ph4ntonn/Stowaway

github.com/ph4ntonn/Stowaway

ph4ntonn

ph4ntonn/
Stowaway

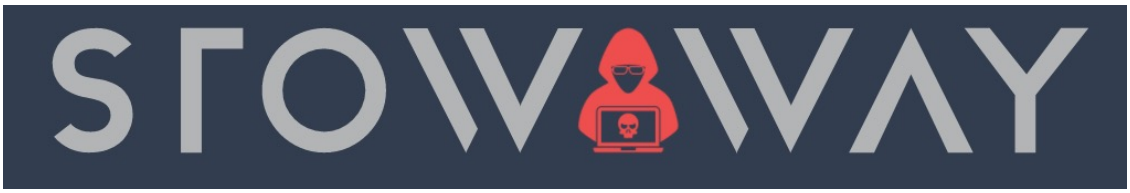
Stowaway – Multi-hop Proxy Tool for pentesters

5 Contributors | 7 Issues | 2k Stars | 328 Forks

master

Name already in use

A tag already exists with the provided branch name. Many Git commands accept both tag and branch names, so creating this branch may cause unexpected behavior. Are you sure you want to create this branch?



Stowaway

English

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

用户可使用此程序将外部流量通过多个节点代理至内网，突破内网访问限制，构造树状节点网络，并轻松实现管理功能

PS:谢谢大家的star，同时欢迎大家使用后提出问题&&Bug 😘。

PPS:**请务必在使用前详细阅读使用方法及文末的注意事项**

> 此工具仅限于安全研究和教学，用户承担因使用此工具而导致的所有法律和相关责任！ 作者不承担任何法律和相关责任！

特性

- 管理端更加友好的交互,支持命令补全/历史
- 一目了然的节点树管理
- 丰富的节点信息展示
- 节点间正向/反向连接
- 节点间支持重连
- 节点间可通过socks5代理进行连接
- 节点间可通过ssh隧道连接
- 节点间流量可选择TCP/HTTP
- 多级socks5流量代理转发,支持UDP/TCP,IPV4/IPV6
- 节点支持ssh访问远程主机
- 远程shell
- 上传及下载文件
- 端口本地/远程映射
- 节点可端口复用
- 自由开关各类服务
- 节点间相互认证
- 节点间流量以AES-256-GCM进行加密
- 相较于v1.0，文件体积减小25%
- 支持各类平台(Linux/Mac/Windows/MIPS/ARM)

下载及演示

- 不想编译的盆油可以直接用release下编译完成的程序
- 演示视频：Youtube

使用方法

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

```
参数:
-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

