

## Угадай хеш

### The condition of a given task:

#### Описание задания

Вам необходимо по внешнему виду хеша определить его тип.

#### Подсказка к решению

Не пользуйтесь сторонними программами и сервисами, они вас лишь замедлят.

### Solution:

just use <https://suip.biz/ru/?act=hashtag> or [hashid](#) to detect type of the hash forgot valid flag but one of the actual is

Answer is:

**719cd9f619da1b4362c29f0eb2227295b5e197bdfd7c749f0ff2c35ff20e66dd022b4f9ff33e31bdfcf18bd687032a1e**

---

## IPTables

### The condition of a given task:

Вам дан дамп правил межсетевого экрана Iptables. Правила добавлялись в хаотичном порядке и перекрывали адреса подсетей откуда шли атаки. Кажется, что сервер теперь успешно изолирован от всего интернета. Или нет? Найдите адреса, которые остались незаблокированными из подсети 112.0.0.0/4.

### Solution:

Let's check what inside your [iptables-dump.txt](#) filter it by regex `\b(?:[0-9]{1,3}\.){3}[0-9]{1,3}/\d{1,2}\b`

At first sort subnets in ascending order of bit length in the mask and check that *you have full path from /4 to /32(single adress)* keep in mind that **every step you're halving your not whitelisted subnet**. 112.0.0.0/4 means 112.0.0.0-127.255.255.255 112.0.0.0/5 means 112.0.0.0-119.255.255.255 so now the *remaining non-blocked addresses* is **120.0.0.0-127.255.255.255** 124.0.0.0/6 means 124.0.0.0-127.255.255.255 120.0.0.0/7 means 120.0.0.0-121.255.255.255 so now the *remaining non-blocked addresses* is **122.0.0.0-123.255.255.255** step by step it's a verbal exercise

```
full solution here below on the right side alive addresses ranges
Main subnet is 112-127.0.0.0 16
112.0.0.0/5 120-127
124.0.0.0/6 120-124
120.0.0.0/7 122-124
122.0.0.0/8 123-124
```

```
first bite is 123
```

```
123.128.0.0/9 123.0-128
123.64.0.0/10 123.0-64
123.32.0.0/11 123.0-32
123.0.0.0/12 123.16-32
123.24.0.0/13 123.16-24 123.20.0.0/14 123.16-20 123.18.0.0/15 123.16-18 123.17.0.0/16 123.16
```

second bite is 16

```
123.16.128.0/17 123.16.0-128 123.16.64.0/18 123.16.0-64 123.16.32.0/19 123.16.0-32 123.16.16.0/20
123.16.0-16 123.16.0.0/21 123.16.8-16 123.16.12.0/22 123.16.8-12 123.16.10.0/23 123.16.8-10
123.16.9.0/24 123.16.8
```

third bite is 8

```
123.16.8.128/25 123.16.8.0-128 123.16.8.64/26 123.16.8.0-64 123.16.8.32/27 123.16.8.0-32
123.16.8.16/28 123.16.8.0-16 123.16.8.0/29 123.16.8.8-16 123.16.8.12/30 123.16.8.8-12 123.16.8.10/31
123.16.8.8-10 123.16.8.9/32 123.16.8.8
```

last bite is 8

Answer is: **123.16.8.8**

## DNS Hidden Record

### The condition of a given task:

⚠ Подключитесь к VPN сети.

Вам доступен DNS сервер. Попробуйте получить как можно больше информации о домене `hiddentext.osint`. Ответом к данному заданию является строка, которую можно извлечь (вида

`98355ba9fcc21782ec6f16975413bb011fe7effb87bce04f551f60944352aa568d25f8782187c89787a9f27c`  
`ff88`).

Подсказка

Например, вам был сгенерирован следующий IP: `10.10.10.10:49666`. Это значит, что DNS сервер располагается на сервере с адресом `10.10.10.10` и слушает на порту `49666`.

### Solution:

turn on your vpn and try to `dig @10.10.10.10 -port <your_port> hiddentext.osint txt` Obviously, the text is stored in a txt record (usually it is), so let's see what txt records this domain has. and he has a txt record with our flag.

