, 0, 535206863, 0, 0,2013-08-15 18:55:47.694

rst, 65.55.203.135, 10.0.0.9, 80, 50008, 0, 4071709905, 0, 0,2013-08-15 18:55:47.700

synack, 50.57.166.186, 10.0.0.9, 80, 60650, 2813653162, 993314545, 0, 0,2013-08-15 18:55:47.704

synack, 152.75.208.114, 10.0.0.9, 80, 52498, 460383682, 4040786862, 0, 0,2013-08-15 18:55:47.707
```

```
synack, 23.72.138.74, 10.0.0.9, 80, 33480, 810393698, 486476355, 0, 0,2013-08-15 18:55:47.710
```

Redis 再分配格式

Redi 再分配格式，不将记录文件中，而是放在一个队列中，用于 zmap 后期的处理

注意:这个 zmap 默认不支持，你可以在编译 zmap 源码时使用该命令来支持 `make REDIS=true`.

7.黑名单和白名单:

Zmap 支持黑名单和白名单，如果你没有给 zmap 指定黑名单和白名单，那么 zmap 将会扫描所有的 Ipv4地址!(包括本地私有地址，保留地址，和互联网组播地址)，提供了黑名单则不会扫描黑名单地址，如果提供了白名单则只会扫描白名单地址，两着可以一起提供，如果黑名单与白名单有交集的地址则不会被扫描，黑白名单的地文件可以命令指定:

-b,--blacklist-file=path 可用 CIDR 格式表示 e.g.192.168.0.0/16

-w,--whitelist-file=path 可用 CIDR 格式表示 e.g.192.168.0.0/16

黑名单文件格式应在 CIDR 表示法，每行一个单一的网络前缀。允许注释使用`#`字符注销。例如:

```
# From IANA IPv4 Special-Purpose Address Registry

# http://www.iana.org/assignments/iana-ipv4-special-registry/iana-ipv4-special-registry.xhtml

# Updated 2013-05-22


0.0.0.0/8           # RFC1122: "This host on this network"

10.0.0.0/8          # RFC1918: Private-Use

100.64.0.0/10       # RFC6598: Shared Address Space

127.0.0.0/8         # RFC1122: Loopback

169.254.0.0/16      # RFC3927: Link Local

172.16.0.0/12       # RFC1918: Private-Use

192.0.0.0/24        # RFC6890: IETF Protocol Assignments

192.0.2.0/24        # RFC5737: Documentation (TEST-NET-1)

192.88.99.0/24      # RFC3068: 6to4 Relay Anycast
```

```
192.168.0.0/16      # RFC1918: Private-Use
192.18.0.0/15       # RFC2544: Benchmarking
198.51.100.0/24     # RFC5737: Documentation (TEST-NET-2)
203.0.113.0/24      # RFC5737: Documentation (TEST-NET-3)
240.0.0.0/4         # RFC1112: Reserved
255.255.255.255/32  # RFC0919: Limited Broadcast

# From IANA Multicast Address Space Registry
# http://www.iana.org/assignments/multicast-addresses/multicast-addresses.xhtml
# Updated 2013-06-25

224.0.0.0/4         # RFC5771: Multicast/Reserved
```

如果你想要进行随机的扫描，请查看 sampling 随机采样，不再使用黑白名单

注意: zmap 默认的黑名单文件地址`/etc/zmap/blacklist.conf`,默认配置文件在

`/etc/zmap/zmap.conf`

8.速率限制和随机采样:

默认的，在你网络支持的范围内进行最快速的扫描，商用硬件经验告诉我们，千兆以太网能理论上能处理95-98%的扫描任务，zmap 不会自动处理发包速度，你可以手动调节发包速度，以减少包的丢失和错误结果

-r,--ate=ps 设置最大的发包速度 数据包/每秒

-B，--bandwidth=bps 设置发包的字节(支持 G，M,K 等单位) 字节/每秒,通过`--rate`来调节

Zmap 运行对 IPV4随机采样扫描，可通过指定 max-targets 或 max-runtime 来设置，因为扫描对象是随机生成的，所以可以在命令行指定随机扫描的主机数量

-n,--max-targets=n 探测数量上限

-N，--max-results=n 最大结果数(有效的结果数量)

-t,--max-runtimes=s 一段时间内发送数据包时间值

-s,--seed=n 多个 zmap 运行时，按照地址排序进行扫描

比如,如果你想对互联网上1w个主机再次扫描,你可以给不同的zmap扫描器设置相同的seed

```
zmap -p 443 -s 3 -n 1000000 -o results
```

为了确定要扫描的1W台主机被正确执行，你可以使用调试模式 dry-run，打印出将要发送出去的包(不是真的发送)

