DoS

The "slowloris" script is not a new attack. But by demonstrating the attack and giving it a personality, it has drawn attention to a significant weakness in Apache HTTPD. We need a response to that, with information on risks and mitigation for server admins.

The slowloris script: threat and limitations

The original slowloris is a perl script, though there are apparently other equivalent scripts floating around. My own testing involved the perl script, on Opensolaris and Linux platforms. It works by opening huge numbers of concurrent connections to the target server, and holding them open so they are unavailable for normal traffic.

The slowloris author notes that the script was ineffective running on Windows, because it only made about 130 concurrent outgoing connections. I observed similar limitations on *X platforms: on Opensolaris it was 252, and on Linux it was 1020. I suspect those could be varied by tuning the host's kernel parameters and/or the Perl build, but I haven't investigated that.

~~The slowloris script is also a big CPU drain on its own host. Running it on my opensolaris box, it took around 50% of the CPU (as shown by top(1)) to hold 252 connections open and trickle data. On linux it was over 99% to hold 1020 connections. Running both slowloris and apache on the linux box, apache responded effortlessly to /server-status requests while servicing the slowloris attack, all while sharing the <1% of CPU left by slowloris with top and the Gnome desktop.~~

[**Update: 29.Apr.2011**] slowloris-perl can be patched (1 line) to reduce CPU drain... (only use 2%, 500 connections in linux-box/threaded, this crash typical server in 15 seconds)

MaxClients

~~Based in this observation, a sufficient (albeit clumsy) defence against a single attacker is to raise maxclients. This is probably a good idea in any case: the defaults shipped by apache and at least some packagers go back to a time when an average server might have 32Mb RAM!~~ 170 clientes drain almost 1Gb-RAM. However, it may create a conflict with applications running on the webserver that cannot reasonably support large numbers of concurrent clients.

Raising MaxClients

The main concern when raising MaxClients is memory usage. With the single-threaded Prefork MPM, this is a serious issue, as each client requires its own process at a marginal cost likely to be significantly in excess of 1Mb RAM, so 1000 slowloris connections will consume gigabytes of RAM. With a threaded MPM such as Worker or Event, each 1Mb memory gives capacity to handle about 10-20 slowloris connections, so a modern server can comfortably accommodate many thousands of clients (though applications may not). Non-Unix MPMs are also threaded, so I would expect them also to work well with high MaxClients settings, but I have no data.

Note that the memory usage reported by tools like ps(1) and top(1) include shared memory, so they report apparent figures that are far higher than apache's actual per-process usage.

Event MPM

The Event MPM is a partially-asynchronous processing model. However, my tests indicate that it is limited by MaxClients in the same way as other MPMs, and doesn't appear to offer any advantage over Worker in mitigating the effect of Slowloris attacks.

Timeout

In http://mail-archives.apache.org/mod_mbox/httpd-users/200711.mbox/%3C22A657FA-0346-47F3-A72F-61EAEEF3F5AE@apache.org%3E , Sander Temme wrote: *If you're being DOS attacked by trickle requests, you could try setting a very low timeout (default is 5 minutes which doesn't seem to be working for you) and perhaps use mod_evasive or somesuch to flag and firewall the bad clients.* TBD: put some numbers to "low timeout".

Resource limits

AcceptFilter

TCP-level defences: e.g. iptables

Modules: e.g. mod_evasive