

## 常见攻击工具特征及检测分析





## 目录

1	隧道通信.....	4
1.1	DNS 隧道 .....	4
1.1.1	CobaltStrike DNS Beacon .....	4
1.2	ICMP 隧道 .....	5
1.2.1	pTunnel .....	5
1.2.2	icmpsh .....	6
2	代理通信.....	7
2.1	NPS .....	7
2.1.1	NPS 简介 .....	7
2.1.2	NPS 通信特征 .....	7
2.1.3	NPS 使用备忘 .....	9
2.2	FRP .....	12
2.2.1	FRP 简介 .....	12
2.2.2	FRP 通信特征 .....	12
2.2.3	FRP 使用备忘 .....	14
2.2.3.1	FRP 配置文件说明 .....	14
2.2.3.2	FRP 典型配置举例 .....	16
2.3	VENOM.....	19
2.3.1	venom 简介 .....	19
2.3.2	venom 通信特征 .....	19
2.3.3	venom 使用备忘 .....	21
2.4	FASTTUNNEL .....	22
2.4.1	FastTunnel 简介 .....	22
2.4.2	FastTunnel 通信特征 .....	22
2.5	TERMITE .....	23
2.5.1	Termite 简介 .....	23
2.5.2	Termite 通信特征 .....	23
2.5.3	Termite 使用备忘 .....	24
2.6	STOWAWAY .....	25
2.6.1	Stowaway 简介 .....	25
2.6.2	Stowaway 通信特征 .....	25
2.6.3	Stowaway 使用备忘 .....	26
2.7	LANPROXY .....	29
2.7.1	Lanproxy 简介 .....	29
2.7.2	Lanproxy 通信特征 .....	29
2.7.3	Lanproxy 使用备忘 .....	30
2.8	长亭百川主机管理助手 .....	30
2.8.1	简介 .....	30
2.8.2	通信和终端行为特征 .....	31



2.8.3	使用备忘 .....	31
2.9	QUASAR.....	33
2.9.1	Quasar 简介.....	33
2.9.2	Quasar 通信特征.....	33
2.9.3	Quasar 使用备忘.....	34
3	扫描渗透工具 .....	34
3.1	RSAS .....	34
3.2	GOBY .....	35
3.3	XRAY .....	35
3.4	EZ .....	35
3.5	FSCAN .....	35
3.6	NMAP .....	35
3.7	METASPLOIT .....	36
结语.....	错误!未定义书签。	

# 1 隧道通信

## 1.1 DNS 隧道

### 1.1.1 CobaltStrike DNS Beacon

默认配置下的 cs4.4，dns 通信时会向固定开头的域名发起 TXT 类型的域名解析请求，

详情		
入侵检测日志告警详情		
告警时间	会话ID	攻击IP
2022-03-24 15:49:12	623c22780f0794411d2f75d401	172.16.10.43
攻击端口	攻击阶段	受害IP
52674	持续控制	114.114.114.114
受害端口	研判结果	告警摘要
53	尝试攻击	DNS
告警详情		
<pre>0A?ix"cb\$""P0'N &amp;EP*"E~ *rrrrfA5Om" 4aaastage5843103ns microsoftcom</pre>		
单包下载		
规则ID	威胁等级	网卡信息
41763	高危	G1/1
响应码	威胁方向	应用描述
0	内->外	
策略动作字段		

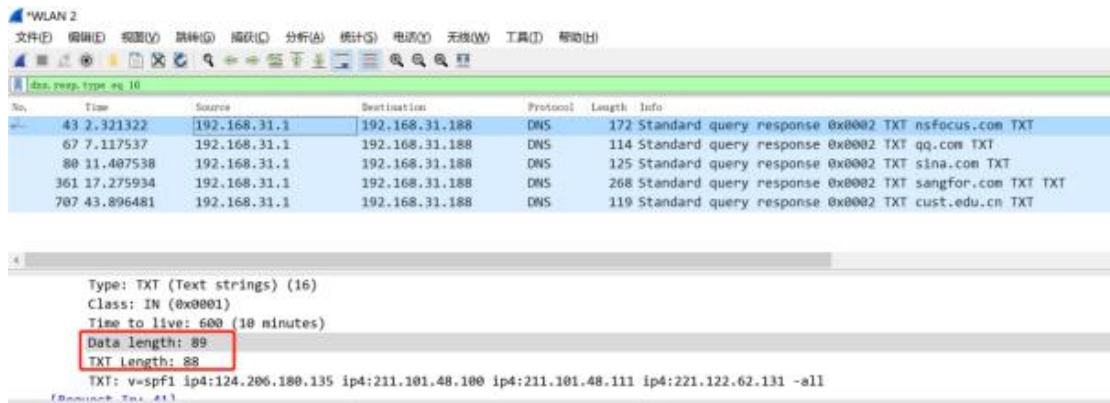
可以自定义 cs 的 profile 文件，set dns\_stager\_subhost ".feeds.123456.";去掉这个域名特征

190	dns-beacon {
191	# Options moved into "dns-beacon" group in version 4.3
192	set dns_idle "74.125.196.113"; #google.com (change this to match your campaign)
193	set dns_max_txt "252";
194	set dns_sleep "0"; # Force a sleep prior to each individual DNS request. (in milliseconds)
195	set dns_ttl "5";
196	set maxdns "255";
197	set dns_stager_prepend ".resources.123456.";
198	set dns_stager_subhost ".feeds.123456.";
199	}

再次发起 DNS Beacon 通信，观察 pcap 包，发现 TXT 类型的 DNS 请求，开头都以 aaa, baa, caa, daa...进行顺序请求



根据观察，正常 TXT 类型应答，数据部分长度在 0-100 字节之间，因此可尝试检测固定长度 255 的 TXT 类型应答的 DNS 响应包



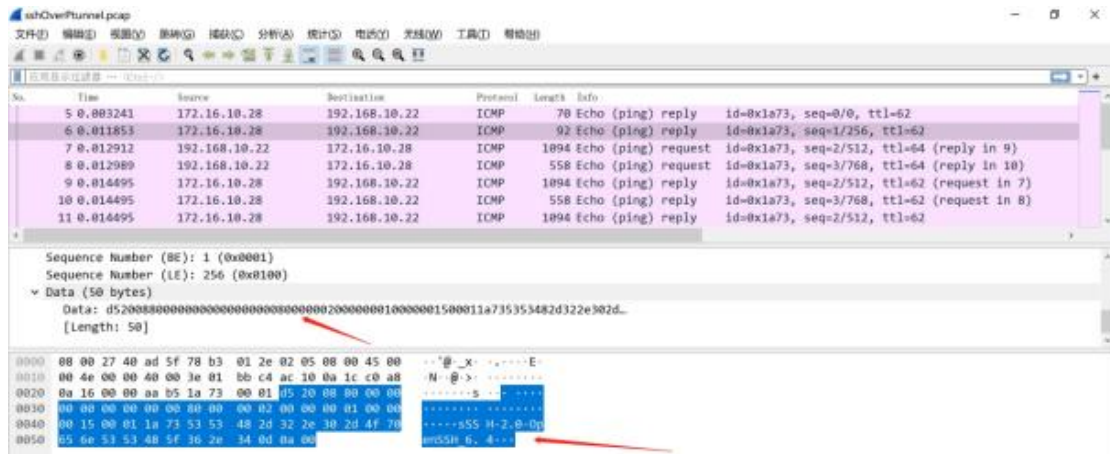
## 1.2 ICMP 隧道

### 1.2.1 pTunnel

ptunnel 特征非常明显，每次通信，ICMP 协议数据部分都会以 d520088000000000000000000400000020000ffff0000002000011a73 开头

ptunnel 不会限制 icmp 数据部分长度，传输时 icmp 数据部分会超过 64 字节

ptunnel 采用明文传输，可检测 icmp 协议中是否包含其他协议通信，例如 SSH

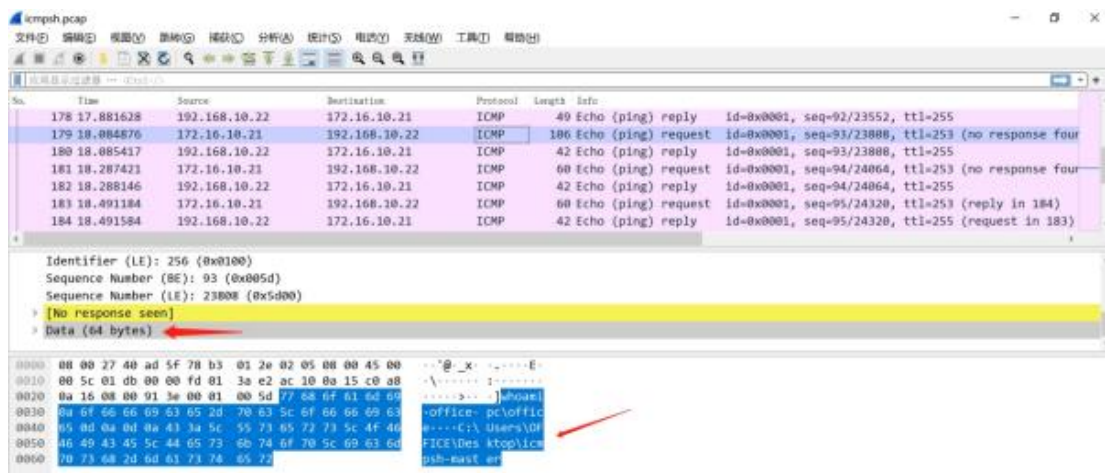
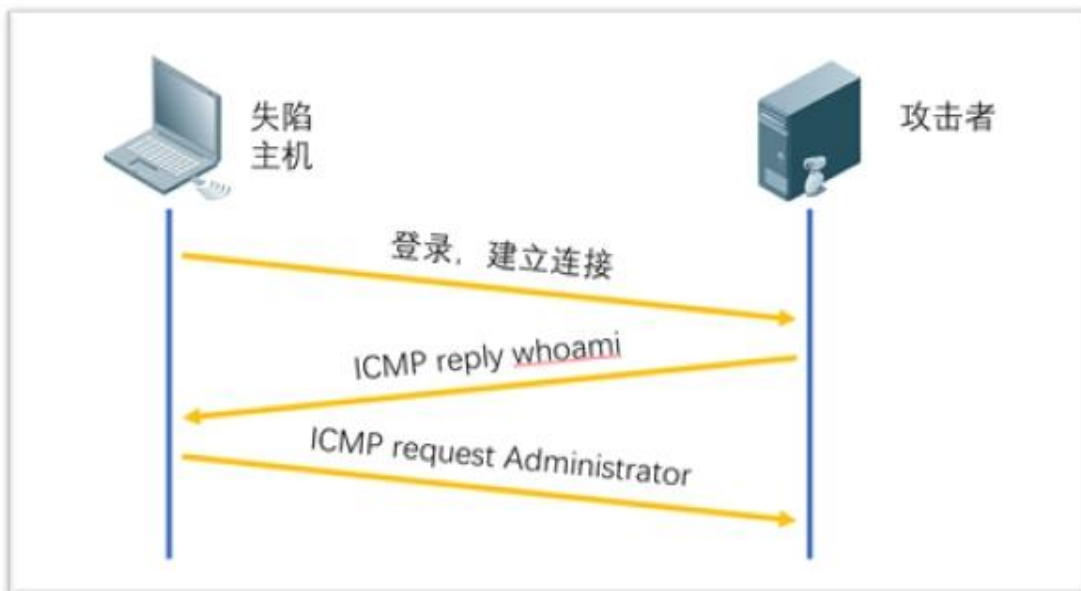