```
参数:
-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 端口复用时复用的IP地址
--report 端口复用时复用的端口号
--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
```

参数用法

-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

- 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`

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致！！！

如下

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

上面这种情况，agent必须设置`--up`为http，否则会导致网络出错

agent间也一样

假设agent-1正在`127.0.0.1:10000`端口上等待子节点的连接，并且设置了`--down http`

那么agent-2也必须设置`--up`为http，否则会导致网络出错

agent-2: `./stowaway_agent -c 127.0.0.1:10000 --up 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`

端口复用机制

当前Stowaway提供基于SO_REUSEPORT和SO_REUSEADDR特性的端口复用功能及基于IPTABLES的端口复用功能

- 在linux下可以大部分的功能端口
- 在windows下不可复用iis，rdp端口，可以复用mysql，apache服务的端口

复用方式

- SO_REUSEPORT和SO_REUSEADDR模式

  假设agent端采用端口复用机制复用80端口

  此时agent端必须设置`--rehost`&&`--report`&&`-s`参数

  - --rehost代表希望复用的IP地址，不可为0.0.0.0，普遍应当是网卡的外部地址
  - --report代表希望复用的端口
  - -s代表通信密钥

  主要支持windows、mac环境下的复用，linux亦可，但限制较多

  - admin端：`./stowaway_admin -c 192.168.0.105:80 -s 123`
  - agent端： `./stowaway_agent --report 80 --rehost 192.168.0.105 -s 123`

- IPTABLES模式

  假设agent端采用端口复用机制复用22端口

  此时agent端必须设置`-l`&&`--report`&&`-s`参数

  - -l 代表无法被正常访问的端口，也就是你真正想让agent监听并接受连接的端口
  - --report代表希望复用的端口
  - -s代表通信密钥

  仅支持linux环境下的复用，agent会自动修改IPTABLES，需要root权限

  - agent端： `./stowaway_agent --report 22 -l 10000 -s 123`

    在agent启动后，请使用`script`目录下的`reuse.py`

    先设置SECRET的值(SECRET的值就是在启动各个节点时所设置的通信密钥),

    之后执行：`python reuse.py --start --rhost xxx.xxx.xxx.xxx --rport xxx`

    - --rhost代表agent的地址
    - --rport代表被复用的端口,在本例中应当为22

  - 此时admin端就可以连接：`./stowaway_admin -c 192.168.0.105:22 -s 123`

注意

- 以上情况只是列举了admin以及agent之间的连接，agent与agent之间的连接亦同，并无差异
- 如果agent被ctrl-c或者kill命令杀死，程序将会自动清理iptables规则，但如果被kill -9 杀死，则无法自动清除

  故而为了防止agent异常退出后，iptables规则没有被清理导致被复用的服务无法访问

  所以当需要关闭时，需运行：`python reuse.py --stop --rhost xxx.xxx.xxx.xxx --rport xxx`

  即可关闭转发规则，使得原服务能够被正常访问

- 如果使用IPTABLES模式下的端口复用模式，将会强制监听在0.0.0.0，无法由`-l`参数来指定ip

如何组成多级网络？

从上面的例子可以看到，只有admin和一个agent出场

而多级网络才是核心

在stowaway中，组成多级网络需要借助admin中的`listen`、`connect`、`sshtunnel`命令来实现

举一个简单的例子

> admin: `./stowaway_admin -l 9999 -s 123`

此时agent-1已经连上admin

> agent-1: `./stowaway_agent -c 127.0.0.1:9999 -s 123`

此时用户还想连接agent-2，如下

> agent-2: `./stowaway_agent -l 10000 -s 123`

那么，此时用户可以通过admin,输入`use 0` -> `connect agent-2的IP:10000`来将其加入网络，并成为agent-1的一个子节点

假如此时用户还希望连入一个节点agent-3，但是通过agent-1无法访问agent-3

那么，此时用户可以通过admin,输入`use 0` -> `listen` -> 选择`1.Normal Passive` -> 输入`10001` 从而使得agent-1监听在10001端口上，并等待子节点的连接

等admin操作完成后，agent-3启动如下

> agent-3: `./stowaway_agent -c 127.0.0.1:10001 -s 123`

就可以将agent-3作为agent-1的另一个子节点加入网络了

关于`listen`以及`sshtunnel`的详细介绍，可以参看下方的命令解析

如何重连？

Stowaway当前支持多种方式的重连，简单概括如下

首先，当父节点掉线后，只有一种节点会主动退出，那就是启动时为主动模式且没有设置重连的节点

如果设置了重连，那么节点将会在指定的时间间隔中尝试重连

另外，所有被动模式启动的节点都不会主动退出，而是会基于启动时的参数重新监听在指定端口上，此时用户仍然可以通过`connect`、`sshtunnel`来将这些节点连回网络

注意

1. **如因网络波动或中间节点掉线，导致某一个分支断开，在主动重连时请务必连接缺失链的头节点 举个例子，admin后接着node1，node1后分为两支，一支是node1->node 2 -> node 3 -> node 4, 一支是node1->node 5 ->node 6，那么如果node2掉线，node3及node4将不会掉线，而是继续保持存活。此时用户若想将node3及node4重新加入网络，那么用户有两种选择，一种是假如node1可以直接访问node3，那么用户可随时在node1将node3用connect或者sshtunnel命令重新加入网络（切记，就算node1同时也可以访问node4，也请不要直接连接node4，请连接整个缺失链(node3->node4)的头节点node3），这样就可以将node3及node4重新加入网络；另一种选择是当node1无法直接访问node3时（即必须经过node2），那么请先将node2重启并加入网络，之后再在node2上使用connect或者sshtunnel命令连接node3，从而将node3及node4加入网络。**
2. **当有节点掉线时，那么此时与此节点及其子节点有关的所有socks，backward，forward服务都会被强制停止**

命令解析

在admin控制台中，用户可以用tab来补全命令，方向键上下左右来查找历史/移动光标

admin控制台分为两个层级，第一层为主panel，包含的命令如下

`help`: 展示主panel的帮助信息

```
(admin) >> help
  help                                                  Show help
information
  detail                                                Display connected
nodes' detail
  topo                                                  Display nodes'
topology
  use        <id>                                       Select the target
node you want to use
  exit                                                  Exit Stowaway
```

`detail`: 展示在线节点的详细信息

```
(admin) >> detail
Node[0] -> IP: 127.0.0.1:10000  Hostname: ph4ntoms-MBP.lan  User: ph4ntom
Memo:
```

`topo`: 展示在线节点的父子关系

```
(admin) >> topo
Node[0]'s children ->
Node[1]

Node[1]'s children ->
```

`use`: 使用某个agent

```
(admin) >> use 0
(node 0) >>
```

`exit`: 退出stowaway

```
(admin) >> exit
[*] Do you really want to exit stowaway?(y/n): y
[*] BYE!
```

当用户使用`use`命令选择了一个agent后，进入第二层node panel，其包含的命令如下

`help`: 展示node panel的帮助信息

