CRYPTCAT AND NETCAT SCANNING

We have previously looked at using the wonderful tool *netcat* as a quick and simple honeypot script (see here) and used for network pipes (see here). But it can do a lot more. In this article we will be looking at using netcat to scan a target, as well as how to overcome the one major flaw netcat has - it's lack of encryption, by using cryptcat.

Netcat scanning
Amongst it's many uses *netcat* can also be used as a port scanner (for a primer on scanning, see here), while it does lack some features that a dedicated scanning program has, it does still have a few useful tricks. Lets start with a straight scan..

```
nc -n -v -z 192.168.10.13 20-30
192.168.10.13 21 (ftp) open
192.168.10.13 22 (ssh) open
192.168.10.13 23 (telnet) open
```

with this type of scan our target will see something like this..

```
 10.830000 192.168.10.80 -> 192.168.10.13 TCP 33264 > 20 [SYN] Seq=2859658862 Ack=0 Win=5840 Len=0
 10.830000 192.168.10.80 -> 192.168.10.13 TCP 33265 > 21 [SYN] Seq=2860437808 Ack=0 Win=5840 Len=0
 10.830000 192.168.10.80 -> 192.168.10.13 TCP 33265 > 21 [ACK] Seq=2860437809 Ack=687860086 Win=5840 Len=0
 10.830000 192.168.10.80 -> 192.168.10.13 TCP 33265 > 21 [FIN, ACK] Seq=2860437809 Ack=687860086 Win=5840
Len=0
 10.830000 192.168.10.80 -> 192.168.10.13 TCP 33265 > 21 [RST, ACK] Seq=2860437810 Ack=687860086 Win=5840
Len=0
 10.830000 192.168.10.80 -> 192.168.10.13 TCP 33266 > 22 [SYN] Seq=2873895244 Ack=0 Win=5840 Len=0
 10.830000 192.168.10.80 -> 192.168.10.13 TCP 33266 > 22 [ACK] Seq=2873895245 Ack=680618227 Win=5840 Len=0
 10.830000 192.168.10.80 -> 192.168.10.13 TCP 33266 > 22 [FIN, ACK] Seq=2873895245 Ack=680618227 Win=5840
Len=0
 10.830000 192.168.10.80 -> 192.168.10.13 TCP 33266 > 22 [RST, ACK] Seq=2873895246 Ack=680618227 Win=5840
Len=0
 10.830000 192.168.10.80 -> 192.168.10.13 TCP 33267 > 23 [SYN] Seq=2864142448 Ack=0 Win=5840 Len=0
 10.830000 192.168.10.80 -> 192.168.10.13 TCP 33267 > 23 [ACK] Seq=2864142449 Ack=681723873 Win=5840 Len=0
 10.830000 192.168.10.80 -> 192.168.10.13 TCP 33267 > 23 [FIN, ACK] Seq=2864142449 Ack=681723873 Win=5840
Len=0
 10.830000 192.168.10.80 -> 192.168.10.13 TCP 33267 > 23 [RST, ACK] Seq=2864142450 Ack=681723873 Win=5840
Len=0
 10.830000 192.168.10.80 -> 192.168.10.13 TCP 33268 > 24 [SYN] Seq=2875949276 Ack=0 Win=5840 Len=0
 10.830000 192.168.10.80 -> 192.168.10.13 TCP 33269 > 25 [SYN] Seq=2869550720 Ack=0 Win=5840 Len=0
 10.830000 192.168.10.80 -> 192.168.10.13 TCP 33270 > 26 [SYN] Seq=2872892129 Ack=0 Win=5840 Len=0
 10.830000 192.168.10.80 -> 192.168.10.13 TCP 33271 > 27 [SYN] Seq=2870979713 Ack=0 Win=5840 Len=0
 10.830000 192.168.10.80 -> 192.168.10.13 TCP 33272 > 28 [SYN] Seq=2861307562 Ack=0 Win=5840 Len=0
 10.830000 192.168.10.80 -> 192.168.10.13 TCP 33273 > 29 [SYN] Seq=2868576553 Ack=0 Win=5840 Len=0
 10.830000 192.168.10.80 -> 192.168.10.13 TCP 33274 > 30 [SYN] Seq=2870288843 Ack=0 Win=5840 Len=0
```

