Key Change Strategies for TCP-MD5

Steven M. Bellovin
smb@cs.columbia.edu
http://www.cs.columbia.edu/~smb

The Problem

- RFC 2385 has no key management
- It's hard to change keys – when one side changes its key, it can't talk to the other
- Synchronization between organizations is hard
- (As we move towards universal VoIP, you won't even be able to call your peers to fix the problem if routing is really borked...)

Goal

- Provide a mechanism for loosely synchronized key rollover
- No over-the-wire protocol changes
- No need to co-ordinate code updates with the other end
- Interoperate with existing code base

Algorithm

- Install a second key on the upgraded side
- When a segment arrives, try to validate it against all keys
- The other end switches keys whenever it wants
- When a segment arrives that uses the new key, delete the old one
- Always transmit using the newest key you've seen from the other side
- Optional: fall back to old key (or switch to new one) if too many retransmissions

Why Not Replace 2385?

- Replacing 2385 with a better-designed protocol is a great idea
- We could get key management, HMAC, AES-CBC-MAC, and more
- But designing a new protocol takes time
- Code and test takes even longer
- Roll-out has to be co-ordinated with the far side
- Even roll-out within an organization is painful

Issues

- CPU denial of service – garbage packets have to be tried against all keys
  - *There won't be more than two keys; GTSM will help*
- Best behavior requires integration with TCP retransmit logic
  - *Hooks may already be there to tie to routing and ARP updates*
- This will take time for design/code/test, too
  - *I think it will be noticeably shorter, and it's easier to deploy*

Acknowledgments

- Ron Bonica
- Randy Bush
- Sam Weiler
- Ross Callon