```
zmap -p 443 -s 3 -n 1000000 --dryrun | grep daddr
```

```
    | awk -F'daddr: ' '{print $2}' | sed 's/ |.*//;'
```

9.发送多个探测包:

Zmap 支持对同一主机发送多次探测，默认是1此，增加这个次数当然也会影响扫描时间和主机到达时间，然而我们发现相比增加主机到达次数而言，增加扫描时间100%增加探测次数，可以通过

-P，--probes=n 设置每个 ip 要探测的次数（默认1）

五:zmap 扩展部分

1.实例应用

Zmap 设计用来接触大量主机，寻找活跃主机，然而，许多使用者想要执行后续的处理，比如某些使用者使用 SYN 扫描，在80端口上执行一次简单的 get 请求，但是又想扫描443端口

Banner 抓取

我们提供了 banner-grab 的典型应用， zmap 允许使用者接收 TCP 服务器返回的 message,Banner-grab 连接到服务器，可随意发送 message，并打印出接收到的服务器第一条消息，Banner-grab 工具可以用来抓取 HTTP SERVER 特定响应，telnet 登录提示，和 SSH server 字符串

```
$ zmap -p 80 -N 1000 -B 10M -o - | ./banner-grab-tcp -p 80 -c 500 -d ./http-req > out
```

更多关于 banner-grab,请查看 `examples/banner-grab` README 文件

注意:zmap 和 banner-grab 同时执行将会影响扫描的效率和精确度，需确保不让 zmap 影响 banner-grab 当前的连接，否则 banner-grab 失效，并引起 Zmap 锁定屏幕标准输出，我们建议调低 Zmap 速度,并增 banner-grab 不超过3000 大于1000 需要你使用 `ulimit -SHn 100000` and `ulimit -HHn 100000`来增加文件最大处理进程数）的并发连接,这些参数将依赖你的服务器性能，我们建议先小样本实验，再大规模应用

Forge Socket(伪造套接字)

Forge Socket，可以重新使用服务器发送回来的 SYN-ACK 响应，zmap 发送 SYN 到每个服务器，并且监听服务器返回的 SYN+ACK，然后 Zmap 服务器接收到相应包，并发送 RST 断开请求，banner-grab 必须马上创建一个新的 TCP 连接相同服务器上，并抓取响应数据

forge-socket，我们利用内核模块相同的名称，运行我们创建任意 TCP 参数，这能让我们支持抑制内核的 RST 包，取而代之的是利用 SYN+ACK 创建一个套接字，通过套接字发送和接收数据

使用 fore-socket,你必须下载 forge-socket 内核模块 github.也可使用命令下载:

`git clone git@github.com:ewust/forge_socket.git`

在 zmap 根目录里使用 cd 命令进入 forget_socket，然后执行 make 编译，用 root 身份安装内核模块 `insmod forge_socket.ko`

你必须告诉内核不要发送 RST 包，禁止 RST 包最简单的方法就是使用 iptalbes

`iptables -A OUTPUT -p tcp -m tcp --tcp-flgas RST,RST RST,RST -j DROP`

另外你可能还会使用--dport X 来限制你正在扫描的端口，当扫描完成后，你可以使用

`iptables -D OUTPUT -p tcp -m tcp --tcp-flags RST,RST RST,RST -j DROP` 来删除防火墙规则

现在，你应该会编译 forge-socket for zmap,运行它你必须使用 extended_file 输出模式

```
$ zmap -p 80 -N 1000 -B 10M -O extended_file -o - | \
```

```
    ./forge-socket -c 500 -d ./http-req > ./http-banners.out
```

更多信息查询 `examples/forge-socket` README 文件

2.编写探针和输出模块

Zmap 通过 probe 模块支持多种类型扫描，并且附加多了多种 output modules 输出模块，可以在命令行界面注册 probe 和 output 模块

--list-probe-modules 列出安装的 probe 模块

--list-output-modules 列出暗转的 output 模块

输出模块:

Zmap 扫描器输出和后续处理可以在初始化完成，输出模块可回调处理任何返回的数据包，尽管默认提供的模块是 simple out 文件输出，这些模块也能捕获执行附加的后续处理(比如跟踪重复和数字输出，而不是 ip 地址)

输出模块的创建结构如下，参考 output_modules.c:

```
struct output_module {

    const char          *name;          // how is output module referenced in the CLI
    unsigned            update_interval; // how often is update called in seconds


    output_init_cb      init;           // 扫描器初始化
    output_update_cb    start;          // 初始化调用扫描器
    output_update_cb    update;         // 更此新周期秒
    output_update_cb    close;          // 在终止扫描仪调用
    output_packet_cb    success_ip;     // 成功时候调用
    output_packet_cb    other_ip;       // 接收到其他响应时调用


