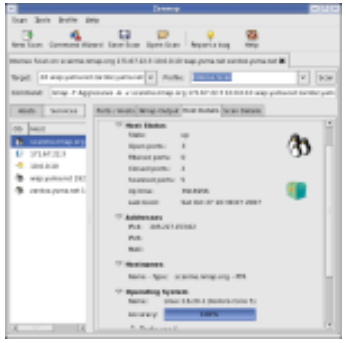

Timing and Performance

One of my highest Nmap development priorities has always been performance. A default scan (**nmap *<hostname>***) of a host on my local network takes a fifth of a second. That is barely enough time to blink, but adds up when you are scanning hundreds or thousands of hosts. Moreover, certain scan options such as UDP scanning and version detection can increase scan times substantially. So can certain firewall configurations, particularly

response rate limiting. While Nmap utilizes parallelism and many advanced algorithms to accelerate these scans, the user has ultimate control over how Nmap runs. Expert users carefully craft Nmap commands to obtain only the information they care about while meeting their time constraints.

Techniques for improving scan times include omitting non-critical tests, and upgrading to the latest version of Nmap (performance enhancements are made frequently). Optimizing timing parameters can also make a substantial difference. Those options are listed below.

Some options accept a `time` parameter. This is specified in seconds by default, though you can append ‘ms’, ‘s’, ‘m’, or ‘h’ to the value to specify milliseconds, seconds, minutes, or hours. So the `--host-timeout` arguments `900000ms`, `900`, `900s`, and `15m` all do the same thing.

`--min-hostgroup <numhosts>`; `--max-hostgroup <numhosts>` (Adjust parallel scan group sizes)

> Nmap has the ability to port scan or version scan multiple hosts in parallel. Nmap does this by dividing the target IP space into groups and then scanning one group at a time. In general, larger groups are more efficient. The downside is that host results can't be provided until the whole group is finished. So if Nmap started out with a group size of 50, the user would not receive any reports (except for the updates offered in verbose mode) until the first 50 hosts are completed.

By default, Nmap takes a compromise approach to this conflict. It starts out with a group size as low as five so the first results come quickly and then increases the groupsize to as high as 1024. The exact default numbers depend on the options given. For efficiency reasons, Nmap uses larger group sizes for UDP or few-port TCP scans.

When a maximum group size is specified with `--max-hostgroup`, Nmap will never exceed that size. Specify a minimum size with `--min-hostgroup` and Nmap will try to keep group sizes above that level. Nmap may have to use smaller groups than you specify if there are not enough target hosts left on a given interface to fulfill the specified minimum. Both may be set to keep the group size within a specific range, though this is rarely desired.

These options do not have an effect during the host discovery phase of a scan. This includes plain ping scans (`-sn`). Host discovery always works in large groups of hosts to improve speed and accuracy.

The primary use of these options is to specify a large minimum group size so that the full scan runs more quickly. A common choice is 256 to scan a network in Class C sized chunks. For a scan with many ports, exceeding that number is unlikely to help much. For scans of just a few port numbers, host group sizes of 2048 or more may be helpful.

`--min-parallelism <numprobes>`; `--max-parallelism <numprobes>` (Adjust probe parallelization)

These options control the total number of probes that may be outstanding for a host group. They are used for port scanning and host discovery. By default, Nmap calculates an ever-changing ideal parallelism based on network performance. If packets are being dropped, Nmap slows down and allows fewer outstanding probes. The ideal probe number slowly rises as the network proves itself worthy. These options place minimum or maximum bounds on that variable. By default, the ideal parallelism can drop to one if the network proves unreliable and rise to several hundred in perfect conditions.

The most common usage is to set `--min-parallelism` to a number higher than one to speed up scans of poorly performing hosts or networks. This is a risky option to play with, as setting it too high may affect accuracy. Setting this also reduces Nmap's ability to control parallelism dynamically based on network conditions. A value of 10 might be reasonable, though I only adjust this value as a last resort.

The `--max-parallelism` option is sometimes set to one to prevent Nmap from sending more than one probe at a time to hosts. The `--scan-delay` option, discussed later, is another way to do this.

`--min-rtt-timeout <time>`, `--max-rtt-timeout <time>`, `--initial-rtt-timeout <time>` (Adjust probe

timeouts)

Nmap maintains a running timeout value for determining how long it will wait for a probe response before giving up or retransmitting the probe. This is calculated based on the response times of previous probes. The exact formula is given in the section called “Idle Scan Implementation Algorithms”. If the network latency shows itself to be significant and variable, this timeout can grow to several seconds. It also starts at a conservative (high) level and may stay that way for a while when Nmap scans unresponsive hosts.

Specifying a lower `--max-rtt-timeout` and `--initial-rtt-timeout` than the defaults can cut scan times significantly. This is particularly true for pingless (`-Pn`) scans, and those against heavily filtered networks. Don't get too aggressive though. The scan can end up taking longer if you specify such a low value that many probes are timing out and retransmitting while the response is in transit.