## 1.2.2 icmpsh

icmpsh 与常规 ICMP ping 通信顺序相反，控制端先发送 ICMP Reply，被控端再回复 ICMP Echo Request

icmpsh 每个包的数据部分最大长度不超过 64 字节

icmpsh 为明文传输，在 icmp 协议数据部分可以直接看到 whoami 等命令传输





## 2 代理通信

### 2.1. NPS

#### 2.1.1 NPS 简介

nps 是一款轻量级、高性能、功能强大的内网穿透代理服务器。目前支持 tcp、udp 流量转发，可支持任何 tcp、udp 上层协议（访问内网网站、本地支付接口调试、ssh 访问、远程桌面，内网 dns 解析等等……），此外还支持内网 http 代理、内网 socks5 代理、p2p 等，并带有功能强大的 web 管理端。

nps 默认配置文件使用了 80，443，8080，8024 端口，80 与 443 端口为域名解析模式默认端口。8080 为 web 管理访问端口。8024 为网桥端口，用于客户端与服务器通信。nps 得 web 管理默认用户名 admin，密码 123。默认客户端认证 vkey 也是 123。

#### 2.1.2 NPS 通信特征

通过 nps 源代码，发现 npc 客户端首次连接 nps 服务端时，特征较为明显，登录流程如下：

- 1、客户端发起测试连接，其中测试连接带有客户端版本信息
- 2、服务端接收客户端版本与本地版本进行比较，版本一致后，将服务端版本做 MD5 加密后发送给客户端

```
217 connection.SetDeadline(time.Now().Add(time.Second * 10))
218 defer connection.SetDeadline(time.Time{})
219 c := conn.NewConn(connection)
220 if _, err := c.Write([]byte(common.CONN_TEST)); err != nil : nil, err }
221 if err := c.WriteLenContent([]byte(version.GetVersion())); err != nil : nil, err }
222 if err := c.WriteLenContent([]byte(version.VERSION)); err != nil : nil, err }
223 b, err := c.GetShortContent(16384)
```

npc代码

- 3、客户端收到服务端 MD5 后与本地版本的 MD5 进行对比，一致后，将 vkey 进行 MD5 加密发送给服务端
- 4、服务端收到 MD5 后的 vkey，与本地数据进行对比，一致后，客户端在服务端验证成功

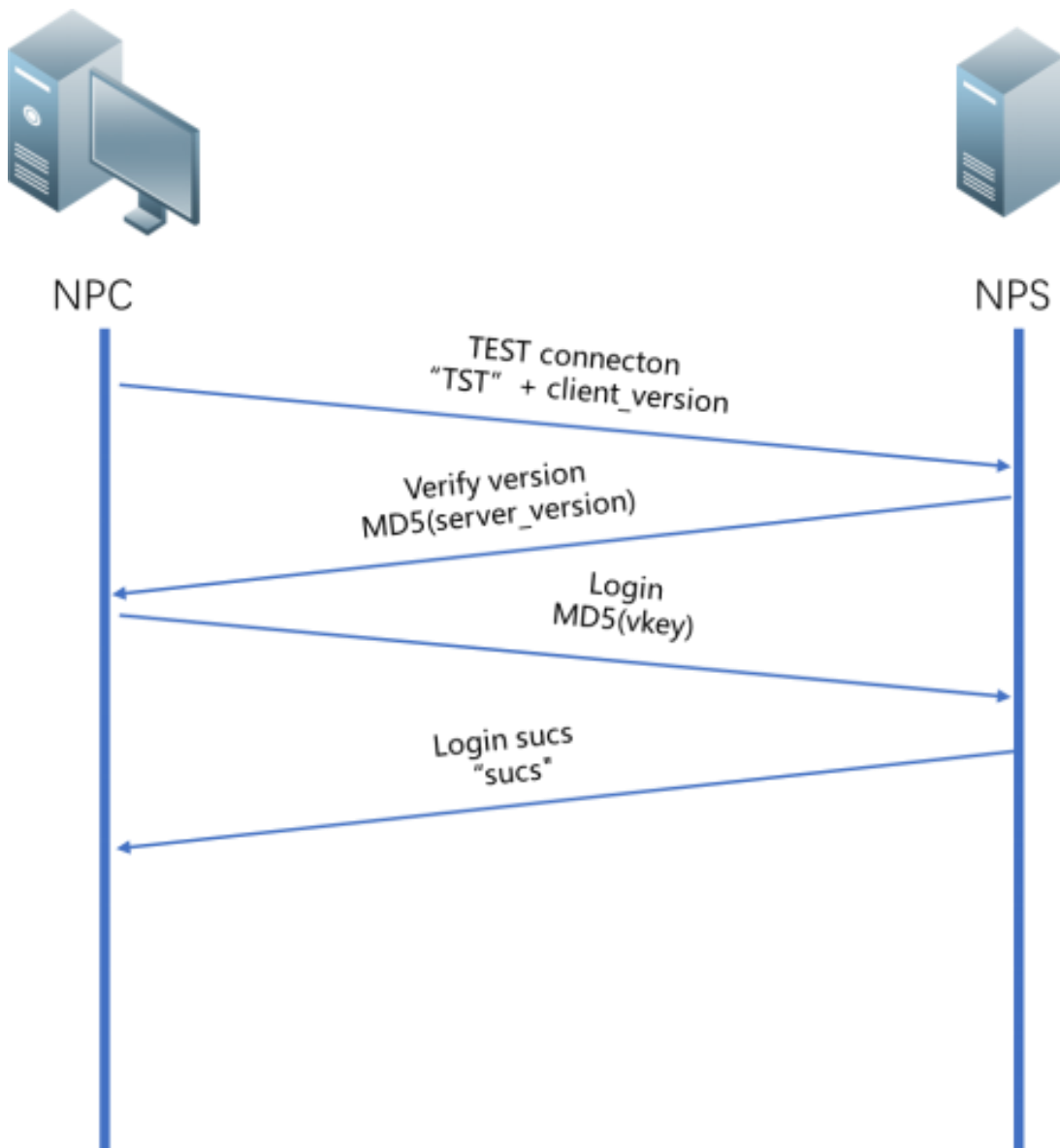
```

181     if vs, err = c.GetShortlenContent(); err != nil {
182         logs.Info("get client %s version error", err.Error())
183         c.Close()
184         return
185     }
186     //write server version to client
187     c.Write([]byte(crypt.Md5(version.GetVersion())))
188     c.SetReadDeadlineBySecond(5)
189     var buf []byte
190     //get vKey from client
191     if buf, err = c.GetShortContent(32); err != nil {
192         c.Close()
193         return
194     }
195     //verify
196     id, err := file.GetDb().GetIdByVerifyKey(string(buf), c.Conn.RemoteAddr().String())
197     if err != nil {
198         logs.Info("Current client connection validation error, close this client:", c.Conn.RemoteAddr())
199         s.verifyError(c)
200         return
201     } else {
202         s.verifySuccess(c)
203     }