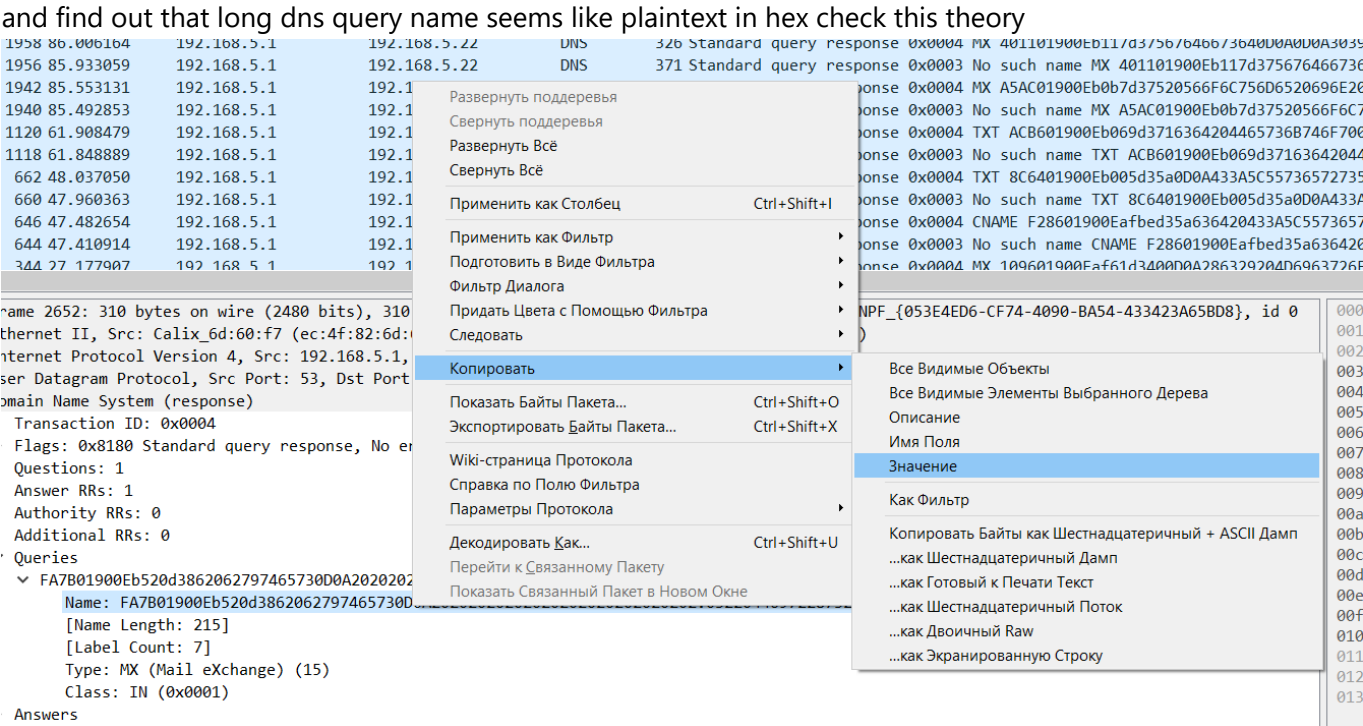
Answer is:

**4f9ccb5104ad6b34a10576732e85e9ba5b1b650abcd1500d4908fce7a84e8806f06c341e3f46d78d38a2265f0f0eb785**

## DNS туннель

Вам дана запись сетевого трафика (PCAP файл). Известно, что передача информации происходила внутри DNS туннеля. Извлеките флаг.

At first open `dnstunnel.pcapng` in wireshark and try filter like `dns.qry.name.len > 100 && !mdns` for detect `dns tunnels` now you can see smth like



and try ([www.rapidtables.com](https://www.rapidtables.com/convert/number/hex-to-ascii.html))[<https://www.rapidtables.com/convert/number/hex-to-ascii.html>] *show the result!!*

Paste hex numbers or drop file

```
7BC401900Eb587d394666C61677B746869735F69735F615F68696464656E5F
6.D6573736167655F696E5F646E735F72657175657374737D0D0A433A5C557
365.72735C67646673645C446F63756D656E74733E.t.freesever.site
```

Character encoding

ASCII



Convert



Reset



Swap

```
{ÄµÓflag{this_is_a_hidden_message_in_dns_requests}
C:\Users\gdfsd\Documents>pîî
```

(bruteforce any of this dns query names to find out flag)

Answer is: ***flag{this\_is\_a\_hidden\_message\_in\_dns\_requests}***

## DNS subdomains

### The condition of a given task:

Вам доступен DNS сервер. Сколько поддоменов содержится в зоне hosting.osint?

Поддомены – короткие (до 6 символов) распространенные английские слова строчными буквами без специальных символов.

Подсказка

Например, вам был сгенерирован следующий IP: 10.10.10.10:49666. Это значит, что DNS сервер располагается на сервере с адресом 10.10.10.10 и слушает на порту 49666.

