BUGTREE

Born to kill bugs. Live by them.

Nmap Anonymization with Proxychains

OCTOBER 17TH, 2017

If you work in the infosec industry and never asked yourself this question, please take your time to think. Our objective here is to accomplish a full scan without revealing our real IP to the target.

Anonymizing an nmap scan

There are multiple ways to accomplish this noble mission. One way would be to execute an Idle Scan, aka "the ultimate stealth scan".

Another idea is just to use proxies or even TOR. Although nmap offers a `--proxy` option, it should be noted that:

```
Warning: this feature is still under development and has limitati
ons. It is implemented within the nsock library and thus has no e
ffect on the ping, port scanning and OS discovery phases of a sca
n. Only NSE and version scan benefit from this option so far—othe
r features may disclose your true address. SSL connections are no
t yet supported, nor is proxy-side DNS resolution (hostnames are
 always resolved by Nmap).
```

Well, it does not give us much of an option instead of not trusting this feature, does it? Then we present you another solution: proxychains.

BUGTREE

Born to kill bugs. Live by them.

What does proxychains do (and does not)?

According to the package description

```
proxy chains - redirect connections through proxy servers

Proxy chains force any tcp connection made by any given tcp clien
t to follow through proxy (or proxy chain). It is a kind of proxi
fier. It acts like sockscap / premeo / eborder driver ( intercept
s TCP calls )

This version supports SOCKS4, SOCKS5 and HTTP CONNECT proxy serve
rs. Different proxy types can be mixed in the same chain.
```

Another thing to note in here:

```
[Proxychains] Allows TCP and DNS tunneling through proxies.
```

These are the two protocols we are going to “secure” while using proxychains. Keep in mind that all other protocols might pose a hazard to your anonimity.

Please do check your proxychains version before using it

The latest version of this tool according to the SourceForge page:

```
ProxyChains README
current version: 3.1
======================
```

However, there is a very well starred repo in GitHub that declares:

```
ProxyChains ver. 4.2.0 README
```

BUGTREE

Born to kill bugs. Live by them.

Now, according to proxychain's man page in Debian 9 (and Kali therefore):

```
This version (2.0) supports SOCKS4, SOCKS5 and HTTP CONNECT proxy
 servers.  Auth-types: socks - "user/pass" , http - "basic".
```

WTF Debian?! To be sure this is just an outdated manpage I checked the package informations:

```
$ apt show proxychains
Package: proxychains
Version: 3.1-7
```

So yeah, it seems ok. The main difference between versions 2 and 3 is the support for DNS requests (which is something really important). Anyway, you should check your version and run some local tests just in case.

nmap scan phases

I am assuming here you are familiarized with it. If not, refer to this page. All an all, the phases are:

1. Script pre-scanning
2. Target enumeration
3. Host discovery (ping scanning)
4. Reverse-DNS resolution
5. Port scanning
6. Version detection
7. OS detection
8. Traceroute
9. Script scanning
10. Output
11. Script post-scanning

BUGTREE

Born to kill bugs. Live by them.

What is a ping to nmap?

First point to make clear is the concept of *ping* in nmap’s terminology:

So let it be clear once and for all:

```
Ping in daily network lingo is a *program* that uses ICMP to "sen
d ICMP ECHO_REQUEST to network hosts".
Ping in nmap lingo is any request done in the host discovery phas
e (**not necessarily ICMP**).
```

Ping scan therefore uses multiple protocols besides ICMP. Keep that in mind. Now we know the difference between these two guys, we may move on.

Proxychains

Our proxychains config is set to use SOCKS5 protocol in order to tunnel our requests through the TOR network. Our TOR client is listening in port 9050 for that matter. This is the default configuration for proxychains, so lets keep it this way.