```

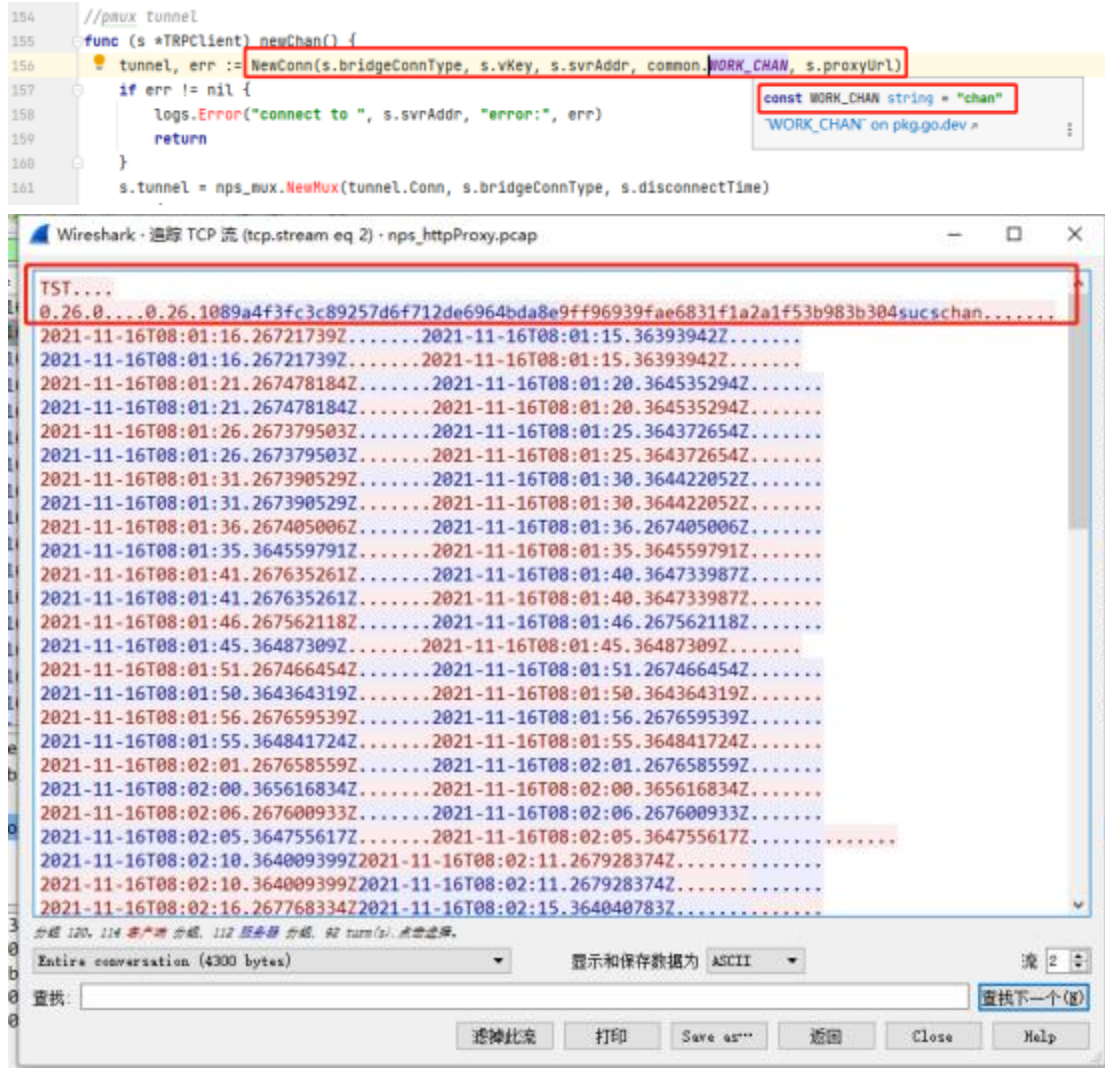
nps代码

具体过程如图所示





经过流量抓包和源码审计，发现 NPS 每次创建代理连接时，会单独建立 TCP 会话，并且客户端先进行登录验证，验证通过后，通过发送"chan"指令建立代理通道。



## 2.1.3 NPS 使用备忘

对于 linux|darwin

```
sudo ./nps install
```

对于 windows，管理员身份运行 cmd，进入安装目录

```
cd c:\nps\
.\nps.exe install
```

安装后 linux 和 darwin 位于/etc/nps，windows 配置文件位于 C:\Program Files\nps。日志位置，linux 下日志位于/var/log/nps.log，windows 下日志位于 nps.exe 同级目录下

```
root@kali:~/npsServer# ./nps install
2021/11/16 13:57:32 copy file ::/root/npsServer/conf/clients.json to /etc/nps/conf/clients.json
2021/11/16 13:57:32 copy file ::/root/npsServer/conf/hosts.json to /etc/nps/conf/hosts.json
2021/11/16 13:57:32 copy file ::/root/npsServer/conf/nps.conf to /etc/nps/conf/nps.conf
2021/11/16 13:57:32 copy file ::/root/npsServer/conf/server.key to /etc/nps/conf/server.key
2021/11/16 13:57:32 copy file ::/root/npsServer/conf/server.pem to /etc/nps/conf/server.pem
2021/11/16 13:57:32 copy file ::/root/npsServer/conf/tasks.json to /etc/nps/conf/tasks.json
2021/11/16 13:57:32 copy file ::/root/npsServer/web/views/client/add.html to /etc/nps/web/views/client/add.html
2021/11/16 13:57:32 mkdir:/etc/nps/web/views/client/
2021/11/16 13:57:32 copy file ::/root/npsServer/web/views/client/edit.html to /etc/nps/web/views/client/edit.html
2021/11/16 13:57:32 copy file ::/root/npsServer/web/views/client/list.html to /etc/nps/web/views/client/list.html
2021/11/16 13:57:32 copy file ::/root/npsServer/web/views/index/add.html to /etc/nps/web/views/index/add.html
2021/11/16 13:57:32 mkdir:/etc/nps/web/views/index/
2021/11/16 13:57:32 copy file ::/root/npsServer/web/views/index/edit.html to /etc/nps/web/views/index/edit.html
```

登录 web 管理端，新建一个客户端，获取 vkey 的值



客户端运行命令，指定 server 和 port，指定 vkey，在 web 界面可以看到客户端上线。

```
[root@localhost npcClient]# ./npc -server 192.168.243.11:8024 -vkey 33ptw4fatelet6ub
2021/11/16 02:24:07.262 [I] [npc.go:231] the version of client is 0.26.10, the core version of client is 0.26.0
2021/11/16 02:24:07.267 [I] [client.go:72] Successful connection with server 192.168.243.11:8024
```



执行./nps -version 查看版本

```
root@kali:~/npsServer# ./nps -version
Version: 0.26.10
Core version: 0.26.0
Same core version of client and server can connect each other
root@kali:~/npsServer#
```

服务端配置文件说明

参数名

参数值

web_port	web 管理端口
web_password	web 界面管理密码
web_username	web 界面管理账号
web_base_url	web 管理主路径,用于将 web 管理置于代理子路径后面
bridge_port	服务端客户端通信端口
https_proxy_port	域名代理 https 代理监听端口
http_proxy_port	域名代理 http 代理监听端口



auth_key	web api 密钥
bridge_type	客户端与服务端连接方式 kcp 或 tcp
public_vkey	客户端以配置文件模式启动时的密钥, 设置为空表示关闭客户端配置文件连接模式
ip_limit	是否限制 ip 访问, true 或 false 或忽略
flow_store_interval	服务端流量数据持久化间隔, 单位分钟, 忽略表示不持久化
log_level	日志输出级别
auth_crypt_key	获取服务端 authKey 时的 aes 加密密钥, 16 位
p2p_ip	服务端 ip, 使用 p2p 模式必填
p2p_port	p2p 模式开启的 udp 端口
pprof_ip	debug pprof 服务端 ip
pprof_port	debug pprof 端口
disconnect_timeout	客户端连接超时, 单位 5s, 默认值 60, 即 300s = 5mins

## 客户端配置文件说明

参数名                      参数值

server_addr	服务端 ip/域名:port
conn_type	与服务端通信模式(tcp 或 kcp)
vkey	服务端配置文件中的密钥(非 web)
username	socks5 或 http(s) 密码保护用户名(可忽略)
password	socks5 或 http(s) 密码保护密码(可忽略)
compress	是否压缩传输(true 或 false 或忽略)
crypt	是否加密传输(true 或 false 或忽略)
rate_limit	速度限制, 可忽略
flow_limit	流量限制, 可忽略
remark	客户端备注, 可忽略
max_conn	最大连接数, 可忽略
pprof_addr	debug pprof ip:port
mode	代理模式: tcp、udp、httpProxy、socks5、secret、p2p、file

## 2.2 FRP

### 2.2.1 FRP 简介

FRP 是一款使用 go 语言编写的，跨平台的流量代理转发工具。目前已知 FRP 支持的协议有 TCP、UDP、HTTP、HTTPS、SOCKS5。FRP 使用 CS 架构，内网主机中运行客户端向公网服务端发起连接，公网服务端开启代理端口供其他人使用。

### 2.2.2 FRP 通信特征

明文传输下的 FRP，特征非常明显。登陆请求：

```
{
  "version": "0.34.0",
  "hostname": "",
  "os": "linux",
  "arch": "amd64",
  "user": "",
  "privilege_key": "ce7709b1e457d1fab0f4a02eb1b00b04",
  "timestamp": 1548733905,
  "run_id": "",
  "metas": {},
  "pool_count": 1
}
```

登录失败响应

```
{
  "version": "0.34.0",
  "run_id": "",
  "server_udp_port": 0,
  "error": "token in login doesn't match token from configuration"
}
```

登录成功响应

```
{
  "version": "0.34.0",
  "run_id": "effc6af4bf48f89a",
  "server_udp_port": 0,
  "error": ""
}
```

默认情况下，FRPC 与 FRPS 之间每 10 秒进行一次存活探测交互，用来检测客户端是否掉线

internet(GE\_0-1).pcap

文件(F) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(Y) 无线(W) 工具(T) 帮助(H)

ip.src\_host eq 192.168.10.1 && top.dstport eq 58080 && top.flags.push eq 1

No.	Time	Source	Destination	Protocol	Length	Info
2949	10312.033795	192.168.10.1	175.20.30.2	TCP	78	46402 → 58080 [P
2956	10322.043958	192.168.10.1	175.20.30.2	TCP	78	46402 → 58080 [P
2957	10322.044265	192.168.10.1	175.20.30.2	TCP	109	46402 → 58080 [P
2963	10322.555834	192.168.10.1	175.20.30.2	TCP	78	46402 → 58080 [P
2964	10322.555890	192.168.10.1	175.20.30.2	TCP	109	46402 → 58080 [P
2972	10332.034466	192.168.10.1	175.20.30.2	TCP	78	46402 → 58080 [P
2976	10332.036521	192.168.10.1	175.20.30.2	TCP	78	46402 → 58080 [P
3042	10352.038049	192.168.10.1	175.20.30.2	TCP	78	46402 → 58080 [P
3045	10352.041113	192.168.10.1	175.20.30.2	TCP	78	46402 → 58080 [P
3048	10352.044300	192.168.10.1	175.20.30.2	TCP	78	46402 → 58080 [P
3049	10352.044355	192.168.10.1	175.20.30.2	TCP	109	46402 → 58080 [P

心跳检测，由 FRPC 发给 FRPS，内容如下：

```
{
  "privilege_key": "",
  "timestamp": 0
}
```

默认配置下的 FRP，使用 TLS 通信时，三次握手建立成功后，会发送一个单包，内容为 0x17，以此识别 FRP。

```
13 // limitations under the License.
14
15 package net
16
17 import (
18   "crypto/tls"
19   "fmt"
20   "net"
21   "time"
22
23   gnet "github.com/fatedier/golib/net"
24 )
25
26 var (
27   FRPTLSHeadByte = 0x17
28 )
29
30 func CheckAndEnableTLSServerConnWithTimeout(
```

应用层显示过截...

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.16.10.12	192.168.10.22	TCP	74	40042 → 58080 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1 TSval=832116992 TSecr=0 WS=128
2	0.000048	192.168.10.22	172.16.10.12	TCP	74	58080 → 40042 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 SACK_PERM=1 TSval=3689969137 TSecr=832116992
3	0.001182	172.16.10.12	192.168.10.22	TCP	60	40042 → 58080 [ACK] Seq=1 Ack=1 Win=65535 Len=0 TSval=832116992 TSecr=3689969137
4	0.001243	192.168.10.22	192.168.10.12	TCP	338	[TCP Reset] Seq=0 Len=0 RST=1 Seq=0
5	0.001545	172.16.10.12	192.168.10.22	TLSv1.3	309	Client Hello
6	0.001565	192.168.10.22	172.16.10.12	TCP	60	58080 → 40042 [ACK] Seq=1 Ack=2 Win=65535 Len=0 TSval=3689969138 TSecr=832116992
7	0.001579	192.168.10.22	172.16.10.12	TCP	60	58080 → 40042 [ACK] Seq=1 Ack=245 Win=65535 Len=0 TSval=3689969138 TSecr=832116992
8	0.003846	192.168.10.22	172.16.10.12	TLSv1.3	1485	Server Hello, Change Cipher Spec, Application Data, Application Data, Application Data
9	0.005825	172.16.10.12	192.168.10.22	TCP	60	40042 → 58080 [ACK] Seq=245 Ack=1420 Win=84128 Len=0 TSval=832116996 TSecr=3689969141
10	0.005396	172.16.10.12	192.168.10.22	TLSv1.3	330	Change Cipher Spec, Application Data
11	0.005404	192.168.10.22	172.16.10.12	TCP	60	58080 → 40042 [ACK] Seq=1420 Ack=300 Win=65535 Len=0 TSval=3689969142 TSecr=832116996

Frame 4: 67 bytes on wire (536 bits), 67 bytes captured (536 bits) on interface 0

Ethernet II, Src: 08:00:27:ad:5f:58, Dst: 08:00:27:ad:5f:58

Internet Protocol Version 4, Src: 172.16.10.12, Dst: 192.168.10.22

Transmission Control Protocol, Src Port: 40042, Dst Port: 58080, Seq: 1, Ack: 1, Len: 1

Transport Layer Security

```
0000 00 00 27 ad 5f 58 17 35 59 02 07 08 00 45 00 --[X]SY...E
0010 00 35 ad 19 40 00 3e 06 1b cf ac 10 0a c0 a8 5 @->.....
0020 0a 10 9c 6a e2 e0 51 0d bc 09 88 7e da 90 80 18 --[Q].....
0030 01 f0 50 f6 00 01 01 00 0a 31 90 19 00 0b f0 -p.....
0040 75 f1 17 .....
```



