Google

Issuing SYN cookies in XDP

Netdev 0x14

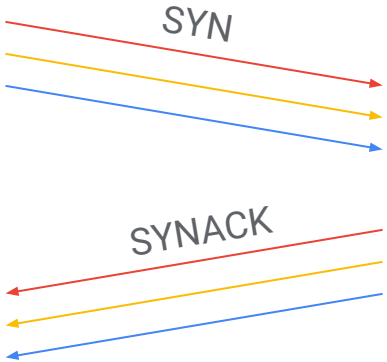
Petar Penkov, Eric Dumazet, Stanislav Fomichev

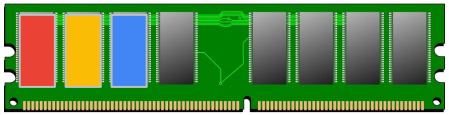
06/2020

Agenda

- Background
- Opportunities
- Design Overview
- Future Work

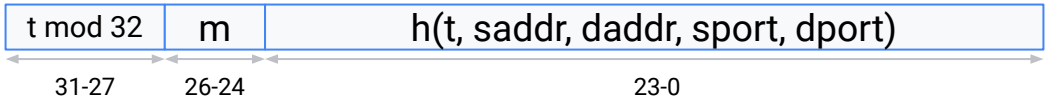
SYN Flood



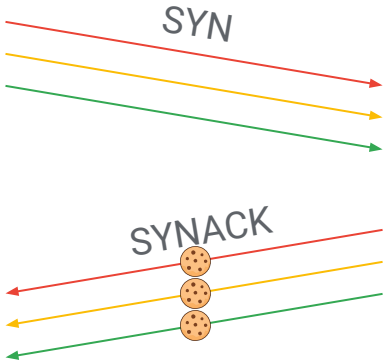


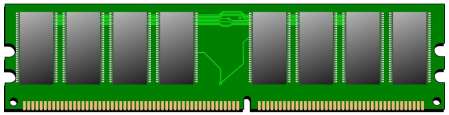
SYN Cookies

- Careful choice of TCP sequence number
  - t - timestamp
  - m - 3-bit mapping of MSS
  - h - 24-bit hash function
- Recomputed on ACK for verification
- No state required!



SYN Cookies (cont.)





Opportunities

Lengthy RX stack

- Multiple allocations, NAPI, RPS/RFS, TCP, IP

Limited visibility

- Global cookie counter

Bypassing the stack with XDP

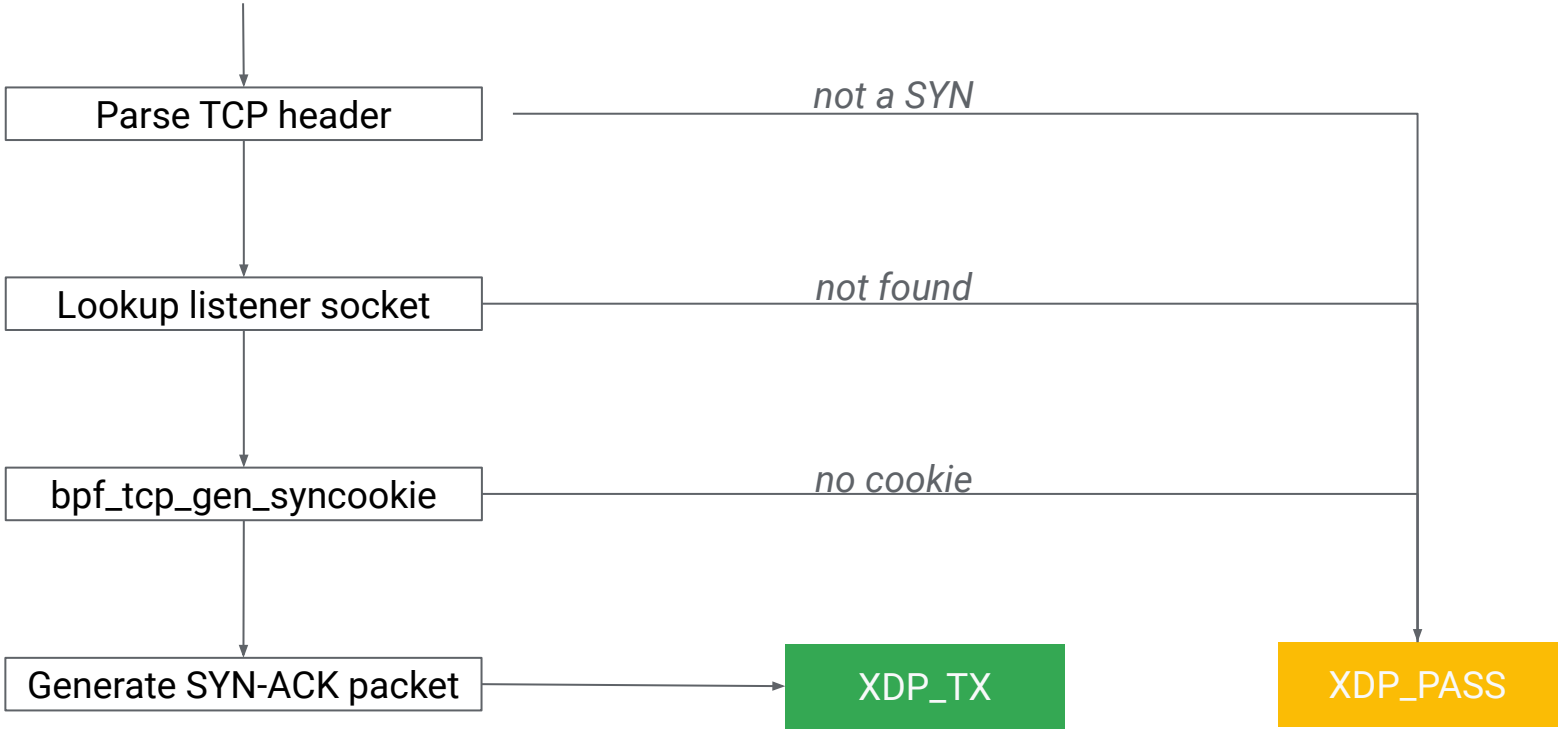
Kernel API (Linux >= 5.4)

```
bpf_tcp_gen_syncookie(struct bpf_sock *sk, void *iph, u32 iph_len,
                      struct tcphdr *th, u32 th_len)
```

