Suricata - Task #3768
research: investigate branch prediction vs likely/unlikely macros
06/12/2020 08:00 AM - Victor Julien

Noting something while doing profiling on an older Xeon (E5-2697 v2). During pktgen with only UDP packets, this code sees lots of branch-misses in perf:

```c
p->udph = (UDPHdr *)pkt;
if (unlikely(len < UDP_GET_LEN(p))) {
    ENGINE_SET_INVALID_EVENT(p, UDP_PKT_TOO_SMALL);
    return -1;
}
if (unlikely(len != UDP_GET_LEN(p))) {
    ENGINE_SET_INVALID_EVENT(p, UDP_HLEN_INVALID);
    return -1;
}
SET_UDP_SRC_PORT(p, &p->sp);
SET_UDP_DST_PORT(p, &p->dp);
```

However then I rewrite it to look like this:

```c
p->udph = (UDPHdr *)pkt;
if (likely(len >= UDP_GET_LEN(p))) {
    if (likely(len == UDP_GET_LEN(p))) {
        SET_UDP_SRC_PORT(p, &p->sp);
        SET_UDP_DST_PORT(p, &p->dp);
    }
    p->payload = (uint8_t *)pkt + UDP_HEADER_LEN;
P->payload_len = len - UDP_HEADER_LEN;
P->proto = IPPROTO_UDP;
    return 0;
} else {
    ENGINE_SET_INVALID_EVENT(p, UDP_HLEN_INVALID);
    return -1;
}
else {
    ENGINE_SET_INVALID_EVENT(p, UDP_PKT_TOO_SMALL);
    return -1;
}
```

the branch misses are gone (or reduced to the point that they don't show up in perf).

My assumption has always been that the likely/unlikely annotations would allow the compiler (and/or CPU?) to optimize this to have the same result, but that seems to be untrue.