## 2.2.3 FRP 使用备忘

### 2.2.3.1 FRP 配置文件说明

FRPS 服务端配置文件

frps 配置文件分为 common 和 plugin 两个模块，common 模块下配置 frps 整体的参数信息：

参数	含义
<code>bind_addr = 0.0.0.0</code>	本地监听 TCP 地址
<code>bind_port = 7000</code>	本地监听 TCP 端口
<code>bind_udp_port = 7001</code>	本地监听 UDP 端口
<code>kcp_bind_port = 7000</code>	KCP 协议监听端口
<code>dashboard_addr = 0.0.0.0</code>	管理面板监听地址
<code>dashboard_port = 7500</code>	管理面板监听端口
<code>dashboard_user = admin</code>	管理面板用户名
<code>dashboard_pwd = admin</code>	管理面板密码
<code>enable_prometheus = true</code>	配置启用 prometheus 监控 frp
<code>log_file = ./frps.log</code>	日志文件位置
<code>log_level = info</code>	日志记录级别
<code>log_max_days = 3</code>	日志记录天数
<code>disable_log_color = false</code>	关闭 console 显示带颜色的日志
<code>detailed_errors_to_client = true</code>	是否发送详细错误给 client
<code>authentication_method = token</code>	认证客户端的方式
<code>authenticate_heartbeats = false</code>	认证心跳
<code>authenticate_new_work_conns = false</code>	认证新的会话
<code>token = 12345678</code>	认证 token 值
<code>oidc_client_id =</code>	oidc 认证下的 client id
<code>oidc_client_secret =</code>	oidc 认证下的 client secret
<code>oidc_audience =</code>	oidc 认证下的 audience



<code>oidc_token_endpoint_url =</code>	oidc 认证下的 URL
<code>heartbeat_timeout = 90</code>	心跳超时，默认 90 秒
<code>allow_ports = 2000-3000, 3001, 3003, 4000-50000</code>	允许客户端开启的代理端口
<code>max_pool_count = 5</code>	最大代理池数量
<code>max_ports_per_client = 0</code>	每个客户端最多开启的端口数
<code>tls_only = false</code>	只使用 TLS 协议通信
<code>subdomain_host = frps.com</code>	使用 http、https 模式时配置子域名
<code>tcp_mux = true</code>	使用 TCP 多路复用模式
<code>udp_packet_size = 1500</code>	UDP 数据包大小，单位是 byte

FRPC 客户端配置文件

common 模块

frpc 配置文件与 frps 配置文件相同，也分为 common 和 plugin 两个模块，common 模块下配置 frpc 整体的参数信息：

参数

含义

<code>server_addr = 0.0.0.0</code>	指定 frps 服务端地址
<code>server_port = 7000</code>	指定 frps 服务端端口
<code>http_proxy = http://user:passwd@192.168.1.128:8080</code>	如果需要使用代理连接到 frps 服务端，需要配置 http_proxy 参数
<code>log_file = ./frpc.log</code>	指定日志文件位置
<code>log_level = info</code>	指定日志记录级别
<code>log_max_days = 3</code>	指定日志最多保存天数
<code>disable_log_color = false</code>	关闭 console 显示带颜色的日志
<code>token = 12345678</code>	服务端使用 token 认证时需要配置 token 值
<code>admin_addr = 127.0.0.1</code>	Web 管理入口地址
<code>admin_port = 7400</code>	Web 管理入口端口
<code>admin_user = admin</code>	Web 管理入口用户名
<code>admin_pwd = admin</code>	Web 管理入口密码



<code>pool_count = 5</code>	代理池最大连接数
<code>tcp_mux = true</code>	使用 TCP 多路复用模式
<code>user = your_name</code>	标识代理用户
<code>login_fail_exit = true</code>	登录错误时结束进程
<code>protocol = tcp</code>	指定代理协议
<code>tls_enable = true</code>	是否使用 TLS 协议
<code>tls_cert_file = client.crt</code>	指定证书文件
<code>tls_key_file = client.key</code>	指定私钥文件
<code>tls_trusted_ca_file = ca.crt</code>	指定 CA 证书
<code>dns_server = 8.8.8.8</code>	代理 DNS 请求时指定 DNS 服务器
<code>start = ssh, dns</code>	需要启动的代理名称
<code>heartbeat_interval = 10</code>	心跳报文传输间隔
<code>heartbeat_timeout = 90</code>	心跳超时时间
<code>meta_var1 = 123</code>	自定义元数据
<code>udp_packet_size = 1500</code>	UDP 报文大小, 单位 byte

plugin 模块  
plugin 模块配置格式固定  
[proxy-name] #这里可以替换成任意名称, 但多个 client 不能重复, 因为名称冲突将不能上线  
type = tcp #指定代理使用的协议类型  
use\_compression = false #不使用报文压缩  
remote\_port = 6001 #指定在服务端开启的代理端口

### 2.2.3.2FRP 典型配置举例

#### 2.2.3.2. 使用 TCP 作为代理协议

##### FRPS 配置

```
[common]
bind_port = 58080
authentication_method = token
```

```
token = test_frp_58080
```

## FRPC 配置

```
[common]
server_addr = 175.20.30.2
server_port = 58080
token = test_frp_58080
[test-tcp]
type = tcp
local_addr = 127.0.0.1
local_port = 8080
remote_port = 40001
```

### 2.2.3.2. 使用 UDP 作为代理协议

## FRPS 配置

```
[common]
bind_port = 58080
authentication_method = token
token = test_frp_58080
```

## FRPC 配置

```
[common]
server_addr = 175.20.30.2
server_port = 58080
token = test_frp_58080
[test-udp]
type = udp
local_addr = 127.0.0.1
local_port = 8080
remote_port = 40001
```



### 2.2.3.2. 使用 HTTP 作为代理协议

#### FRPS 配置

```
[common]
bind_port = 58080
authentication_method = token
token = test_frp_58080
```

#### FRPC 配置

```
[common]
server_addr = 175.20.30.2
server_port = 58080
token = test_frp_58080
[test-http]
type = tcp
remote_port = 40001
plugin = http_proxy
```

### 2.2.3.2. 使用 TLS 作为代理协议

#### FRPS 配置

```
[common]
bind_port = 58080
authentication_method = token
token = test_frp_58080
tls_only = true
```

#### FRPC 配置

```
[common]
server_addr = 175.20.30.2
server_port = 58080
token = test_frp_58080
tls_enable = true
[test-http]
type = tcp
```

```
remote_port = 40001  
plugin = http_proxy
```

### 2.2.3.2. 使用 socks5 作为代理协议

#### FRPS 配置

```
[common]  
bind_port = 58080  
authentication_method = token  
token = test_frp_58080
```

#### FRPC 配置

```
[common]  
server_addr = 175.20.30.2  
server_port = 58080  
token = test_frp_58080  
[test-socks5]  
type = tcp  
remote_port = 40001  
plugin = socks5  
plugin_user = test  
plugin_passwd = pass
```

## 2.3 venom

### 2.3.1 venom 简介

使用 Go 语言开发，支持多级代理的内网穿透工具，支持正向和反向连接，支持部分业务的端口复用。

admin 节点，通常作为攻击者管理端

agent 节点，通常部署在已获取权限的服务器中

### 2.3.2 venom 通信特征

通过抓包观察及代码审计，发现 Venom 在应用层交互时，会通过第一次发包前 8 字节是否为“ABCDEFGH”，来判断是否为 Venom 流量。

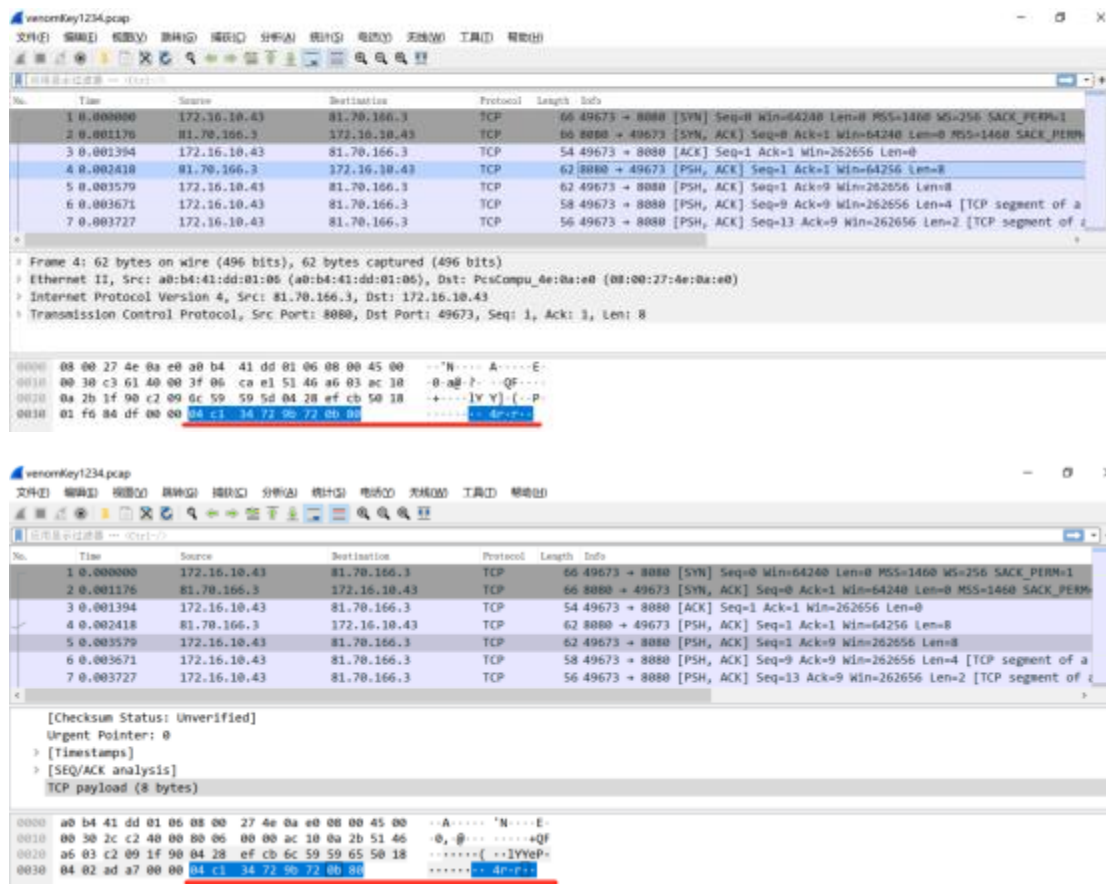
agent.go  agent\_iot.go  global.go 

```
13 const BUFFER_SIZE = 1024
14
15 // 协议数据分隔符
16 var PROTOCOL_SEPARATOR = "VCMD"
17
18 // 协议特征，用于在端口重用时鉴别
19 var PROTOCOL_FEATURE = "ABCDEFGH"
20
21 // 密钥
22 var SECRET_KEY []byte
23
```