Leaks in Host Discovery

Since we can only use TCP and DNS within Proxychains, we must disable all other protocols used in Host Discovery phase. According to the Nmap Network Scanning Guide:

```
Note that if you specify any of the -P options discussed in this
 section, they *replace* the default discovery probes rather than
 adding to them.
```

So if we want to skip any UDP or ICMP packets in host discovery and stick only to TCP probes, it should be enough to add the -PS option. This should orverrule all the other ping types.

BUGTREE

Born to kill bugs. Live by them.

```
#proxychains nmap -n -PS -sT 216.58.222.14 -p80 --packet-trace -v
v
ProxyChains-3.1 (http://proxychains.sf.net)

Starting Nmap 7.40 ( https://nmap.org ) at 2017-10-17 20:12 -02
Initiating Ping Scan at 20:12
Scanning 216.58.222.14 [1 port]
SENT (0.0689s) TCP 192.168.0.11:48630 > 216.58.222.14:80 S ttl=52
 id=6588 iplen=44  seq=630702871 win=1024 <mss 1460>
RCVD (0.2618s) TCP 216.58.222.14:80 > 192.168.0.11:48630 SA ttl=4
6 id=10400 iplen=44  seq=1872872446 win=42780 <mss 1380>
Completed Ping Scan at 20:12, 0.43s elapsed (1 total hosts)
Initiating Connect Scan at 20:12
Scanning 216.58.222.14 [1 port]
|S-chain|-<>-127.0.0.1:9050-<><>-216.58.222.14:80-<><>-OK
CONN (1.0838s) TCP localhost > 216.58.222.14:80 => Connected
Discovered open port 80/tcp on 216.58.222.14
Completed Connect Scan at 20:12, 0.59s elapsed (1 total ports)
Nmap scan report for 216.58.222.14
Host is up, received syn-ack ttl 46 (0.24s latency).
Scanned at 2017-10-17 20:12:47 -02 for 1s
PORT   STATE SERVICE REASON
80/tcp open  http    syn-ack

Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 1.08 seconds
           Raw packets sent: 1 (44B) | Rcvd: 1 (44B)
```

But no luck this time! In fact, we can check whether our host is probing the target directly:

```
# tcpdump -vvv host 216.58.222.14
tcpdump: listening on wlp2s0, link-type EN10MB (Ethernet), captur
e size 262144 bytes
20:12:47.846175 IP (tos 0x0, ttl 52, id 6588, offset 0, flags [no
ne], proto TCP (6), length 44)
```

BUGTREE

Born to kill bugs. Live by them.

```
    fahrzeug.48630 > gru06s25-in-f14.1e100.net.http: Flags [S], c
ksum 0x7634 (correct), seq 630702871, win 1024, options [mss 146
0], length 0
20:12:48.039067 IP (tos 0x0, ttl 46, id 10400, offset 0, flags [n
one], proto TCP (6), length 44)
    gru06s25-in-f14.1e100.net.http > fahrzeug.48630: Flags [S.],
 cksum 0x9fb6 (correct), seq 1872872446, ack 630702872, win 4278
0, options [mss 1380], length 0
20:12:48.039121 IP (tos 0x0, ttl 64, id 61273, offset 0, flags [D
F], proto TCP (6), length 40)
    fahrzeug.48630 > gru06s25-in-f14.1e100.net.http: Flags [R], c
ksum 0x91ed (correct), seq 630702872, win 0, length 0

3 packets captured
3 packets received by filter
0 packets dropped by kernel
```

Two SYN pings from our host to the target, no proxy. Bad bad OPSEC, folks.