If all the hosts are on a local network, 100 milliseconds (`--max-rtt-timeout 100ms`) is a reasonable aggressive value. If routing is involved, ping a host on the network first with the ICMP ping utility, or with a custom packet crafter such as Nping that is more likely to get through a firewall. Look at the maximum round trip time out of ten packets or so. You might want to double that for the `--initial-rtt-timeout` and triple or quadruple it for the `--max-rtt-timeout`. I generally do not set the maximum RTT below 100 ms, no matter what the ping times are. Nor do I exceed 1000 ms.

`--min-rtt-timeout` is a rarely used option that could be useful when a network is so unreliable that even Nmap's default is too aggressive. Since Nmap only reduces the timeout down to the minimum when the network seems to be reliable, this need is unusual and should be reported as a bug to the *nmap-dev* mailing list.

`--max-retries <numtries>` (Specify the maximum number of port scan probe retransmissions)

When Nmap receives no response to a port scan probe, it could mean the port is filtered. Or maybe the probe or response was simply lost on the network. It is also possible that the target host has rate limiting enabled that temporarily blocked the response. So Nmap tries again by retransmitting the initial probe. If Nmap detects poor network reliability, it may try many more times before giving up on a port. While this benefits accuracy, it also lengthen scan times. When performance is critical, scans may be sped up by limiting the number of retransmissions allowed. You can even specify `--max-retries 0` to prevent any retransmissions, though that is only recommended for situations such as informal surveys where occasional missed ports and hosts are acceptable.

The default (with no `-T` template) is to allow ten retransmissions. If a network seems reliable and the target hosts aren't rate limiting, Nmap usually only does one retransmission. So most target scans aren't even affected by dropping `--max-retries` to a low value such as three. Such values can substantially speed scans of slow (rate limited) hosts. You usually lose some information when Nmap gives up on ports early, though that may be preferable to letting the `--host-timeout` expire and losing all information about the target.

`--host-timeout <time>` (Give up on slow target hosts)

Some hosts simply take a *long* time to scan. This may be due to poorly performing or unreliable networking hardware or software, packet rate limiting, or a restrictive firewall. The slowest few percent of the scanned hosts can eat up a majority of the scan time. Sometimes it is best to cut your losses and skip those hosts initially. Specify `--host-timeout` with the maximum amount of time you are willing to wait. For example, specify `30m` to ensure that Nmap doesn't waste more than half an hour on a single host. Note that Nmap may be scanning other hosts at the same time during that half an hour, so it isn't a complete loss. A host that times out is skipped. No port table, OS detection, or version detection results are printed for that host.

`--scan-delay <time>`; `--max-scan-delay <time>` (Adjust delay between probes)

This option causes Nmap to wait at least the given amount of time between each probe it sends to a given host. This is particularly useful in the case of rate limiting. Solaris machines (among many others) will usually respond to UDP scan probe packets with only one ICMP message per second. Any more than that sent by Nmap will be wasteful. A `--scan-delay` of `1s` will keep Nmap at that slow rate. Nmap tries to detect rate limiting and adjust the scan delay accordingly, but it doesn't hurt to specify it explicitly if you already know what rate works best.

When Nmap adjusts the scan delay upward to cope with rate limiting, the scan slows down dramatically. The `--max-scan-delay` option specifies the largest delay that Nmap will allow. A low `--max-scan-delay` can speed up Nmap, but it is risky. Setting this value too low can lead to wasteful packet retransmissions and possible missed ports when the target implements strict rate limiting.

Another use of `--scan-delay` is to evade threshold based intrusion detection and prevention systems (IDS/IPS). This technique is used in the section called “A practical example: bypassing default Snort 2.2.0 rules” to defeat the default port scan detector in Snort IDS. Most other intrusion detection systems can be defeated in the same way.

`--min-rate <number>`; `--max-rate <number>` (Directly control the scanning rate)

Nmap's dynamic timing does a good job of finding an appropriate speed at which to scan. Sometimes, however, you may happen to know an appropriate scanning rate for a network, or you may have to guarantee that a scan will be finished by a certain time. Or perhaps you must keep Nmap from scanning too quickly. The `--min-rate` and `--max-rate` options are designed for these situations.

When the `--min-rate` option is given Nmap will do its best to send packets as fast as or faster than the given rate. The argument is a positive real number representing a packet rate in packets per second. For example, specifying `--min-rate 300` means that Nmap will try to keep the sending rate at or above 300 packets per second. Specifying a minimum rate does not keep Nmap from going faster if conditions warrant.

Likewise, `--max-rate` limits a scan's sending rate to a given maximum. Use `--max-rate 100`, for example, to limit sending to 100 packets per second on a fast network. Use `--max-rate 0.1` for a slow scan of one packet every ten seconds. Use `--min-rate` and `--max-rate` together to keep the rate inside a certain range.

These two options are global, affecting an entire scan, not individual hosts. They only affect port scans and host discovery scans. Other features like OS detection implement their own timing.

There are two conditions when the actual scanning rate may fall below the requested minimum. The first is if the minimum is faster than the fastest rate at which Nmap can send, which is dependent on hardware. In this case Nmap will simply send packets as fast as possible, but be aware that such high rates are likely to cause a loss of accuracy. The second case is when Nmap has nothing to send, for example at the end of a scan when the last probes have been sent and Nmap is waiting for them to time out or be responded to. It's normal to see the scanning rate drop at the end of a scan or in between hostgroups. The sending rate may temporarily exceed the maximum to make up for unpredictable delays, but on average the rate will stay at or below the maximum.