} output_module_t;
```

输出模块必须要给一个名称，以实现在命令行的引用，success_ip 通过当前的 probe 探测模块，在接收到数据并被归类为 sucess 时调用(e.g TCP syn/ack 或者 ICMP 扫描的返回信息)，当 ICMP 路由不可达或者 TCP RST 造成的非成功但是有效的响应，则会调用 other_ip

,这些回调必须定义功能相匹配的 `output_packet_cb` 定义：

```
int (*output_packet_cb) (

    ipaddr_n_t    saddr,          // 已扫描的 ip 地址
    ipaddr_n_t    daddr,          // 需要扫描的 ip 地址

    const char*   response_type, // 用于包的分类
    int           is_repeat,      // {0: 主机第一次响应, 1: 主机后来的响应}
    int           in_cooldown,    // {0: 非冷却状态, 1: 扫描器进入冷却状态}

    const u_char* packet,         // pointer to struct iphdr of IP packet
    size_t        packet_len      // 包的长度
);
```

输出模块也可以在扫描器初始化时注册回调函数来执行命令(比如打开一个 output 文件)，开始扫描时候打开黑名单文件，在扫描过程中更行进度，或关闭打开文件，这些回调提供了完整扫描配置和当前状态

```
int (*output_update_cb)(struct state_conf*, struct state_send*, struct state_recv*);
```

在定义输出模块的时候，可以参考 src/output_modules/module_extended_file.c.

探针模块：

Probe 模块让抽象的数据包创建和执行响应分类，zmap 有两种扫描模块，分别是 `tcp_synscan` and `icmp_echoscan.`，默认的 zmap 是使用 `tcp_synscan` 方式，用来发送 TCP SYN 数据包以及把每个主机的 ack 响应进行分类(接受到 ACK 或者 RST 掉的)，zmap 也允许开发者编写他们自己的探测模块，可以参考下面的 API

每种扫描类型的都是需要在 `send_module_t 结构中`通过开发和注册回调函数来实现

```
typedef struct probe_module {
    const char                  *name;              // 命令行调用时所需名称
    size_t                      packet_length;      // 探测包长度
```

```
    const char                  *pcap_filter;       // PCAP 对返回包进行过滤
    size_t                      pcap_snaplen;      // libpcap 可捕获的最大字节
    uint8_t                     port_args;         // 如果 zmap 需要一个--target-port,默认
                                                   //设置为 1
    probe_global_init_cb        global_initialize; // 扫描器初始化时调用
    probe_thread_init_cb        thread_initialize; // 每个数据包有 buffer 时调用
    probe_make_packet_cb        make_packet;       //  每次主机状态更新时调用
    probe_validate_packet_cb validate_packet;     // 每次接受到数据包时调用
                                                   // 如果数据包无效则返回 0
                                                   // 非 0 则否
    probe_print_packet_cb       print_packet;      // 在预执行模式中调用
    probe_classify_packet_cb classify_packet;     // 每次对返回数据包进行分类调用
    probe_close_cb              close;             // 结束调用
    response_type_t             *responses;        //可能的返回响应分类

} probe_module_t;
```

在扫描器初始化话，`global_initialize`被调用,用来指定全局配置,然而`global_initialize`并不具有线程访问能力，`thread_initialize`则是用来初始化线程并构造带有全局配置及目标值的探测包,这个回调函数应被用于构造:主机不可知的包结构，比如只有特定值的(e.g 目标主机和校验)需要更新的每个主机,例如以太网报头不会改变头(减去校验是通过网卡硬件计算)因此可以提前定义时间，减少扫描时间开销;

`make_packet`回调将会被每个被扫描的主机调用,这些主机是允许探测模块来更新主机特定的值(ip 地址的值)，不透明的验证字符串,t 探测模块负责收集大量的验证字符串，依照这种方式，当一个 server 返回一个有效的响应，探测模块将会验证是否是当前的，比如，一个 syn 扫描，tcp_syncan 探测模块将会使用 TCP 源地址和序列号来存储有效的字符串，返回的包(SYN+ACK)将会包含有效的，在目标端口和确认值

```
int make_packet(
    void        *packetbuf,  // packet buffer
    ipaddr_n_t  src_ip,      // 网络源 ip 地址
    ipaddr_n_t  dst_ip,      // 目标 IP 地址
    uint32_t    *validation, // 需放到探针里的验证字符串
    int         probe_num    // if sending multiple probes per host,
                             // this will be which probe number for this
                             // host we are currently sending
);
```

分类功能在将会接受数据包缓冲区并返回一个返回类型指针在接收到的每个包上,这个分类功能将会被使用作为决定是否输出