Обратите внимание, на каком транспортном протоколе по умолчанию работает DNS?

### Solution:

turn on your vpn and try to `gobuster dns -d hosting.osint -w <path/to/your/wordlist> -r 10.10.10.10:<your_port> -q -o output` it's **biggest dogshit in tasks** that i solved... `./massdns -r <file with 10.10.10.10:<your_port>> -t A -o S -w <outputfile> <wordlist>` Answer is: **688**

## DNS very special domain

### The condition of a given task:

⚠ Подключитесь к VPN сети.

Вам доступен DNS сервер. Какой IP адрес имеет домен, IP которого отличается от всех остальных (в зоне hosting.osint)? Искомые поддомены – короткие (до 6 символов) распространенные английские слова строчными буквами без специальных символов.

Подсказка

Например, вам был сгенерирован следующий IP: 10.10.10.10:49666. Это значит, что DNS сервер располагается на сервере с адресом 10.10.10.10 и слушает на порту 49666.

### Solution:

with your domain list you can *bruteforce* answer via *this command*: `cat <path/to/domains/list> | while read domain; do dig @10.10.10.10 -p <your_port> $domain A +noall +answer | grep -v 1.2.3.4; done` `grep -v 1.2.3.4` cause all except one domains have ip 1.2.3.4 in output we can see that domain *talk.hosting.osint* has ipv4 172.168.9.19

Answer is: **172.168.9.19**

---

## Прослушка

### The condition of a given task:

Вам дан доступ к виртуальной машине по SSH (логин: user, пароль: user). Известно, что в сети с данной машиной расположена ещё одна, которая периодически пересылает секретную информацию по одному из распространенных протоколов.

### Solution:

at first do `arp -a` and see that we have only *one* host in our network this host have ip **172.19.0.2** and name **trafik.task** we can scan ports with *nc* via `for port in {1..65535} do nc -zv 172.19.0.2 $port done` see that 172.19.0.2 has two open ports: **80** and **8080** with `wget 172.19.0.2:8080` we can see index.html of this web page ***no way dead end road***

**better try use `tcpdump` cause you can detect some packets** in log we see a requests like 11:46:39.405189 IP (tos 0x0, ttl 64, id 16566, offset 0, flags [DF], proto UDP (17), length 144) 884c13fa-449d-43a7-bdbf-fcebab8b7084-checker-sender-1.884c13fa-449d-43a7-bdbf-fcebab8b7084\_local.52582 > 255.255.255.255.5005: [bad udp cksum 0xc138 -> 0x3bb5!] UDP, length 116 and **tryna check payloads via opening dump file** that we can create and print via `tcpdump -w <filename>; cat <filename>` inside of it we see secret flag: 80453fa8dce79cd29ee5a79c321a744059dad0d78fceb7bff68de8f505f339b9aca66ad8beede13b0c99965f

b98424c7

```

ec691e7fb32a:~# tcpdump -vvv
tcpdump: listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
11:46:19.384360 IP (tos 0x0, ttl 64, id 13437, offset 0, flags [DF], proto UDP (17), length 144)
    884c13fa-449d-43a7-bdbf-fcebab8b7084-checker-sender-1.884c13fa-449d-43a7-bdbf-fcebab8b7084_local.41082 > 255.255.255.255.5005: [bad udp cksum 0xc138 -> 0x68a1!] UDP, length 116
11:46:29.394769 IP (tos 0x0, ttl 64, id 14550, offset 0, flags [DF], proto UDP (17), length 144)
    884c13fa-449d-43a7-bdbf-fcebab8b7084-checker-sender-1.884c13fa-449d-43a7-bdbf-fcebab8b7084_local.49246 > 255.255.255.255.5005: [bad udp cksum 0xc138 -> 0x48bd!] UDP, length 116
11:46:39.405189 IP (tos 0x0, ttl 64, id 16566, offset 0, flags [DF], proto UDP (17), length 144)
    884c13fa-449d-43a7-bdbf-fcebab8b7084-checker-sender-1.884c13fa-449d-43a7-bdbf-fcebab8b7084_local.52582 > 255.255.255.255.5005: [bad udp cksum 0xc138 -> 0x3bb5!] UDP, length 116
11:46:49.415558 IP (tos 0x0, ttl 64, id 17908, offset 0, flags [DF], proto UDP (17), length 144)
    884c13fa-449d-43a7-bdbf-fcebab8b7084-checker-sender-1.884c13fa-449d-43a7-bdbf-fcebab8b7084_local.43592 > 255.255.255.255.5005: [bad udp cksum 0xc138 -> 0x5ed3!] UDP, length 116
^C
4 packets captured
4 packets received by filter
0 packets dropped by kernel
ec691e7fb32a:~# tcpdump -vvv -w output
tcpdump: listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
^C2 packets captured
2 packets received by filter
0 packets dropped by kernel
ec691e7fb32a:~# cat output
0
eH@b|8Hello, secret flag: 80453fa8dce79cd29ee5a79c321a744059dad0d78fceb7bfff68de8f505f339b9aca66ad8beede13b0c99965fb98424c7
eL@b|8Hello, secret flag: 80453fa8dce79cd29ee5a79c321a744059dad0d78fceb7bfff68de8f505f339b9aca66ad8beede13b0c99965fb98424c7ec691e7fb32a:~#