Venom 通过启动时指定 `-passwd` 参数生成密钥，通信使用 AES/CTR 模式进行加密。流



量中没有明文传输那样明显的特征，不过 Venom 的 admin 端和 agent 端还是会固定发送 8 字节数据。



## 2.3.3 venom 使用备忘

反向连接

admin 监听端口，agent 发起连接：

```
./admin_macos_x64 -lport 9999
./agent_linux_x64 -rhost 192.168.0.103 -rport 9999
```

正向连接

agent 监听端口，admin 发起连接：

```
./agent_linux_x64 -lport 8888
./admin_macos_x64 -rhost 192.168.204.139 -rport 8888
```

admin 节点参数

Usage:

\$ ./venom\_admin -lport [port]

\$ ./venom\_admin -rhost [ip] -rport [port]

-lport

指定本地监听端口

-passwd	指定加密通信密钥
-rhost	指定 agent 节点 IP
-rport	指定 agent 节点端口

agent 节点参数

Usage:

```
$ ./venom_agent -lport [port]
$ ./venom_agent -rhost [ip] -rport [port]
$ ./venom_agent -lhost [ip] -reuse-port [port]
$ ./venom_agent -lport [port] -reuse-port [port]
```

**-lhost** 指定本地监听 IP

-lport	指定本地监听端口
-passwd	指定加密通信密钥
-rhost	指定 admin 节点 IP
-rport	指定 admin 节点端口
-reuse-port	指定端口复用的端口

## 2.4 FastTunnel

### 2.4.1 FastTunnel 简介

高性能跨平台内网穿透工具，使用它可以实现将内网服务暴露到公网供自己或任何人访问。

- 与其他穿透工具不同的是，FastTunnel 项目致力于打造一个易于扩展、易于维护的内网穿透框架。
- 你可以通过引用 FastTunnel.Core 的 nuget 包构建出自己的穿透应用，并针自己所需的业务扩展功能。

### 2.4.2 FastTunnel 通信特征

明文传输下的 FastTunnel 采用 http websocket 协议进行通信，特征如下图所示，HTTP 请求头中包含 FT\_VERSION、FT\_TOKEN 字段且 Connection 字段值为 Upgrade；HTTP 响应代码 101，且响应头 Server 字段值固定为 Kestrel

```
Wireshark · 追踪 HTTP 流 (tcp.stream eq 0) · fastTunnelConn.pcap
GET / HTTP/1.1
Host: 11.22.33.1:1270
FT_VERSION: 2.1.0.0
FT_TOKEN: 123456
Connection: Upgrade
Upgrade: websocket
Sec-WebSocket-Version: 13
Sec-WebSocket-Key: qUIYuc1vJES0bChBin2UMw==

HTTP/1.1 101 Switching Protocols
Connection: Upgrade
Date: Mon, 11 Jul 2022 09:37:16 GMT
Server: Kestrel
Upgrade: websocket
Sec-WebSocket-Accept: GlnxpobTifE4y0F67pmWSBrYos=
```

## 2.5 Termite

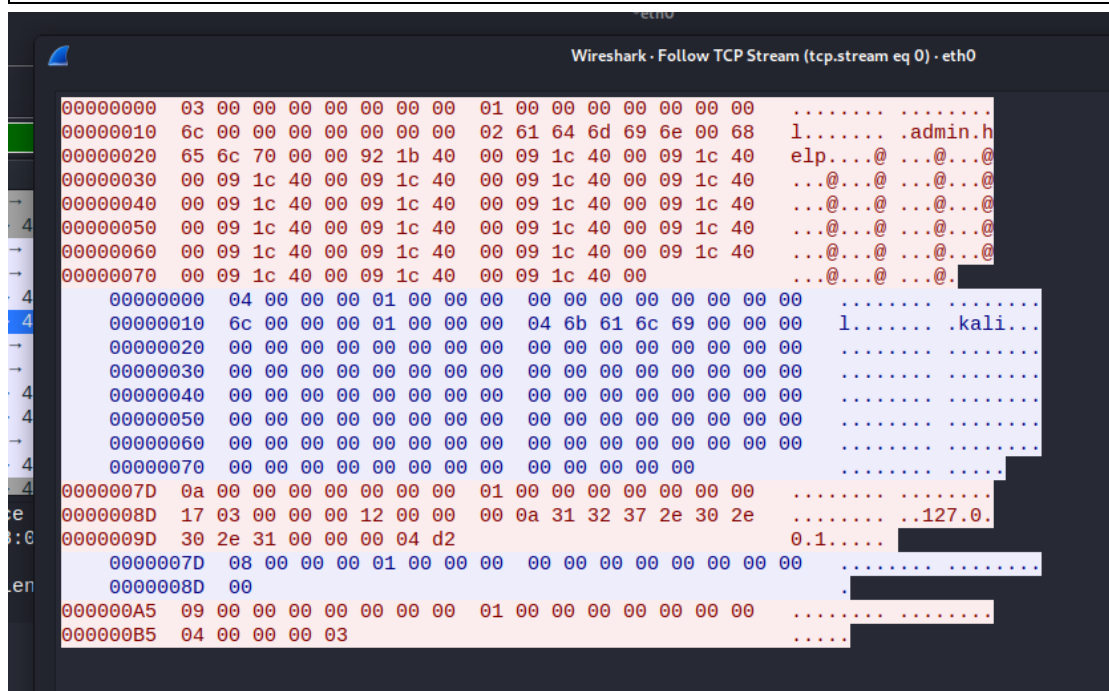
### 2.5.1 Termite 简介

Termite 是一款内网穿透利器，与 Venom 类似。分为管理端 admin 和代理端 agent。它支持多平台、跳板机间正反向级联、内置 shell 管理等。

### 2.5.2 Termite 通信特征

Termite 在 TCP 流中特征如下：

00 09 1c 40 00 09 1c 40



```
Wireshark · Follow TCP Stream (tcp.stream eq 0) · eth0

00000000 03 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00 .....
00000010 6c 00 00 00 00 00 00 00 02 61 64 6d 69 6e 00 68 l..... .admin.h
00000020 65 6c 70 00 00 92 1b 40 00 09 1c 40 00 09 1c 40 elp...@...@...@
00000030 00 09 1c 40 00 09 1c 40 00 09 1c 40 00 09 1c 40 ...@...@...@...@
00000040 00 09 1c 40 00 09 1c 40 00 09 1c 40 00 09 1c 40 ...@...@...@...@
00000050 00 09 1c 40 00 09 1c 40 00 09 1c 40 00 09 1c 40 ...@...@...@...@
00000060 00 09 1c 40 00 09 1c 40 00 09 1c 40 00 09 1c 40 ...@...@...@...@
00000070 00 09 1c 40 00 09 1c 40 00 09 1c 40 00 09 1c 40 ...@...@...@...@

00000000 04 00 00 00 01 00 00 00 00 00 00 00 00 00 00 00 .....
00000010 6c 00 00 00 01 00 00 00 04 6b 61 6c 69 00 00 00 l..... .kali...
00000020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

0000007D 0a 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00 .....
0000008D 17 03 00 00 00 12 00 00 00 0a 31 32 37 2e 30 2e ..... ..127.0.
0000009D 30 2e 31 00 00 00 04 d2 00 00 00 00 00 00 00 00 0.1.....

0000007D 08 00 00 00 01 00 00 00 00 00 00 00 00 00 00 00 .....
0000008D 00 .....

000000A5 09 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00 .....
000000B5 04 00 00 00 03 .....

```

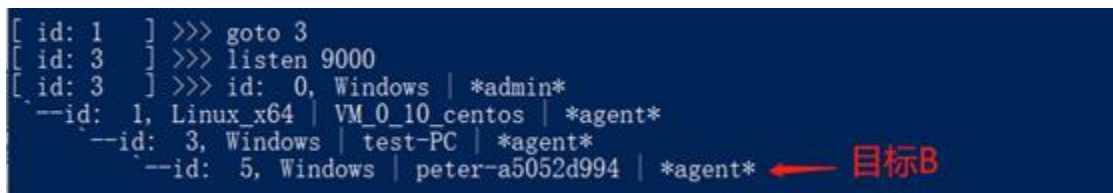
## 2.5.3 Termite 使用备忘

### 正向连接

```
agent_win32.exe -l 8888  
admin_win32.exe -c 目标 ip -p 8888
```

### 反向连接

```
在 vps 运行: agent_win32.exe -l 8888  
在 PC 运行: admin_win32.exe -c vps_ip -p 8888  
在出网机器 A 运行: agent_win32.exe -c vps_ip -p 8888  
在 PC 运行: goto A_id -> listen 9000  
在目标 B 运行: agent_win32.exe -c A_ip -p 9000
```



```
[ id: 1 ] >>> goto 3  
[ id: 3 ] >>> listen 9000  
[ id: 3 ] >>> id: 0, Windows | *admin*  
--id: 1, Linux_x64 | VM_0_10_centos | *agent*  
--id: 3, Windows | test-PC | *agent*  
--id: 5, Windows | peter-a5052d994 | *agent* ← 目标B
```

### admin 节点参数

```
VERSION : Free 1.2  
Eg:  
$ ./xxx -h  
$ ./xxx -c [rhost] -p [rport]  
-----  
options :  
-c tohost Remote host address.  
-p toport The port on remote host.  
-h help This help page.  
-v version Show the version.  
-a about Show the about text.  
-d detailed Show the detailed text.
```

### agent 节点参数

```
VERSION : Free 1.2  
Eg:  
$ ./xxx -h  
$ ./xxx -l [lport] -n [name]  
$ ./xxx -c [rhost] -p [rport] -n [name]  
-----  
options :  
-l listen Listen Mode.  
-c tohost Remote host address for `Connect Mode`.  
-p toport The port on remote host for `Connect Mode`.
```

```
-n name Setting the name ("agent" default).
-h help This help text.
-v version Version.
-a about About text.
-d detailed Detailed text.
```