Since proxychains is not tunneling our ping scan, even when we use SYN scan only, how can we avoid leaking our IP during the host discovery phase? Simple: skip it! Make sure the target host is up using some other method and then go for the scanning phase.

```
# proxychains nmap -n -Pn -sT 216.58.222.14 -p80 --packet-trace -
vv
ProxyChains-3.1 (http://proxychains.sf.net)

Starting Nmap 7.40 ( https://nmap.org ) at 2017-10-17 20:13 -02
Initiating Connect Scan at 20:13
Scanning 216.58.222.14 [1 port]
|S-chain|-<>-127.0.0.1:9050-<><>-216.58.222.14:80-<><>-OK
CONN (0.7309s) TCP localhost > 216.58.222.14:80 => Connected
Discovered open port 80/tcp on 216.58.222.14
Completed Connect Scan at 20:13, 0.70s elapsed (1 total ports)
Nmap scan report for 216.58.222.14
Host is up, received user-set (0.70s latency).
```

BUGTREE

Born to kill bugs. Live by them.

```
Scanned at 2017-10-17 20:13:45 -02 for 1s
PORT   STATE SERVICE REASON
80/tcp open  http    syn-ack

Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 0.73 seconds
```

Now our tcpdump result is clean:

```
# tcpdump -vvv host 216.58.222.14
tcpdump: listening on wlp2s0, link-type EN10MB (Ethernet), captur
e size 262144 bytes

0 packets captured
0 packets received by filter
0 packets dropped by kernel
```

In conclusion, it is not safe to go through the host discovery phase when using proxychains.

Leaks in DNS resolution

This time we are concerned with leaking our IP address through DNS requests. Let's begin with a simple TCP scan:

```
# proxychains nmap -Pn -sT -p80 www.google.com -vv
ProxyChains-3.1 (http://proxychains.sf.net)

Starting Nmap 7.40 ( https://nmap.org ) at 2017-10-17 12:36 -02
|DNS-request| www.google.com
|S-chain|-<>-127.0.0.1:9050-<><>-4.2.2.2:53-<><>-OK
|DNS-response| www.google.com is 74.125.29.104
74.125.29.104/0 looks like an IPv6 target specification -- you ha
ve to use the -6 option.
Read data files from: /usr/bin/../share/nmap
```

BUGTREE

Born to kill bugs. Live by them.

```
WARNING: No targets were specified, so 0 hosts scanned.
Nmap done: 0 IP addresses (0 hosts up) scanned in 1.18 seconds
```

The DNS request was correctly made through proxychains, which is using TOR. Strangely, nmap interprets 216.58.214.228/0 as an IPv6 address and then breaks. It seems nmap does not get along with our SOCKS5 DNS resolution.

Let's see what happens when we scan the IP address directly, instead of the name:

```
# proxychains nmap -Pn -sT -p80 216.58.222.14 --packet-trace -vv
ProxyChains-3.1 (http://proxychains.sf.net)

Starting Nmap 7.40 ( https://nmap.org ) at 2017-10-17 12:52 -02
NSOCK INFO [0.0310s] nsock_iod_new2(): nsock_iod_new (IOD #1)
NSOCK INFO [0.0310s] nsock_connect_udp(): UDP connection requeste
d to 8.8.8.8:53 (IOD #1) EID 8
NSOCK INFO [0.0310s] nsock_read(): Read request from IOD #1 [8.8.
8.8:53] (timeout: -1ms) EID 18
NSOCK INFO [0.0310s] nsock_iod_new2(): nsock_iod_new (IOD #2)
NSOCK INFO [0.0310s] nsock_connect_udp(): UDP connection requeste
d to 8.8.4.4:53 (IOD #2) EID 24
NSOCK INFO [0.0310s] nsock_read(): Read request from IOD #2 [8.8.
4.4:53] (timeout: -1ms) EID 34
NSOCK INFO [0.0310s] nsock_iod_new2(): nsock_iod_new (IOD #3)
NSOCK INFO [0.0310s] nsock_connect_udp(): UDP connection requeste
d to 172.16.100.1:53 (IOD #3) EID 40
NSOCK INFO [0.0310s] nsock_read(): Read request from IOD #3 [172.
16.100.1:53] (timeout: -1ms) EID 50
Initiating Parallel DNS resolution of 1 host. at 12:52
NSOCK INFO [0.0310s] nsock_write(): Write request for 44 bytes to
 IOD #1 EID 59 [8.8.8.8:53]
NSOCK INFO [0.0310s] nsock_trace_handler_callback(): Callback: CO
NNECT SUCCESS for EID 8 [8.8.8.8:53]
NSOCK INFO [0.0310s] nsock_trace_handler_callback(): Callback: WR
ITE SUCCESS for EID 59 [8.8.8.8:53]
NSOCK INFO [0.0310s] nsock_trace_handler_callback(): Callback: CO
NNECT SUCCESS for EID 24 [8.8.4.4:53]
```

