Notes on Security Models for Transport Eveolution

Eric Rescorla
Mozilla
ekr@rtfm.com

November 9, 2014

1 Introduction

The common assumption seems to be that next-generation transport protocols will – at least for the short-to-medium term – be layered over a UDP transport. Previous attempts to design transports that ran over an IP transport have run into serious deployment problems, and tunnelling over UDP seems like the easiest way to get past them.

The IETF is starting to develop some experience in this area, for example with WebRTC Data Channels, which use a three-layer stack of SCTP over DTLS over UDP. That experience points up some of the security considerations that we would need to consider for solving the more general problem. These include:

- Communications consent and cross-protocol issues
- Interaction with NAT/firewall state tracking
- Interaction with firewall security policy
- Communications security

The rest of this document contains brief notes on each of these.

2 Communications Consent

When a new transport is designed to run over a preexisting transport, we need to worry about its interaction with existing systems which use that transport. In particular, it is important that attackers not be able to make use of the new protocol to attack systems which implement some older UDP-based protocol. This was a particular concern in WebRTC because the target addresses for the protocol were set by Web sites and so there was an opportunity for the attacker to send malicious packets directly to some victim that pretended to be (for instance) DNS or RTP datagrams.

In WebRTC, this issue was addressed in two ways:

1. A reachability check was performed using ICE. The probability that an existing system would generate compatible packets seems extremely small, especially because the ICE responses include an integrity check based on a per-connection key.
2. All traffic is protected with DTLS, thus making it difficult for the attacker to construct targeted packets which look like something else.

While there are no known cross-protocol attacks on this design, it has also only seen modest review. In addition, it is not clear how well it scales. For instance, could multiple application layer protocols built on a similar stack be able to forced to talk to each other incorrectly (bad)? This is not currently a problem for WebRTC but might be if we adopted the same new transport univeersally.

3 Interaction with NAT/firewall state tracking

NATs and Firewalls currently use state tracking to enforce policies on connection lifetimes. For instance, a common policy is to allow only outbound connections. With TCP this is relatively straightforward but because UDP has no explicit connection management with UDP this function is largely performed via timers. This of course causes both false positives and false negatives and requires applications to to send packets very frequently (on the order of 15-30 seconds) to keep the NAT bindings alive. This is especially bad in mobile networks because waking up the radio to send keepalives is expensive.

The obvious solution is explicit state management in the protocol which is interpreted by the middlebox. However, this creates its own problems, including:

- How do you securely inform the middlebox of your state.
- How do you determine when you can stop sending keepalives because all on-path middleboxes know about the new mechanism?

These issues will need to be addressed in order to have a successful general purpose protocol.

4 Interaction with Firewall Security Policy

Many firewall administrators block UDP, both because of the state issues mentioned above and because there is concern over allowing some of the applications that often run over UDP. In order to get past that, it has been suggested to explicitly indicate (e.g., via an ALPN-like mechanism) the appplication that is running over the transport, thus allowing administrators to filter. There is a tension here between privacy issues (and the desire to hide what you are doing) and administrator security preferences that we will need to resolve somehow. One important element of this discussion is that the emerging everything-over-HTTP architecture combined with the move to greater use of TLS is starting to deny administrators this capability already, even before we design a new optimized transport (though of course HTTP can be viewed as a new transport itself, albeit a clunky one).

5 Communications Security

Finally, we need to consider the question of communications security. This is less a technical question than a policy one, namely: what will our policy be with regard to security requirements and the use of insecure transports. Two useful case studies here are WebRTC and HTTP2.

WebRTC. In WebRTC, the decision was made fairly early that all traffic had to be cryptographically protected. Thus, all RTP traffic is secured with SRTP (keyed with DTLS) and the datachannels are implemented using SCTP over DTLS. Despite some initial concerns about deployability, this has proven to be fairly successful, perhaps due to the fact that the major cost was engineering; no extra credentials were required.

HTTP2. HTTP2 had a long and difficult debate over whether to allow cleartext HTTP or to require HTTPS or something in between. Ultimately, there was no consensus to require HTTPS or even unauthenticated TLS. There were too many concerns raised to do justice too here, but among them were concerns about the ability to get certificates and the detailed interaction of the Web security model (and in particular HTTPS) with existing sites.

If the IETF decides to design a new generic transport, we will need to grapple with this issue, which is likely to be even thornier in the generic case than it was with HTTP.