- Works on both XDP and TC
- Performs basic checks: header length, correct TCP flags
- Issues a cookie in accordance with net.ipv4.tcp_syncookies
- Internally picks an MSS and embeds it in the cookie
- Returns a 64-bit value:
  - Sign-extended error

  - MSS << 32 | Cookie

BPF Program



Visibility

The kernel exports simple SNMP counters.

In BPF there is an opportunity to enhance the exported metrics to understand ongoing attacks better:

- Per-port counters
- Heavy hitters
- Packet size distribution

Collection can be expensive!

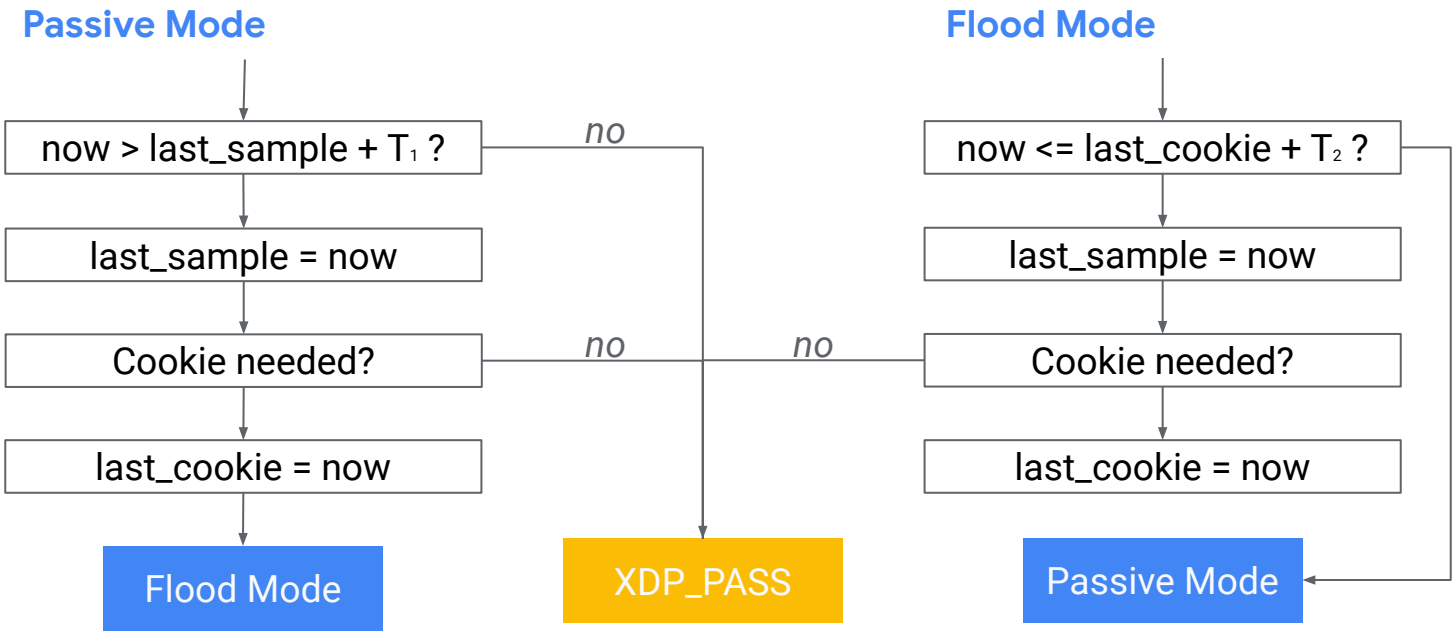
Simplified Implementation

```
int xdp(struct xdp_md *ctx) {
      struct tcphdr *tch = parse_tcp_header(ctx);
      if (!tch || !tch->syn || tch->ack)
            return XDP_PASS;
      struct bpf_sock *sk = bpf_skc_lookup_tcp(...);
      if (!sk)
            return XDP_PASS;
      s64 seq_mss = bpf_tcp_gen_syncookie(...);
      if (seq_mss < 0) {
            bpf_sk_release(sk);
            return XDP_PASS;
      } else {
            u32 cookie = (u32)seq_mss;
            u16 mss = seq_mss >> 32;
            account_cookie(...);
            make_ack_packet(tch, cookie, mss);
            bpf_sk_release(sk);
            return XDP_TX;
      }
}
```