As you can see this is very easily recognisable as a scan, with the sequential port number checking and the timeframe in which the ports are tested. So lets see if we cannot be a bit sneakier..

```
nc -i 30 -v -r -n -z 192.168.10.13 20-30
192.168.10.13 21 (ftp) open
192.168.10.13 22 (ssh) open
192.168.10.13 23 (telnet) open
```

I have added in the *-i* option to specify a time in seconds to wait before each port check, as well as the *-r* option which will randomise the order in which the ports are checked, lets take a look at what our target see's now..

```
  0.000000 192.168.10.80 -> 192.168.10.13 TCP 31819 > 26 [SYN] Seq=3815677051 Ack=0 Win=5840 Len=0
 30.010000 192.168.10.80 -> 192.168.10.13 TCP 24319 > 22 [SYN] Seq=3843462878 Ack=0 Win=5840 Len=0
 30.010000 192.168.10.80 -> 192.168.10.13 TCP 24319 > 22 [ACK] Seq=3843462879 Ack=1686788269 Win=5840 Len=0
 30.010000 192.168.10.80 -> 192.168.10.13 TCP 24319 > 22 [FIN, ACK] Seq=3843462879 Ack=1686788269 Win=5840
Len=0
 30.020000 192.168.10.80 -> 192.168.10.13 TCP 24319 > 22 [RST] Seq=3843462880 Ack=0 Win=0 Len=0
 60.030000 192.168.10.80 -> 192.168.10.13 TCP 15851 > 27 [SYN] Seq=3871369641 Ack=0 Win=5840 Len=0
 90.050000 192.168.10.80 -> 192.168.10.13 TCP 46343 > 25 [SYN] Seq=3905097860 Ack=0 Win=5840 Len=0
120.070000 192.168.10.80 -> 192.168.10.13 TCP 11103 > 29 [SYN] Seq=3924723237 Ack=0 Win=5840 Len=0
150.080000 192.168.10.80 -> 192.168.10.13 TCP 24366 > 30 [SYN] Seq=3966992870 Ack=0 Win=5840 Len=0
180.100000 192.168.10.80 -> 192.168.10.13 TCP 58480 > 23 [SYN] Seq=3993644398 Ack=0 Win=5840 Len=0
```

```
180.100000 192.168.10.80 -> 192.168.10.13 TCP 58480 > 23 [ACK] Seq=3993644399 Ack=1843324074 Win=5840 Len=0
180.100000 192.168.10.80 -> 192.168.10.13 TCP 58480 > 23 [FIN, ACK] Seq=3993644399 Ack=1843324074 Win=5840
Len=0
180.120000 192.168.10.80 -> 192.168.10.13 TCP 58480 > 23 [RST] Seq=3993644400 Ack=0 Win=0 Len=0
210.120000 192.168.10.80 -> 192.168.10.13 TCP 8734 > 21 [SYN] Seq=4045270195 Ack=0 Win=5840 Len=0
210.120000 192.168.10.80 -> 192.168.10.13 TCP 8734 > 21 [ACK] Seq=4045270196 Ack=1895170596 Win=5840 Len=0
210.120000 192.168.10.80 -> 192.168.10.13 TCP 8734 > 21 [FIN, ACK] Seq=4045270196 Ack=1895170596 Win=5840
Len=0
210.160000 192.168.10.80 -> 192.168.10.13 TCP 8734 > 21 [RST] Seq=4045270197 Ack=0 Win=0 Len=0
240.140000 192.168.10.80 -> 192.168.10.13 TCP 13150 > 20 [SYN] Seq=4068613888 Ack=0 Win=5840 Len=0
270.150000 192.168.10.80 -> 192.168.10.13 TCP 19569 > 24 [SYN] Seq=4096134956 Ack=0 Win=5840 Len=0
300.170000 192.168.10.80 -> 192.168.10.13 TCP 24504 > 28 [SYN] Seq=4130776223 Ack=0 Win=5840 Len=0
```