Specifying a minimum rate should be done with care. Scanning faster than a network can support may lead to a loss of accuracy. In some cases, using a faster rate can make a scan take *longer* than it would with a slower rate. This is because Nmap's adaptive retransmission algorithms will detect the network congestion caused by an excessive scanning rate and increase the number of retransmissions in order to improve accuracy. So even though packets are sent at a higher rate, more packets are sent overall. Cap the number of retransmissions with the `--max-retries` option if you need to set an upper limit on total scan time.

`--defeat-rst-ratelimit`

Many hosts have long used rate limiting to reduce the number of ICMP error messages (such as port-unreachable errors) they send. Some systems now apply similar rate limits to the RST (reset) packets they generate. This can slow Nmap down dramatically as it adjusts its timing to reflect those rate limits. You can tell Nmap to ignore those rate limits (for port scans such as SYN scan which *don't* treat non-responsive ports as `open`) by specifying `--defeat-rst-ratelimit`.

Using this option can reduce accuracy, as some ports will appear non-responsive because Nmap didn't wait long enough for a rate-limited RST response. With a SYN scan, the non-response results in the port being labeled `filtered` rather than the `closed` state we see when RST packets are received. This option is useful when you only care about open ports, and distinguishing between `closed` and `filtered` ports isn't worth the extra time.

`--nsock-engine epoll|kqueue|poll|select`

Enforce use of a given nsock IO multiplexing engine. Only the `select(2)`-based fallback engine is guaranteed to be available on your system. Engines are named after the name of the IO management facility they leverage. Engines currently implemented are `epoll`, `kqueue`, `poll`, and `select`, but not all will be present on any platform. Use **nmap -V** to see which engines are supported.

`-T paranoid|sneaky|polite|normal|aggressive|insane` (Set a timing template)

While the fine-grained timing controls discussed in the previous section are powerful and effective, some people find them confusing. Moreover, choosing the appropriate values can sometimes take more time than the scan you are trying to optimize. So Nmap offers a simpler approach, with six timing templates. You can specify them with the `-T` option and their number (0–5) or their name. The template names are `paranoid` (`0`), `sneaky` (`1`), `polite` (`2`), `normal` (`3`), `aggressive` (`4`), and `insane` (`5`). The first two are for IDS evasion. Polite mode slows down the scan to use less bandwidth and target machine resources. Normal mode is the default and so `-T3` does nothing. Aggressive mode speeds scans up by making the assumption that you are on a reasonably fast and reliable network. Finally insane mode assumes that you are on an extraordinarily fast network or are willing to sacrifice some accuracy for speed.

These templates allow the user to specify how aggressive they wish to be, while leaving Nmap to pick the exact timing values. The templates also make some minor speed adjustments for which fine-grained control options do not currently exist. For example, `-T4` prohibits the dynamic scan delay from exceeding 10 ms for TCP ports and `-T5` caps that value at 5 ms. Templates can be used in combination with fine-grained

controls, and the fine-grained controls will you specify will take precedence over the timing template default for that parameter. I recommend using `-T4` when scanning reasonably modern and reliable networks. Keep that option even when you add fine-grained controls so that you benefit from those extra minor optimizations that it enables.

If you are on a decent broadband or ethernet connection, I would recommend always using `-T4`. Some people love `-T5` though it is too aggressive for my taste. People sometimes specify `-T2` because they think it is less likely to crash hosts or because they consider themselves to be polite in general. They often don't realize just how slow `-T polite` really is. Their scan may take ten times longer than a default scan. Machine crashes and bandwidth problems are rare with the default timing options (`-T3`) and so I normally recommend that for cautious scanners. Omitting version detection is far more effective than playing with timing values at reducing these problems.

While `-T0` and `-T1` may be useful for avoiding IDS alerts, they will take an extraordinarily long time to scan thousands of machines or ports. For such a long scan, you may prefer to set the exact timing values you need rather than rely on the canned `-T0` and `-T1` values.

The main effects of `T0` are serializing the scan so only one port is scanned at a time, and waiting five minutes between sending each probe. `T1` and `T2` are similar but they only wait 15 seconds and 0.4 seconds, respectively, between probes. `T3` is Nmap's default behavior, which includes parallelization. `-T4` does the equivalent of `--max-rtt-timeout 1250ms --min-rtt-timeout 100ms --initial-rtt-timeout 500ms --max-retries 6` and sets the maximum TCP scan delay to 10 milliseconds. `T5` does the equivalent of `--max-rtt-timeout 300ms --min-rtt-timeout 50ms --initial-rtt-timeout 250ms --max-retries 2 --host-timeout 15m` as well as setting the maximum TCP scan delay to 5 ms.

Nmap Scripting Engine (NSE)

Firewall/IDS Evasion and Spoofing

[Nmap | Sec Tools | Mailing Lists | Site News | About/Contact | Advertising | Privacy]