## 2.6 Stowaway

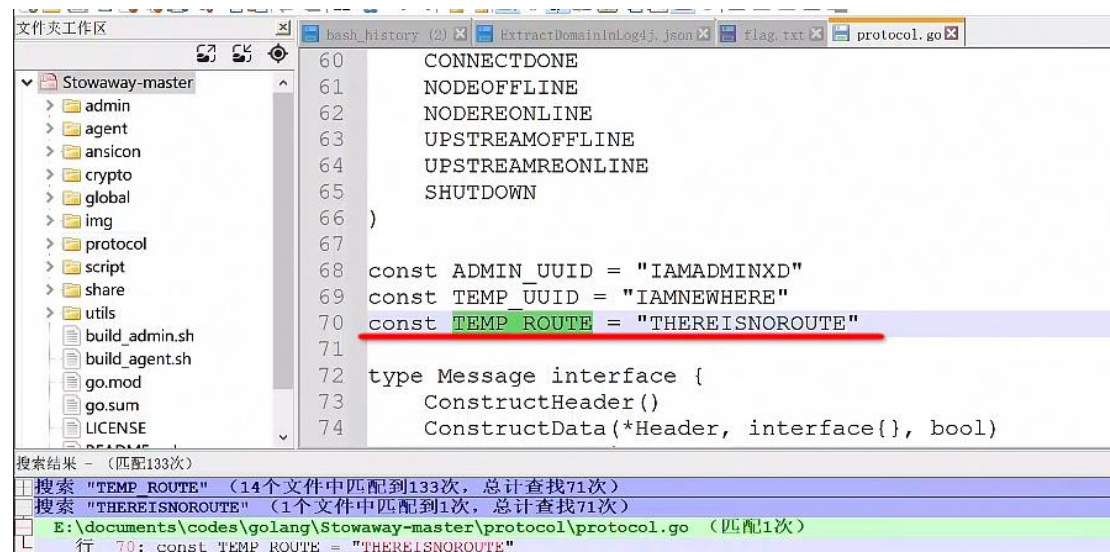
### 2.6.1 Stowaway 简介

Stowaway 是一个利用 go 语言编写、专为渗透测试工作者制作的多级代理工具

github 地址: <https://github.com/ph4ntonn/Stowaway>

### 2.6.2 Stowaway 通信特征

在 Stowaway-master\protocol\protocol.go 文件中, 存在变量 TEMP\_ROUTE:

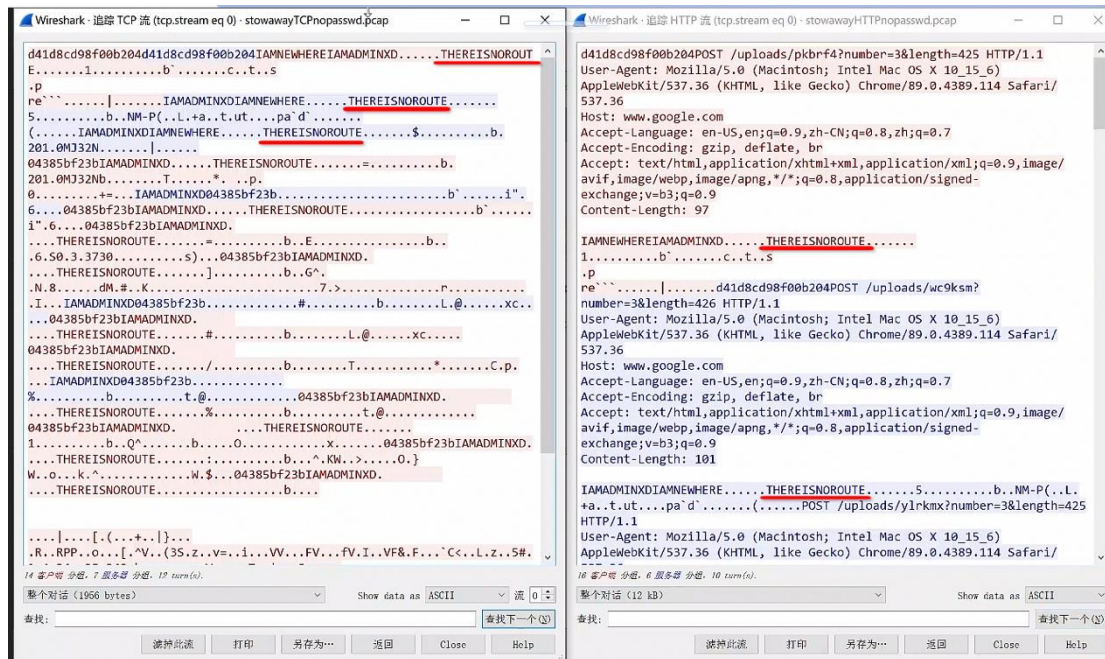


```
60    CONNECTDONE
61    NODEOFFLINE
62    NODEREONLINE
63    UPSTREAMOFFLINE
64    UPSTREAMREONLINE
65    SHUTDOWN
66 )
67
68 const ADMIN_UUID = "IAMADMINXD"
69 const TEMP_UUID = "IAMNEWHERE"
70 const TEMP_ROUTE = "THEREISNOROUTE"
71
72 type Message interface {
73     ConstructHeader()
74     ConstructData(*Header, interface{}, bool)
```

搜索结果 - (匹配133次)

- 搜索 "TEMP\_ROUTE" (14个文件中匹配到133次, 总计查找71次)
- 搜索 "THEREISNOROUTE" (1个文件中匹配到1次, 总计查找71次)
- E:\documents\codes\golang\Stowaway-master\protocol\protocol.go (匹配1次)
- 行 70: const TEMP\_ROUTE = "THEREISNOROUTE"

在流量中, admin 端与 agent 端通信中多次出现 TEMP\_ROUTE 变量:



## 2.6.3 Stowaway 使用备忘

### 角色

Stowaway 一共包含两种角色，分别是：

- admin 渗透测试者使用的主控端
- agent 渗透测试者部署的被控端

### 名词定义

- 节点：指 admin || agent
- 主动模式：指当前操作的节点主动连接另一个节点
- 被动模式：指当前操作的节点监听某个端口，等待另一个节点连接
- 上游：指当前操作的节点与其父节点之间的流量
- 下游：指当前操作的节点与其所有子节点之间的流量

### 参数解析

#### admin 端

- l 被动模式下的监听地址[ip]:<port>
- s 节点通信加密密钥,所有节点(admin&&agent)必须一致
- c 主动模式下的目标节点地址
- proxy socks5 代理服务器地址
- proxyu socks5 代理服务器用户名(可选)
- proxyp socks5 代理服务器密码(可选)
- down 下游协议类型,默认为裸 TCP 流量,可选 HTTP

#### agent 端



```
-l 被动模式下的监听地址[ip]:<port>
-s 节点通信加密密钥
-c 主动模式下的目标节点地址
--proxy socks5 代理服务器地址
--proxyu socks5 代理服务器用户名(可选)
--proxyp socks5 代理服务器密码(可选)
--reconnect 重连时间间隔
--rehost 端口复用复用的 IP 地址
--report 端口复用复用的端口号
--up 上游协议类型,默认为裸 TCP 流量,可选 HTTP
--down 下游协议类型,默认为裸 TCP 流量,可选 HTTP
--cs 运行平台的 shell 编码类型, 默认为 utf-8, 可选 gbk
```

#### -l

此参数 admin&&agent 用法一致, 仅用在被动模式下  
若不指定 IP 地址, 则默认监听在 0.0.0.0 上

```
admin: ./stowaway_admin -l 9999 or ./stowaway_admin -l 127.0.0.1:9999
agent: ./stowaway_agent -l 9999 or ./stowaway_agent -l 127.0.0.1:9999
```

#### -s

此参数 admin&&agent 用法一致, 可用在主动&&被动模式下  
可选, 若为空, 则代表通信不被加密, 反之则通信基于用户所给出的密钥加密

```
admin: ./stowaway_admin -l 9999 -s 123
agent: ./stowaway_agent -l 9999 -s 123
```

#### -c

此参数 admin&&agent 用法一致, 仅用在主动模式下  
代表了希望连接到的节点的地址

```
admin: ./stowaway_admin -c 127.0.0.1:9999
agent: ./stowaway_agent -c 127.0.0.1:9999
```

#### --proxy/--proxyu/--proxyp

这三个参数 admin&&agent 用法一致, 仅用在主动模式下  
--proxy 代表 socks5 代理服务器地址, --proxyu 以及 --proxyp 可选  
无用户名密码:

```
admin: ./stowaway_admin -c 127.0.0.1:9999 --proxy xxx.xxx.xxx.xxx
agent: ./stowaway_agent -c 127.0.0.1:9999 --proxy xxx.xxx.xxx.xxx
```

有用户名密码:

```
admin: ./stowaway_admin -c 127.0.0.1:9999 --proxy xxx.xxx.xxx.xxx --proxyu xxx --proxyp
xxx
agent: ./stowaway_agent -c 127.0.0.1:9999 --proxy xxx.xxx.xxx.xxx --proxyu xxx --proxyp
xxx
```

### --up/--down

这两个参数 admin&&agent 用法一致，可用在主动&&被动模式下

但注意 admin 上没有 --up 参数

这两个参数可选，若为空，则代表上/下游流量为裸 TCP 流量

若希望上/下游流量为 HTTP 流量，设置此两参数即可

**注意一点，当你设置了某一节点上/下游为 TCP/HTTP 流量后，与其连接的父/子节点的下/上游流量必须设置为一一致!!!**

```
admin: ./stowaway_admin -c 127.0.0.1:9999 --down http
agent: ./stowaway_agent -c 127.0.0.1:9999 --up http or ./stowaway_agent -c
127.0.0.1:9999 --up http --down http
```

### --reconnect

此参数仅用在 agent，且仅用在主动模式下

参数可选，若不设置，则代表节点在网络连接断开后不会主动重连，若设置，则代表节点会每隔 x(你设置的秒数)秒尝试重连至父节点

```
admin: ./stowaway_admin -l 9999
agent: ./stowaway_agent -c 127.0.0.1:9999 --reconnect 10
```

上面这种情况下，代表如果 agent 与 admin 之间的连接断开，agent 会每隔十秒尝试重连回 admin

agent 之间也与上面情况一致

并且 --reconnect 参数可以与 --proxy/--proxyu/--proxyp 一起使用，agent 将会参照启动时的设置，通过代理尝试重连

### --rehost/--report

这两个参数比较特别，仅用在 agent 端，详细请参见下方的端口复用机制

### --cs

此参数仅用在 agent，可用在主动&&被动模式下 主要旨在解决'shell'功能乱码问题，当用户将 agent 运行于控制台编码为 gbk 的平台上(例如一般情况下的 Windows)并且同时 admin 运行于控制台编码为 utf-8 的平台上时，请务必将此参数设置为'gbk'

```
Windows: ./stowaway_agent -c 127.0.0.1:9999 -s 123 --cs gbk
```