As you can see, the portscan was randomised, and (I know you cannot see it just by the results, so you will have to take my word for it) the time between each port scanned was 30 seconds. Just a couple of changes to try to get under any defensive measures. Also remember that netcat can scan for TCP and UDP ports (the *-t* or *-u* switches respectively) and can choose the source port it operates from (the *-p* switch).

Cryptcat

Now while *netcat* is without doubt a fine and useful tool, it does not encrypt the traffic flowing through it. Thats where *cryptcat* comes in. It can be found at the farm9 website, and the guys there have added some encryption to the *netcat* program. Cryptcat can still be used as normal *netcat* could be used (for example, to do port scanning) except with the added cryptography benefit. Lets take a pratical look at what I am talking about. We'll start with a simple *netcat* client and server setup..

Server	`nc -l -n -v -x -p 3344`
Client	`echo hi | nc -n -c 192.168.10.13 3344`

When the client executes, the server process will show this..

```
nc -l -n -v -x -p 3344
Connection from 0.99.1.64:8
hi
Received 3 bytes from the socket
00000000  68 69 0A                                          hi.
```

A person monitoring the traffic would have seen this..

```
tcpdump -i eth1 -l -t -q -X "port 3344"
tcpdump: listening on eth1
192.168.10.80.32784 > 192.168.10.13.3344: tcp 0 (DF)
0x0000   4500 003c 1a2d 4000 4006 8ae1 c0a8 0a50        E..<.-@.@......P
0x0010   c0a8 0a0d 8010 0d10 2c07 37db 0000 0000        ........,.7.....
0x0020   a002 16d0 7c06 0000 0204 05b4 0402 080a        ....|...........
0x0030   008a 2df6 0000 0000 0103 0300                  ..-.........
192.168.10.13.3344 > 192.168.10.80.32784: tcp 0 (DF)
0x0000   4500 003c 0000 4000 4006 a50e c0a8 0a0d        E..<..@.@.......
0x0010   c0a8 0a50 0d10 8010 a7e0 47ae 2c07 37dc        ...P......G.,.7.
0x0020   a012 16a0 a8cd 0000 0204 05b4 0402 080a        ................
0x0030   0217 e1b1 008a 2df6 0103 0300                  ......-.....
192.168.10.80.32784 > 192.168.10.13.3344: tcp 0 (DF)
0x0000   4500 0034 1a2e 4000 4006 8ae8 c0a8 0a50        E..4..@.@......P
0x0010   c0a8 0a0d 8010 0d10 2c07 37dc a7e0 47af        ........,.7...G.
0x0020   8010 16d0 d762 0000 0101 080a 008a 2df6        .....b........-.
0x0030   0217 e1b1                                      ....
192.168.10.80.32784 > 192.168.10.13.3344: tcp 3 (DF)
0x0000   4500 0037 1a2f 4000 4006 8ae4 c0a8 0a50        E..7./@.@......P
0x0010   c0a8 0a0d 8010 0d10 2c07 37dc a7e0 47af        ........,.7...G.
0x0020   8018 16d0 64ee 0000 0101 080a 008a 2df6        ....d.........-.
0x0030   0217 e1b1 6869 0a                              ....hi.
192.168.10.13.3344 > 192.168.10.80.32784: tcp 0 (DF)
0x0000   4500 0034 a61e 4000 4006 fef7 c0a8 0a0d        E..4..@.@.......
0x0010   c0a8 0a50 0d10 8010 a7e0 47af 2c07 37df        ...P......G.,.7.
0x0020   8010 16a0 d78f 0000 0101 080a 0217 e1b1        ................
0x0030   008a 2df6                                      ..-.
192.168.10.80.32784 > 192.168.10.13.3344: tcp 0 (DF)
0x0000   4500 0034 1a30 4000 4006 8ae6 c0a8 0a50        E..4.0@.@......P
```

```
0x0010   c0a8 0a0d 8010 0d10 2c07 37df a7e0 47af        ........,.7...G.
0x0020   8011 16d0 d75e 0000 0101 080a 008a 2df6        .....^........-.
0x0030   0217 e1b1                                      ....
192.168.10.80.32784 > 192.168.10.13.3344: tcp 0 (DF)
0x0000   4500 0034 1a31 4000 4006 8ae5 c0a8 0a50        E..4.1@.@......P
0x0010   c0a8 0a0d 8010 0d10 2c07 37e0 a7e0 47af        ........,.7...G.
0x0020   8014 16d0 d75a 0000 0101 080a 008a 2df6        .....Z........-.
0x0030   0217 e1b1                                      ....
192.168.10.80.32784 > 192.168.10.13.3344: tcp 0 (DF)
0x0000   4500 0028 0000 4000 4006 a522 c0a8 0a50        E..(..@.@.."...P
0x0010   c0a8 0a0d 8010 0d10 2c07 37df 0000 0000        ........,.7.....
0x0020   5004 0000 292c 0000 0000 0000 0000             P...),........
```

```
ngrep -d eth1 hi "port 3344"
interface: eth1 (192.168.10.0/255.255.255.0)
filter: ip and ( port 3344 )
match: hi
####
T 192.168.10.80:32799 -> 192.168.10.13:3344 [AP]
  hi.
####
```