BUGTREE

Born to kill bugs. Live by them.

```
NSOCK INFO [0.0310s] nsock_trace_handler_callback(): Callback: CO
NNECT SUCCESS for EID 40 [172.16.100.1:53]
NSOCK INFO [0.2920s] nsock_trace_handler_callback(): Callback: RE
AD SUCCESS for EID 18 [8.8.8.8:53] (125 bytes)
NSOCK INFO [0.2920s] nsock_read(): Read request from IOD #1 [8.8.
8.8:53] (timeout: -1ms) EID 66
NSOCK INFO [0.2920s] nsock_iod_delete(): nsock_iod_delete (IOD #
1)
NSOCK INFO [0.2920s] nevent_delete(): nevent_delete on event #66
 (type READ)
NSOCK INFO [0.2920s] nsock_iod_delete(): nsock_iod_delete (IOD #
2)
NSOCK INFO [0.2920s] nevent_delete(): nevent_delete on event #34
 (type READ)
NSOCK INFO [0.2920s] nsock_iod_delete(): nsock_iod_delete (IOD #
3)
NSOCK INFO [0.2920s] nevent_delete(): nevent_delete on event #50
 (type READ)
Completed Parallel DNS resolution of 1 host. at 12:52, 0.26s elap
sed
Initiating Connect Scan at 12:52
Scanning gru06s25-in-f14.1e100.net (216.58.222.14) [1 port]
|S-chain|-<>-127.0.0.1:9050-<><>-216.58.222.14:80-<><>-OK
CONN (0.9788s) TCP localhost > 216.58.222.14:80 => Connected
Discovered open port 80/tcp on 216.58.222.14
Completed Connect Scan at 12:52, 0.69s elapsed (1 total ports)
Nmap scan report for gru06s25-in-f14.1e100.net (216.58.222.14)
Host is up, received user-set (0.69s latency).
Scanned at 2017-10-17 12:52:19 -02 for 1s
PORT   STATE SERVICE REASON
80/tcp open  http    syn-ack

Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 0.98 seconds
```

As you can see, a reverse DNS resolution to Google's DNS server (8.8.8.8) is done without a tunnel. Although we are not leaking our IP directly to the target, it still is

BUGTREE

Born to kill bugs. Live by them.

being leaked. Confirming with tcpdump:

```
# tcpdump -vvv host 8.8.8.8 and udp


tcpdump: listening on wlp2s0, link-type EN10MB (Ethernet), captur
e size 262144 bytes
12:52:19.621972 IP (tos 0x0, ttl 64, id 10688, offset 0, flags [D
F], proto UDP (17), length 72)
    fahrzeug.57577 > google-public-dns-a.google.com.domain: [udp
 sum ok] 32419+ PTR? 14.222.58.216.in-addr.arpa. (44)
12:52:19.882782 IP (tos 0x0, ttl 55, id 31896, offset 0, flags [n
one], proto UDP (17), length 153)
    google-public-dns-a.google.com.domain > fahrzeug.57577: [udp
 sum ok] 32419 q: PTR? 14.222.58.216.in-addr.arpa. 4/0/0 14.222.5
8.216.in-addr.arpa. [23h47m25s] PTR gru06s25-in-f14.1e100.net., 1
4.222.58.216.in-addr.arpa. [23h47m25s] PTR gru06s25-in-f14.1e100.
net., 14.222.58.216.in-addr.arpa. [23h47m25s] PTR gru06s25-in-f1
4.1e100.net., 14.222.58.216.in-addr.arpa. [23h47m25s] PTR gru06s2
5-in-f14.1e100.net. (125)
```

Due to this situation, it seems wise to use the **-n** option (skip name resolution) when scanning through a proxy.