Caveat

1. Most of the time a machine is not under SYN flood
2. The socket lookup can be expensive
3. Therefore, if we did this for every packet, there would be a regression on normal traffic

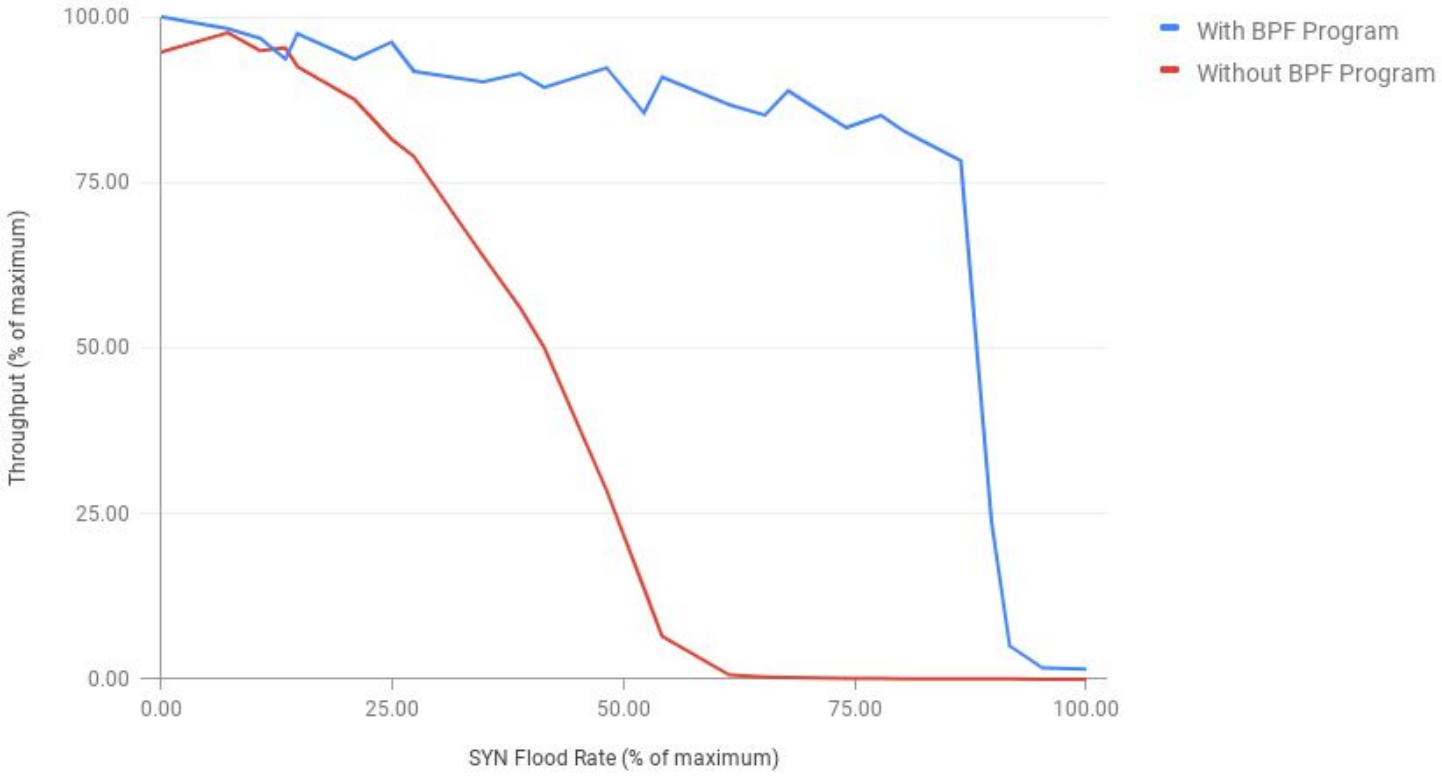
Bimodal Operation



Simplified Implementation

```
/* global variables */
const u64 passive_timer; // T_1
const u64 flood_timer; // T_2
u64 last_sample;
u64 last_cookie;
bool skip() {
      u64 now = bpf_ktime_get_ns();
      if (now < last_cookie + flood_timer) {
            // Flood mode: never skip
            last_sample = now;
            return true;
      }
      if (now > last_sample + passive_timer) {
            // Passive mode: don't skip
            last_sample = now;
            return true;
      }
      // Passive mode: skip
      return false;
}
```

Performance



Challenges & Next Steps

- Parsing variable number of TCP options is challenging for the verifier
- XDP & Multi-buffer packets
- Verifying SYN cookies in XDP - How do we propagate this to the TCP stack?

Questions?

Thank You