```

Answer is:

**80453fa8dce79cd29ee5a79c321a744059dad0d78fceb7bfff68de8f505f339b9aca66ad8beede13b0c99965fb98424c7**

## Raw ICMP

### #### The condition of a given task:

Известно, что ваш сосед по сети любит общаться через сообщения внутри ICMP протокола. Отправьте ему ICMP сообщение с текстом `give me flag` и будьте уверены, что он в ответ пришлет вам ICMP сообщение с флагом. Внимание, флаг обычно является строкой из примерно 92 латинских символов.

Подсказка: сервер отправляет сообщение с помощью `ping -p`

### Solution:

`apt install net-tools; arp -a` to discover hosts in our network at first let's find host that responds to the request `ping -I <network_interface> -s 12 -p 67697665206D6520666C6167 -b -c 1 255.255.255.255` with something other than the usual echo reply like a path of flag (cause 16 bytes - limit) and we need `tcpdump` to check the payloads from **13b7c5ac-45a9-4531-bae7-59584e0202bc-internal-1.13b7c5ac-45a9-4531-bae7-59584e0202bc\_local:(victim)** that answers smth like

```

13:35:39.379513 02:42:ac:1b:00:03 (oui Unknown) > 02:42:ac:1b:00:02 (oui Unknown), ethertype IPv4 (0x0800), length 98: (tos 0x0, ttl 64, id 43459, offset 0, flags [none], proto ICMP (1), length 84)
f4b7e46b397a > 13b7c5ac-45a9-4531-bae7-59584e0202bc-internal-1.13b7c5ac-45a9-4531-bae7-59584e0202bc_local: ICMP echo reply, id 24522, seq 1, length 64
0x0000: 4500 0054 a9c3 0000 4001 78aa ac1b 0003  E..T....@.x....
0x0010: ac1b 0002 0000 6e87 5fca 0001 ab75 1965  ....n.....u.e
0x0020: 0000 0000 38ca 0500 0000 0000 6537 6566  ....8.....e7ef
0x0030: 3265 6136 3762 6535 3730 3666 6537 6566  2ea67be5706fe7ef
0x0040: 3265 6136 3762 6535 3730 3666 6537 6566  2ea67be5706fe7ef
0x0050: 3265 6136                               2ea6