```
(node 0) >> help
  help                                              Show help information
  listen                                            Start port listening on
current node
  addmemo    <string>                               Add memo for current
node
  delmemo                                           Delete memo of current
node
  ssh        <ip:port>                              Start SSH through
current node
  shell                                             Start an interactive
shell on current node
  socks      <lport> [username] [pass]              Start a socks5 server
  stopsocks                                         Shut down socks
services
  connect    <ip:port>                              Connect to a new node
  sshtunnel  <ip:sshport> <agent port>              Use sshtunnel to add
the node into our topology
  upload     <local filename> <remote filename>     Upload file to current
node
  download   <remote filename> <local filename>     Download file from
current node
  forward    <lport> <ip:port>                      Forward local port to
specific remote ip:port
  stopforward                                       Shut down forward
services
  backward    <rport> <lport>                       Backward remote
port(agent) to local port(admin)
  stopbackward                                      Shut down backward
services
  shutdwon                                          Terminate current node
  back                                              Back to parent panel
  exit                                              Exit Stowaway
```

`listen`: 命令agent监听某个端口并等待子节点的连入

```
(node 0) >> listen
[*] MENTION! If you choose IPTables Reuse or SOReuse,you MUST CONFIRM
that the node was initially started in the corresponding way!
[*] When you choose IPTables Reuse or SOReuse, the node will use the
initial config(when node started) to reuse port!
[*] Please choose the mode(1.Normal passive / 2.IPTables Reuse /
3.SOReuse): 1
[*] Please input the [ip:]<port> : 10001
[*] Waiting for response......
[*] Node is listening on 10001
```

注意，`listen`是比较特殊的一个命令，可以看到，`listen`命令有三种模式

1. `Normal passive`: 此选项意味着agent将会以普通的方式监听在目标端口，并等待子节点连入

2. `IPTables Reuse`：此选项意味着agent将会以IPTables Reuse的方式复用端口，并等待子节点连入
3. `SOReuse`：此选项意味着agent将会以SOReuse的方式复用端口，并等待子节点连入

第一个模式是最普遍使用的，若父节点以这种方式监听，那么子节点仅需要`-c 父节点ip:port`就可以加入网络

第二个和第三个模式是比较特殊的，若用户选择第二或第三个模式，那么用户必须保证当前操作的节点本身就是以端口复用的方式启动的，否则将无法使用这两个模式

第二和第三个模式将不需要用户输入任何信息，节点将会自动使用其自身启动时的参数来复用端口，并准备接受子节点的连接

另外，`listen`一次只能接受一个子节点的连入，若需要多个子节点连入，请执行相应次数的`listen`命令

`addmemo`: 为当前节点添加备忘

```
(node 0) >> addmemo test
[*] Memo added!
(node 0) >> exit
(admin) >> detail
Node[0] -> IP: 127.0.0.1:10000  Hostname: ph4ntoms-MBP.lan  User: ph4ntom
Memo:  test
```

`delmemo`: 删除当前节点的备忘

```
(node 0) >> delmemo
[*] Memo deleted!
(node 0) >> exit
(admin) >> detail
Node[0] -> IP: 127.0.0.1:10000  Hostname: ph4ntoms-MBP.lan  User: ph4ntom
Memo:
```

`ssh`: 命令节点以ssh方式连接目标机器

```
(node 0) >> ssh 127.0.0.1:22
[*] Please choose the auth method(1.username&&password / 2.certificate):
1
[*] Please enter the username: ph4ntom
[*] Please enter the password: *****
[*] Waiting for response.....
[*] Connect to target host via ssh successfully!
 # ph4ntom @ ph4ntoms-MBP in ~ 👑 [17:03:56]
$ whoami
ph4ntom
 # ph4ntom @ ph4ntoms-MBP in ~ 👑 [17:04:16]
$
```

在此模式下，tab键将被禁止

`shell`: 获取当前节点的shell

```
(node 0) >> shell
[*] Waiting for response.....
[*] Shell is started successfully!

bash: no job control in this shell

The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
bash-3.2$ whoami
ph4ntom
bash-3.2$
```

在此模式下，tab键将被禁止

`socks`：在当前节点上启动socks5服务

```
(node 0) >> socks 7777
[*] Trying to listen on 0.0.0.0:7777......
[*] Waiting for response......
[*] Socks start successfully!
(node 0) >>
```

注意一点，此处的7777端口不是在agent上开启的，而是在admin上开启

另外，若需要设置用户名密码，可将上方命令改为`socks 7777 <your username> <your password>`

`stopsocks`: 停止在当前节点上的socks5服务

