A SYN flood works by sending SYN packets from randomized spoofed addresses to a target. The target will respond with a SYN-ACK to the spoofed address, however since the address was spoofed, the computer that receives the SYN-ACK will not respond since they did not try to initiate a connection. Meanwhile, the target will maintain an open connection while waiting for the ACK, thus using up resources on the victim system.

SYN cookies allow systems to encode information about the connection into the sequence number of the SYN-ACK sent as a response. Since the information is encoded in the packet, the server can drop information about the connection and rebuild it later, when the other device sends and ACK. If the client was spoofed, the packet will just be dropped and there will be no issue. Since this is more computationally intensive than just keeping track of the connection, this is only done when resources are nearly all used up. Since the target won’t keep track of excess connections, a SYN flood would not cause significant problems and cannot halt system operations.

This is the script used to generate legitimate traffic:

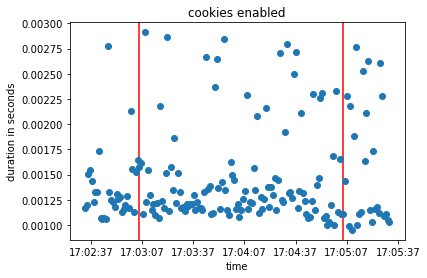
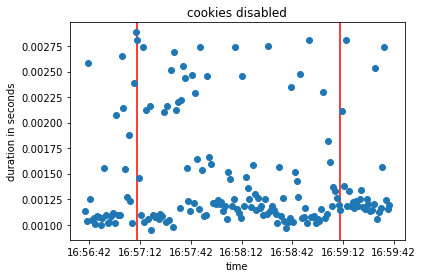
#!/bin/bash

while sleep 1; do curl index.html@bpc063 > /dev/null; done

And this is the script to execute a flooding attack

sudo flooder --dst bpc063 --src 1.1.2.0 --srcmask 255.255.255.0 --highrate 100

Here are the graphs generated when cookies are disabled and enabled respectively:



If the attack was effective, it’s effect was extremely minimal. Cookies might have had less trouble later on in terms of duration for each packet, however it seems like every packet managed to make it’s way through eventually. Perhaps the rate of the TCP flood was too low (I did try with a much higher rate, but the impact seemed minimal).