What about other scan types?

We have been using so far the infamous -sT (SYN scan). In fact, -sS is the default scan when running as *root*, while -sT is the default scan for unprivileged users. But what about -sS under proxychains?

```
# proxychains nmap -Pn -n -sS -p80 216.58.222.14 --packet-trace -
vv
ProxyChains-3.1 (http://proxychains.sf.net)

Starting Nmap 7.40 ( https://nmap.org ) at 2017-10-17 12:59 -02
```

BUGTREE

Born to kill bugs. Live by them.

```
Initiating SYN Stealth Scan at 12:59
Scanning 216.58.222.14 [1 port]
SENT (0.0730s) TCP 172.16.100.250:43067 > 216.58.222.14:80 S ttl=
59 id=23061 iplen=44  seq=774587308 win=1024 <mss 1460>
RCVD (0.1134s) TCP 216.58.222.14:80 > 172.16.100.250:43067 SA ttl
=55 id=63050 iplen=44  seq=2278336372 win=42780 <mss 1380>
Discovered open port 80/tcp on 216.58.222.14
Completed SYN Stealth Scan at 12:59, 0.22s elapsed (1 total port
s)
Nmap scan report for 216.58.222.14
Host is up, received user-set (0.040s latency).
Scanned at 2017-10-17 12:59:19 -02 for 0s
PORT   STATE SERVICE REASON
80/tcp open  http    syn-ack ttl 55

Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 0.28 seconds
           Raw packets sent: 1 (44B) | Rcvd: 1 (44B)
```

And in tcpdump:

```
# tcpdump -vvv host 216.58.222.14
tcpdump: listening on wlp2s0, link-type EN10MB (Ethernet), captur
e size 262144 bytes
12:59:19.705036 IP (tos 0x0, ttl 59, id 23061, offset 0, flags [n
one], proto TCP (6), length 44)
    fahrzeug.43067 > gru06s25-in-f14.1e100.net.http: Flags [S], c
ksum 0xb26f (correct), seq 774587308, win 1024, options [mss 146
0], length 0
12:59:19.745452 IP (tos 0x0, ttl 55, id 63050, offset 0, flags [n
one], proto TCP (6), length 44)
    gru06s25-in-f14.1e100.net.http > fahrzeug.43067: Flags [S.],
 cksum 0xe050 (correct), seq 2278336372, ack 774587309, win 4278
0, options [mss 1380], length 0
12:59:19.745507 IP (tos 0x0, ttl 64, id 3672, offset 0, flags [D
F], proto TCP (6), length 40)
    fahrzeug.43067 > gru06s25-in-f14.1e100.net.http: Flags [R], c
```

BUGTREE

Born to kill bugs. Live by them.

```
ksum 0xce28 (correct), seq 774587309, win 0, length 0

3 packets captured
3 packets received by filter
0 packets dropped by kernel
```

From what we conclude the safest option from what we have tested so far is:

```
$ proxychains nmap -Pn -n -sT <target IP>
```

That is all for now, folks! Please note this is a first draft and ~~most certainly~~ might contain ~~multiple~~ some mistakes. Please feel free to share your corrections, suggestions and observations.

BUGTREE

Born to kill bugs. Live by them.

pentest (2)

pentest (1) , nmap (1)

Share Post

Twitter Facebook
Google+

Marcos Valle

Born to kill bugs. Live by them.

← Previous

Next →

BUGTREE

Born to kill bugs. Live by them.

0 Comments Bugtree Login

Recommend Tweet Share Sort by Best

Start the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

Be the first to comment.

Subscribe Add Disqus to your site Disqus' Privacy Policy DISQUS