```
(node 0) >> stopsocks
Socks Info ---> ListenAddr: 0.0.0.0:7777    Username: <null>    Password:
<null>
[*] Do you really want to shutdown socks?(yes/no): yes
[*] Closing......
[*] Socks service has been closed successfully!
(node 0) >>
```

`connect`: 命令当前节点连接至另一个子节点

```
agent-1: ./stowaway_agent -l 10002
```

```
(node 0) >> connect 127.0.0.1:10002
[*] Waiting for response......
[*] New node come! Node id is 1

(node 0) >>
```

`sshtunnel`: 命令当前节点以ssh隧道的方式连接至另一个子节点

```
agent-2: ./stowaway_agent -l 10003
```

```
(node 0) >> sshtunnel 127.0.0.1:22 10003
[*] Please choose the auth method(1.username&&password / 2.certificate):
1
[*] Please enter the username: ph4ntom
[*] Please enter the password: ******
[*] Waiting for response.....
[*] New node come! Node id is 2

(node 0) >>
```

在严格受限的网络环境下，可以利用ssh隧道的方式来将stowaway的流量伪装为ssh流量，从而避开防火墙的限制

`upload`: 向当前节点上传文件

```
(node 0) >> upload test.7z test.xxx
[*] File transmitting, please wait...
136.07 KiB / 136.07 KiB [-----------------------------------------------
----------------------------------] 100.00% ? p/s 0s
```

`download`: 下载当前节点上的文件

```
(node 0) >> download test.xxx test.xxxx
[*] File transmitting, please wait...
136.07 KiB / 136.07 KiB [-----------------------------------------------
----------------------------------] 100.00% ? p/s 0s
```

`forward`: 映射admin上的端口至远程端口

```
(node 0) >> forward 9000 127.0.0.1:22
[*] Trying to listen on 0.0.0.0:9000......
[*] Waiting for response......
[*] Forward start successfully!
(node 0) >>
```

```
$ ssh 127.0.0.1 -p 9000
Password:
 # ph4ntom @ ph4ntoms-MBP in ~ 👑 [17:19:51]
$
```

`stopforward`: 关闭当前节点的远程映射

```
(node 0) >> stopforward
[0] All
[1] Listening Addr : [::]:9000 , Remote Addr : 127.0.0.1:22 , Current
Active Connnections : 1
[*] Do you really want to shutdown forward?(yes/no): yes
[*] Please choose one to close: 1
[*] Closing......
[*] Forward service has been closed successfully!
```

`backward`: 反向映射当前agent上的端口至admin的本地端口

```
(node 0) >> backward 9001 22
[*] Trying to ask node to listen on 0.0.0.0:9001......
[*] Waiting for response......
[*] Backward start successfully!
(node 0) >>
```

```
$ ssh 127.0.0.1 -p 9001
Password:
 # ph4ntom @ ph4ntoms-MBP in ~ 🌈 [17:22:14]
$
```

`stopbackward`: 关闭当前节点的反向映射

```
(node 0) >> stopbackward
[0] All
[1] Remote Port : 9001 , Local Port : 22 , Current Active Connnections :
1
[*] Do you really want to shutdown backward?(yes/no): yes
[*] Please choose one to close: 1
[*] Closing......
[*] Backward service has been closed successfully!
```

`shutdown`: 命令当前节点下线

```
(node 1) >> shutdown
(node 1) >>
[*] Node 1 is offline!
```

`back`: 退回到主panel

```
(node 1) >> back
(admin) >>
```

`exit`: 退出stowaway

```
(node 1) >> exit
[*] Do you really want to exit stowaway?(y/n): y
[*] BYE!
```

TODO

- 修复bug
- 支持TLS
- 支持多startnode的形式

注意事项

- 此程序仅是闲暇时开发学习，结构及代码结构不够严谨，功能可能存在bug，请多多谅解
- admin不在线时，新节点将不允许加入
- admin仅支持一个直接连接的agent节点，agent节点则无此限制
- 如果用户使用windows下的admin端，请先下载ansicon，或者在这里下载，之后进入对应系统位数的文件夹，执行`ansicon.exe -i`即可，不然admin端会出现乱码的问题
- 本程序仅支持标准的基于RFC1928所阐述的`UDP ASSOCIATE`，请在使用socks5 udp代理时注意您所使用的程序(例如扫描器等)，包构造方式必须遵守标准的RFC1928，并且需要自行处理丢包状况。

404星链计划

Stowaway 现已加入 404星链计划

致谢

感谢所有为此项目贡献代码以及建议的师傅们~

- lz520520
- SignorMercurio
- MM0x00
- r0ck3rt

参考项目

- rootkiter#Termite
- Venom