配置举例

单跳网络

启动 admin 端

```
./stowaway_admin -l 81.70.166.3:8080
```

启动 agent 端

```
./stowaway_agent -c 81.70.166.3:8080
```

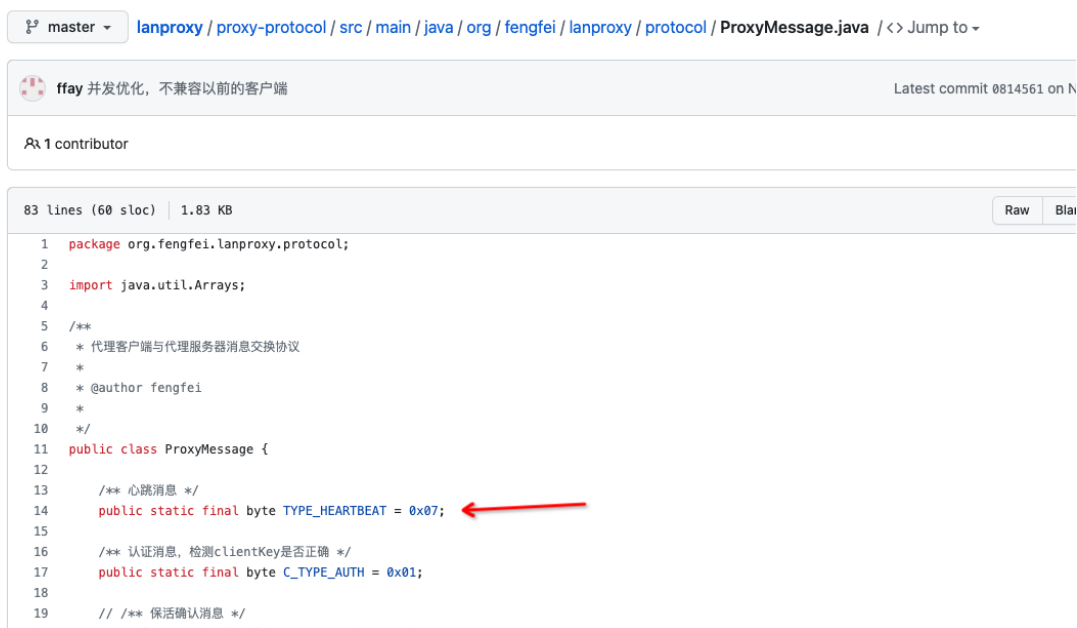
## 2.7 Lanproxy

### 2.7.1 Lanproxy 简介

lanproxy 是一个将局域网个人电脑、服务器代理到公网的内网穿透工具，支持 tcp 流量转发，可支持任何 tcp 上层协议（访问内网网站、本地支付接口调试、ssh 访问、远程桌面...）。目前市面上提供类似服务的有花生壳、TeamView、GoToMyCloud 等等，但要使用第三方的公网服务器就必须为第三方付费，并且这些服务都有各种各样的限制，此外，由于数据包会流经第三方，因此对数据安全也是一大隐患。

### 2.7.2 Lanproxy 通信特征

Lanproxy 心跳报文特征比较明显，在代码中可以看到心跳固定为 0x07

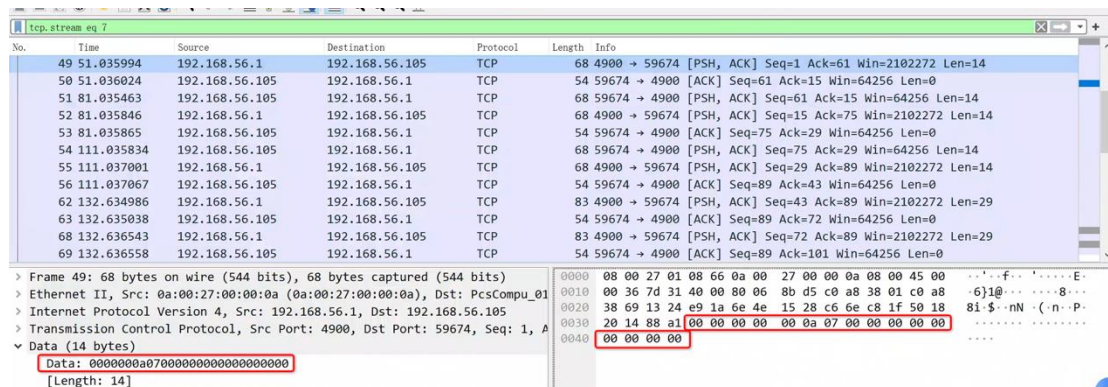


```

1 package org.fengfei.lanproxy.protocol;
2
3 import java.util.Arrays;
4
5 /**
6  * 代理客户端与代理服务器消息交换协议
7  *
8  * @author fengfei
9  *
10 */
11 public class ProxyMessage {
12
13     /** 心跳消息 */
14     public static final byte TYPE_HEARTBEAT = 0x07;
15
16     /** 认证消息，检测clientKey是否正确 */
17     public static final byte C_TYPE_AUTH = 0x01;
18
19     /** 保活确认消息 */
20     public static final byte TYPE_ACK = 0x02;
21 }

```

在 Wireshark 中查看如下：



当 Client 使用 SSL 通信时，SSL 指纹如下：

JA3: c3a6cf0bf2e690ac8e1ecf6081f17a50

JA3S: fbe78c619e7ea20046131294ad087f05

## 2.7.3 Lanproxy 使用备忘

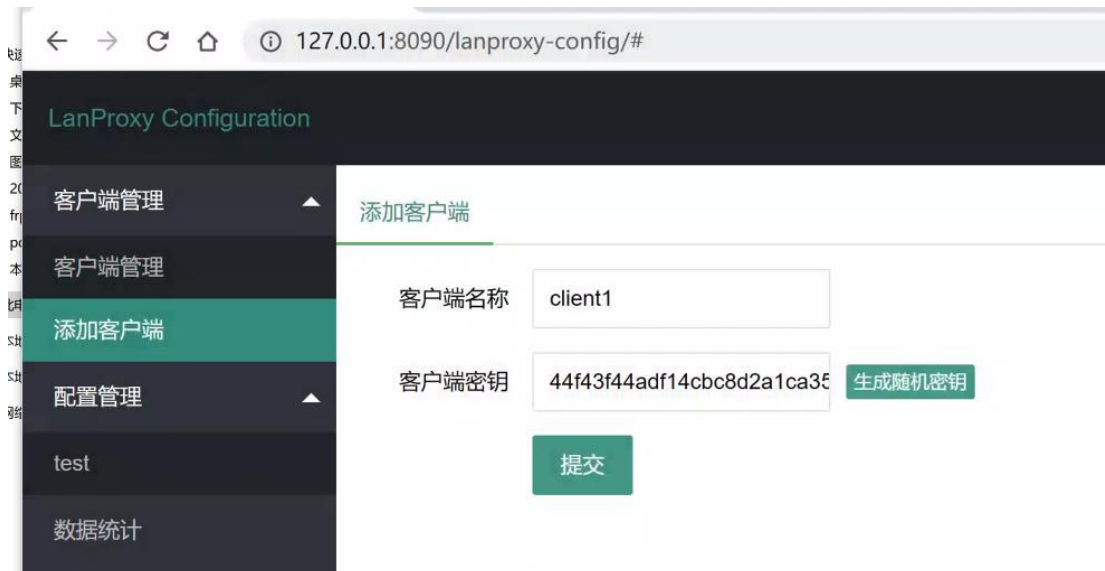
下载服务端:

<https://file.nioee.com/seafhttp/files/09f05ad2-d7f8-4401-afc9-65eb611e47d5/proxy-server-0.1.zip>

下载 amd64 版本 Linux 客户端:

<https://file.nioee.com/seafhttp/files/cf20ab75-bec8-40f4-a54a-beab6f00aed4/lanproxy-client-linux-amd64-20190523.tar.gz>

下载完成后, 打开服务端中\proxy-server-0.1\bin\下的 startup.bat, 服务端会默认监听 4900 端口, SSL 端口为: 4993, 管理端端口为: 8090, 默认用户名 admin, 密码 admin。直接访问 <http://127.0.0.1:8090> 登录管理端, 添加客户端, 生成客户端 Key



在客户端环境下运行, 客户端上线成功

```
./client_linux_amd64 -s 192.168.56.1 -p 4900 -k 44f43f44adf14cbc8d2a1ca35b37d510
```

## 2.8 长亭百川主机管理助手

### 2.8.1 简介

有别于其他相似的产品, 牧云主机管理助手致力于让技术爱好者能方便、快捷、优雅的管理远程主机, 而非傻瓜式管理, 主要优势在于:

- 连接远程主机不再受到内网 NAT 的限制



- 不用记密码，不用存私钥，认证都交给“微信登录”来托管
- 轻松解决性能监控问题
- 统一管理大批量主机

## 2.8.2 通信和终端行为特征

安装时需要请求 DNS: collie-agent.chaitin.com

安装完成后实时通信，通信协议使用 TLS1.3

client

JA3: 8bbd943d14de60c73f5f627d91caccb5

server

JA3S: 15af977ce25de452b96affa2addb1036

IP: 121.40.127.235

在/tmp 下新建 lighter-installer 文件

创建用户

collie:x:998:998::/opt/collie:/usr/sbin/nologin

创建进程

/opt/collie/collie daemon -w --service-type systemd --service-name collie --runtime-dir

/opt/collie --log-dir /var/log/co

\\_ /opt/collie/services/collie-agent/engine daemon --runtime-dir /opt/collie/services/collie-agent

## 2.8.3 使用备忘

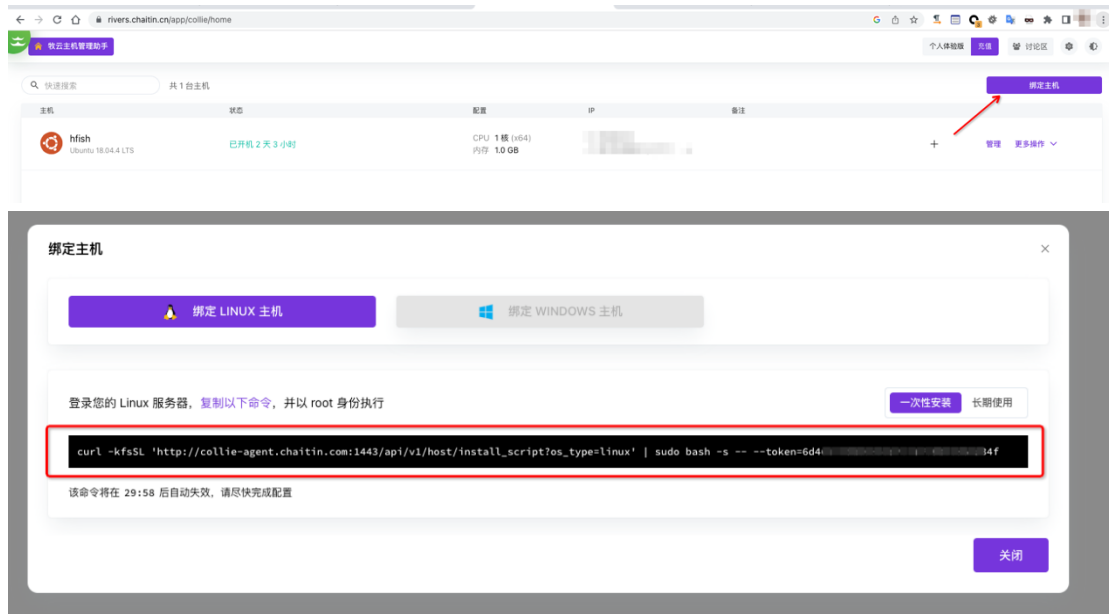
登录长亭百川平台

<https://rivers.chaitin.cn/>

开通主机管理助手产品



进入管理界面添加主机





```
root@suricata:~# curl -kfsSL 'http://collie-agent.chaitin.com:1443/api/v1/host/install_script?os_type=linux' | sudo bash -s -- --token   
sudo: unable to resolve host s: Name or service not known  
[INFO] Now downloading the Chaitin Collie lighter installer, please wait...  
[INFO] Chaitin Collie lighter installer has been downloaded successfully  
[1/6] Prepare > Prepare environment done  
[2/6] Check > Check done  
[3/6] Register > Register done  
[4/6] Release > Release all done  
[5/6] Adjust > Adjust done  
[6/6] Autostart > Startup done  
All done spend 2 seconds  
root@suricata:~#
```