Thus you can see that any regular traffic passes through the netcat pipe unencrypted and quite easily viewable. Let's see if we cannot use cryptcat to fix that. Lets start with the same type of server and client process..

Server	cryptcat -l -n -v -p 3344
Client	echo hi | cryptcat -w 2 -n 192.168.10.13 3344

Once the client executes, the server will see this..

```
cryptcat -l -n -v -p 3344
listening on [any] 3344 ...
connect to [192.168.10.13] from (UNKNOWN) [192.168.10.80] 32802
hi
```

So far, exactly what we want. Lets take a look and see what any nosy people would have seen..

```
tcpdump -i eth1 -l -t -q -X "port 3344"
tcpdump: listening on eth1
192.168.10.80.32802 > 192.168.10.13.3344: tcp 0 (DF)
0x0000   4500 003c d82e 4000 4006 ccdf c0a8 0a50        E..<..@.@......P
0x0010   c0a8 0a0d 8022 0d10 7774 d204 0000 0000        ....."..wt......
0x0020   a002 16d0 ba27 0000 0204 05b4 0402 080a        .....'..........
0x0030   008c 0a2a 0000 0000 0103 0300                  ...*........
192.168.10.13.3344 > 192.168.10.80.32802: tcp 0 (DF)
0x0000   4500 003c 0000 4000 4006 a50e c0a8 0a0d        E..<..@.@.......
0x0010   c0a8 0a50 0d10 8022 f5d9 12a9 7774 d205        ...P..."....wt..
0x0020   a012 16a0 f1a5 0000 0204 05b4 0402 080a        ................
0x0030   0219 be04 008c 0a2a 0103 0300                  .......*....
192.168.10.80.32802 > 192.168.10.13.3344: tcp 0 (DF)
0x0000   4500 0034 d82f 4000 4006 cce6 c0a8 0a50        E..4./@.@......P
0x0010   c0a8 0a0d 8022 0d10 7774 d205 f5d9 12aa        ....."..wt......
0x0020   8010 16d0 203b 0000 0101 080a 008c 0a2a        .....;.........*
0x0030   0219 be04                                      ....
192.168.10.80.32802 > 192.168.10.13.3344: tcp 16 (DF)
0x0000   4500 0044 d830 4000 4006 ccd5 c0a8 0a50        E..D.0@.@......P
0x0010   c0a8 0a0d 8022 0d10 7774 d205 f5d9 12aa        ....."..wt......
0x0020   8018 16d0 c651 0000 0101 080a 008c 0a2a        .....Q.........*
0x0030   0219 be04 c604 1a77 e48f 6239 0db2 59bd        .......w..b9..Y.
0x0040   ac72 1eaa                                      .r..
192.168.10.13.3344 > 192.168.10.80.32802: tcp 0 (DF)
0x0000   4500 0034 acae 4000 4006 f867 c0a8 0a0d        E..4..@.@..g....
0x0010   c0a8 0a50 0d10 8022 f5d9 12aa 7774 d215        ...P..."....wt..
0x0020   8010 16a0 205b 0000 0101 080a 0219 be04        .....[..........