```

keep in mind that maximum of bytes in payload is 16, and after first 16 bytes server repeats payload. the flag has length like 92 bytes and we can take the first 16 bytes from every reply response and concatenate it to get the flag

```
13:35:39.363870 02:42:ac:1b:00:02 (oui Unknown) > 02:42:ac:1b:00:03 (oui Unknown), ethertype IPv4 (0x0800), length 98: (tos 0x0, ttl 64, id 20726, offset 0, flags [DF], proto ICMP (1), length 84)
13b7c5ac-45a9-4531-bae7-59584e0202bc-internal-1.13b7c5ac-45a9-4531-bae7-59584e0202bc_local > f4b7e46b397a: ICMP echo request, id 24518, seq 1, length 64
0x0000: 4500 0054 50f6 4000 4001 9177 ac1b 0002 E..TP.@.@.w....
0x0010: ac1b 0003 0800 c5cf 5fc6 0001 ab75 1965 .....u.e
0x0020: 0000 0000 2f8d 0500 0000 0000 3765 3235 ....7e25
0x0030: 3531 3765 6361 3264 6239 3338 3765 3235 517eca2db9387e25
0x0040: 3531 3765 6361 3264 6239 3338 3765 3235 517eca2db9387e25
0x0050: 3531 3765 517e

13:35:39.363905 02:42:ac:1b:00:02 (oui Unknown) > 02:42:ac:1b:00:03 (oui Unknown), ethertype IPv4 (0x0800), length 98: (tos 0x0, ttl 64, id 43455, offset 0, flags [none], proto ICMP (1), length 84)
f4b7e46b397a > 13b7c5ac-45a9-4531-bae7-59584e0202bc-internal-1.13b7c5ac-45a9-4531-bae7-59584e0202bc_local: ICMP echo reply, id 24518, seq 1, length 64
0x0000: 4500 0054 a9bf 0000 4001 78ae ac1b 0003 E..T...@.x....
0x0010: ac1b 0002 0000 cdcf 5fc6 0001 ab75 1965 .....u.e
0x0020: 0000 0000 2f8d 0500 0000 0000 3765 3235 ....7e25
0x0030: 3531 3765 6361 3264 6239 3338 3765 3235 517eca2db9387e25
0x0040: 3531 3765 6361 3264 6239 3338 3765 3235 517eca2db9387e25
0x0050: 3531 3765 517e

13:35:39.367891 02:42:ac:1b:00:02 (oui Unknown) > 02:42:ac:1b:00:03 (oui Unknown), ethertype IPv4 (0x0800), length 98: (tos 0x0, ttl 64, id 20727, offset 0, flags [DF], proto ICMP (1), length 84)
13b7c5ac-45a9-4531-bae7-59584e0202bc-internal-1.13b7c5ac-45a9-4531-bae7-59584e0202bc_local > f4b7e46b397a: ICMP echo request, id 24519, seq 1, length 64
0x0000: 4500 0054 50f7 4000 4001 9176 ac1b 0002 E..TP.@.@.v....
0x0010: ac1b 0003 0800 2fff 5fc7 0001 ab75 1965 .....u.e
0x0020: 0000 0000 f89c 0500 0000 0000 3862 3036 .....8b06
0x0030: 6433 3235 3733 3266 3262 3863 3862 3036 d325732f2b8c8b06
0x0040: 6433 3235 3733 3266 3262 3863 3862 3036 d325732f2b8c8b06
0x0050: 6433 3235 d325

13:35:39.367928 02:42:ac:1b:00:02 (oui Unknown) > 02:42:ac:1b:00:03 (oui Unknown), ethertype IPv4 (0x0800), length 98: (tos 0x0, ttl 64, id 43456, offset 0, flags [none], proto ICMP (1), length 84)
f4b7e46b397a > 13b7c5ac-45a9-4531-bae7-59584e0202bc-internal-1.13b7c5ac-45a9-4531-bae7-59584e0202bc_local: ICMP echo reply, id 24519, seq 1, length 64
0x0000: 4500 0054 a9c0 0000 4001 78ad ac1b 0003 E..T...@.x....
0x0010: ac1b 0002 0000 37ff 5fc7 0001 ab75 1965 .....7...u.e
0x0020: 0000 0000 f89c 0500 0000 0000 3862 3036 .....8b06
0x0030: 6433 3235 3733 3266 3262 3863 3862 3036 d325732f2b8c8b06
0x0040: 6433 3235 3733 3266 3262 3863 3862 3036 d325732f2b8c8b06
0x0050: 6433 3235 d325

13:35:39.371662 02:42:ac:1b:00:02 (oui Unknown) > 02:42:ac:1b:00:03 (oui Unknown), ethertype IPv4 (0x0800), length 98: (tos 0x0, ttl 64, id 20728, offset 0, flags [DF], proto ICMP (1), length 84)
13b7c5ac-45a9-4531-bae7-59584e0202bc-internal-1.13b7c5ac-45a9-4531-bae7-59584e0202bc_local > f4b7e46b397a: ICMP echo request, id 24520, seq 1, length 64
0x0000: 4500 0054 50f8 4000 4001 9175 ac1b 0002 E..TP.@.@.u....
0x0010: ac1b 0003 0800 57fa 5fc8 0001 ab75 1965 .....w.....u.e
0x0020: 0000 0000 b0ab 0500 0000 0000 3932 3738 .....9278
0x0030: 3235 3062 3330 6161 3064 6162 3932 3738 250b30aa0dab9278
0x0040: 3235 3062 3330 6161 3064 6162 3932 3738 250b30aa0dab9278
0x0050: 3235 3062 250b
```

Press ↵ to Reconnect

but be **careful with duplicates**

Answer is:

***7e25517eca2db9388b06d325732f2b8c9278250b30aa0dab8e61c66b01e039d3e7ef2ea67be5706f5058fe9e9e3607a3f***