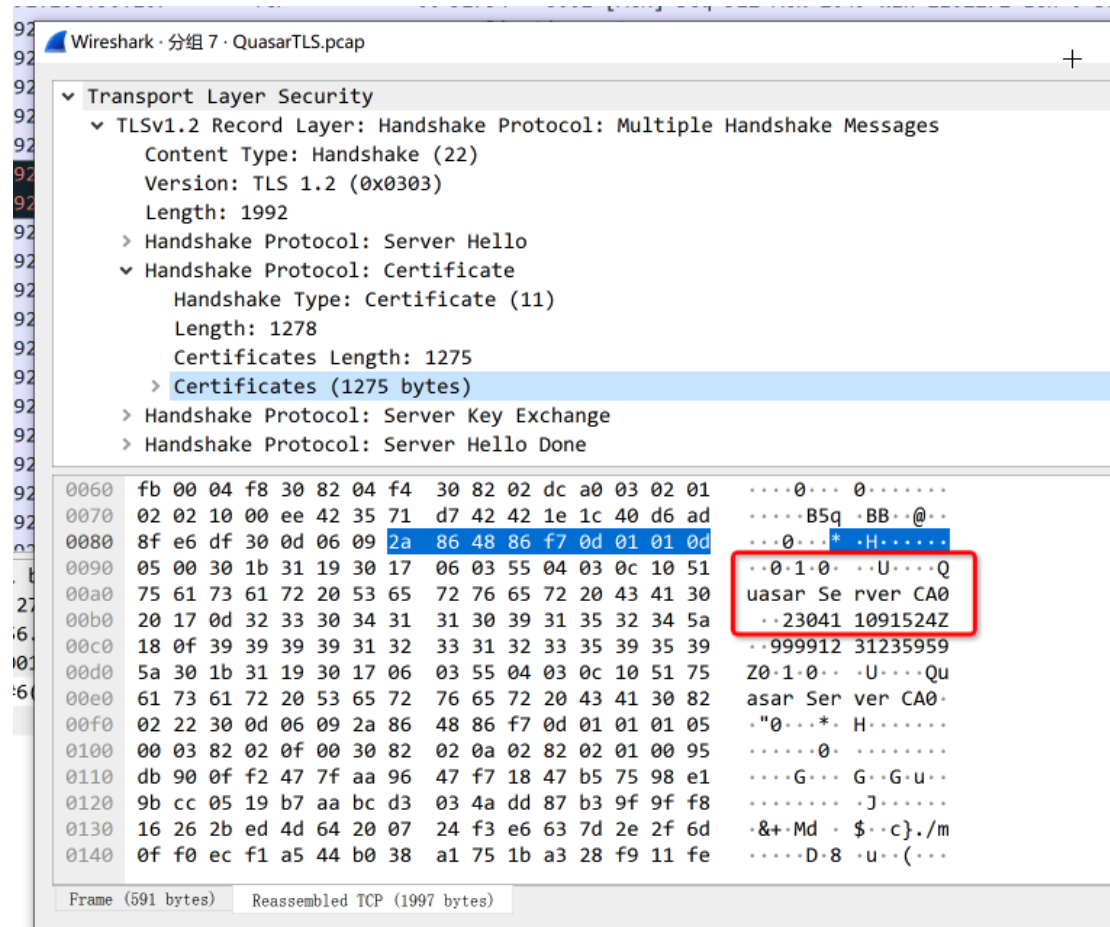
## 2.9 Quasar

### 2.9.1 Quasar 简介

Quasar 是一种公开可用的开源远程访问木马(RAT)，主要针对 Windows 操作系统，它通过恶意附件在网络钓鱼电子邮件中分发，该项目最初是由 GitHub 用户 MaxXor 开发，用于合法用途，然而该工具此后被黑客用于各种网络间谍活动  
<https://github.com/quasar/Quasar>

### 2.9.2 Quasar 通信特征

Server Hello 包中默认 TLS 证书颁发机构为 Quasar Server CA



## 2.9.3 Quasar 使用备忘

参考: <https://cloud.tencent.com/developer/article/2098666>

# 3 扫描渗透工具

## 3.1 RSAS

User-Agent

Mozilla/5.0 (Windows; U; Windows NT 6.1; zh-CN; rv:1.9.2.4) Gecko/20100611 Firefox/3.6.4



## 3.2 Goby

User-Agent

Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_14\_3) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/12.0.3 Safari/605.1.15

## 3.3 Xray

User-Agent

Mozilla/5.0 (Windows NT 10.0; rv:78.0) Gecko/20100101 Firefox/78.0

## 3.4 ez

User-Agent

Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36

## 3.5 fscan

User-Agent

Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/104.0.0.0 Safari/537.36

MS17-010 扫描

fscan 扫描 MS17-010, 登录时会使用用户名: anonymous

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.56.102	192.168.56.101	TCP	74	38388 → 445 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3...
2	0.000341	192.168.56.101	192.168.56.102	TCP	74	445 → 38388 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 ...
3	0.000357	192.168.56.102	192.168.56.101	TCP	66	38388 → 445 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3539813489 TSe...
4	0.000413	192.168.56.102	192.168.56.101	SMB	203	Negotiate Protocol Request
5	0.000886	192.168.56.101	192.168.56.102	SMB	177	Negotiate Protocol Response
6	0.000896	192.168.56.102	192.168.56.101	TCP	66	38388 → 445 [ACK] Seq=138 Ack=112 Win=64256 Len=0 TSval=3539813489...
7	0.000935	192.168.56.102	192.168.56.101	SMB	206	Session Setup AndX Request, User: anonymous
8	0.001569	192.168.56.101	192.168.56.102	SMB	255	Session Setup AndX Response
9	0.001657	192.168.56.102	192.168.56.101	SMB	166	Tree Connect AndX Request, Path: \\192.168.175.128\IPC\$
10	0.001883	192.168.56.101	192.168.56.102	SMB	126	Tree Connect AndX Response
11	0.002021	192.168.56.102	192.168.56.101	SMB Pipe	144	PeekNamedPipe Request, FID: 0x0000
12	0.002286	192.168.56.101	192.168.56.102	SMB	105	Trans Response, Error: STATUS_INSUFF_SERVER_RESOURCES
13	0.002331	192.168.56.102	192.168.56.101	SMB	148	Trans2 Request, SESSION_SETUP
14	0.002745	192.168.56.101	192.168.56.102	SMB	105	Trans2 Response, SESSION_SETUP, Error: STATUS_NOT_IMPLEMENTED
15	0.002926	192.168.56.102	192.168.56.101	TCP	66	38388 → 445 [FIN, ACK] Seq=538 Ack=439 Win=64128 Len=0 TSval=35398...

## 3.6 Nmap

nmap 扫描 MS17-010, 登录时会使用用户名: WIN7\guest



No.	Time	Source	Destination	Protocol	Length	Info
2	0.000327	192.168.56.101	192.168.56.102	TCP	74	445 → 36760 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 ...
3	0.000339	192.168.56.102	192.168.56.101	TCP	66	36760 → 445 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1578680715 TSe...
4	0.000416	192.168.56.102	192.168.56.101	SMB	119	Negotiate Protocol Request
5	0.000513	192.168.56.101	192.168.56.102	SMB	197	Negotiate Protocol Response
6	0.000575	192.168.56.102	192.168.56.101	TCP	66	36760 → 445 [ACK] Seq=54 Ack=132 Win=64128 Len=0 TSval=1578680723 ...
7	0.000895	192.168.56.102	192.168.56.101	SMB	215	Session Setup AndX Request, NTLMSSP_NEGOTIATE
8	0.000935	192.168.56.101	192.168.56.102	SMB	334	Session Setup AndX Response, NTLMSSP_CHALLENGE, Error: STATUS_MORE...
9	0.011558	192.168.56.102	192.168.56.101	SMB	315	Session Setup AndX Request, NTLMSSP_AUTH, User: WIN7\guest
10	0.014536	192.168.56.101	192.168.56.102	SMB	105	Session Setup AndX Response, Error: STATUS_ACCOUNT_DISABLED
11	0.014731	192.168.56.102	192.168.56.101	SMB	215	Session Setup AndX Request, NTLMSSP_NEGOTIATE
12	0.015080	192.168.56.101	192.168.56.102	SMB	334	Session Setup AndX Response, NTLMSSP_CHALLENGE, Error: STATUS_MORE...
13	0.015333	192.168.56.102	192.168.56.101	SMB	246	Session Setup AndX Request, NTLMSSP_AUTH, User: \
14	0.015622	192.168.56.101	192.168.56.102	SMB	184	Session Setup AndX Response
15	0.015718	192.168.56.102	192.168.56.101	SMB	141	Tree Connect AndX Request, Path: \\192.168.56.101\IPC\$
16	0.015963	192.168.56.101	192.168.56.102	SMB	116	Tree Connect AndX Response

Nmap

## 3.7 Metasploit

Metasploit 扫描 MS17-010，登录时会使用用户名：.\

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.56.102	192.168.56.101	TCP	74	44799 → 445 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3...
2	0.000550	192.168.56.101	192.168.56.102	TCP	74	445 → 44799 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 ...
3	0.000576	192.168.56.102	192.168.56.101	TCP	66	44799 → 445 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3540123939 TSe...
4	0.002441	192.168.56.102	192.168.56.101	SMB	154	Negotiate Protocol Request
5	0.003034	192.168.56.101	192.168.56.102	SMB	197	Negotiate Protocol Response
6	0.003050	192.168.56.102	192.168.56.101	TCP	66	44799 → 445 [ACK] Seq=89 Ack=132 Win=64128 Len=0 TSval=3540123942 ...
7	0.005041	192.168.56.102	192.168.56.101	SMB	213	Session Setup AndX Request, NTLMSSP_NEGOTIATE
8	0.005701	192.168.56.101	192.168.56.102	SMB	303	Session Setup AndX Response, NTLMSSP_CHALLENGE, Error: STATUS_MORE...
9	0.007910	192.168.56.102	192.168.56.101	SMB	398	Session Setup AndX Request, NTLMSSP_AUTH, User: .\
10	0.008827	192.168.56.101	192.168.56.102	SMB	105	Session Setup AndX Response, Error: STATUS_LOGON_FAILURE
11	0.009684	192.168.56.102	192.168.56.101	SMB	169	Session Setup AndX Request, User: .\
12	0.009951	192.168.56.101	192.168.56.102	SMB	183	Session Setup AndX Response
13	0.012468	192.168.56.102	192.168.56.101	SMB	142	Tree Connect AndX Request, Path: \\192.168.56.101\IPC\$
14	0.012617	192.168.56.101	192.168.56.102	SMB	116	Tree Connect AndX Response

Metasploit