0x0030   008c 0a2a                                      ...*
192.168.10.80.32802 > 192.168.10.13.3344: tcp 19 (DF)
0x0000   4500 0047 d831 4000 4006 ccd1 c0a8 0a50        E..G.1@.@......P
0x0010   c0a8 0a0d 8022 0d10 7774 d215 f5d9 12aa        ....."..wt......
0x0020   8018 16d0 7dbf 0000 0101 080a 008c 0a2a        ....}..........*
```

```
0x0030   0219 be04 0978 2bb3 7d3a 7946 f1b9 c605        .....x+.}:yF....
0x0040   e55d 3188 dafd cd                               .]1....
192.168.10.13.3344 > 192.168.10.80.32802: tcp 0 (DF)
0x0000   4500 0034 acaf 4000 4006 f866 c0a8 0a0d        E..4..@.@..f....
0x0010   c0a8 0a50 0d10 8022 f5d9 12aa 7774 d228        ...P..."....wt.(
0x0020   8010 16a0 2048 0000 0101 080a 0219 be04        .....H..........
0x0030   008c 0a2a                                      ...*
192.168.10.80.32802 > 192.168.10.13.3344: tcp 0 (DF)
0x0000   4500 0034 d832 4000 4006 cce3 c0a8 0a50        E..4.2@.@......P
0x0010   c0a8 0a0d 8022 0d10 7774 d228 f5d9 12aa        ....."..wt.(....
0x0020   8011 16d0 1e87 0000 0101 080a 008c 0bba        ................
0x0030   0219 be04                                      ....
192.168.10.13.3344 > 192.168.10.80.32802: tcp 0 (DF)
0x0000   4500 0034 acb0 4000 4006 f865 c0a8 0a0d        E..4..@.@..e....
0x0010   c0a8 0a50 0d10 8022 f5d9 12aa 7774 d229        ...P..."....wt.)
0x0020   8011 16a0 1d26 0000 0101 080a 0219 bf94        .....&..........
0x0030   008c 0bba                                      ....
192.168.10.80.32802 > 192.168.10.13.3344: tcp 0 (DF)
0x0000   4500 0034 d833 4000 4006 cce2 c0a8 0a50        E..4.3@.@......P
0x0010   c0a8 0a0d 8022 0d10 7774 d229 f5d9 12ab        ....."..wt.)....
0x0020   8010 16d0 1cf6 0000 0101 080a 008c 0bba        ................
0x0030   0219 bf94                                      ....
```

```
ngrep -d eth1 hi "port 3344"
interface: eth1 (192.168.10.0/255.255.255.0)
filter: ip and ( port 3344 )
match: hi
##########exit
10 received, 0 dropped
```

Much better, all nicely encrypted, making it that much harder for any eavesdroppers, and thats always a good thing.

Last Words

Netcat really is a great tool, here we looked at just one trick and also at something it has inspired. But it is capable of much more, play around and take a look. As always, have fun and learn.