draft-weis-tcp-auth-auto-ks-01

Brian Weis
Chandra Appanna
David McGrew
Anantha Ramaiah

Issues with using TCP-EA manual keys

- The TCP Extended Authentication (TCP-EA) Option (draft-bonica-tcp-auth-04) specifies how to manipulate a set of MAC session keys.
  - The MAC keys are entered into the router configuration manually, and stored in a key chain
- Manual keys are non-optimal with respect to security and operations.
  - Often poorly chosen (based on passwords) and used for too long (never replaced)
  - On the other hand, if BCP are followed they become an operational burden
- Any replacement for RFC 2385 ought to address these problems.

A TCP-EA extension

- We propose a method of generating TCP-EA session keys.
- We proposing doing this without introducing a heavy-weight out-of-band negotiation protocol.
  - Session key generation must be light-weight, in terms of complexity
- This process also enables use of significantly better performing MAC algorithms
  - Algorithms that can't be safely used with manual keying

Design Guidelines

- No persistent storage available for storing ephemeral data (e.g., session keys)
  - Often only have NVRAM/Flash, both with limited write capabilities
- Not enough CPU available to do extensive cryptographic processing during TCP stack processing.
  - The TCP stack isn't the right place to be designing a sophisticated key management or key establishment protocol
- Every TCP segment must be protected

Rejected Methods

- Derive an ephemeral session key using Diffie-Hellman
  - Would make protecting the TCP SYN segment problematic, since no shared secret yet existed.
  - DH is too expensive to do in a TCP stack.
- Use a shared master key to generate a sequence of pair-wise keys
  - Would require frequently-written persistent storage to store the “current” index in the sequence.
- Distribute a session key encrypted under a peer’s public key
  - We currently only have 40 total option bytes, which aren’t enough!
  - Public key algorithms are too expensive to do in a TCP stack.

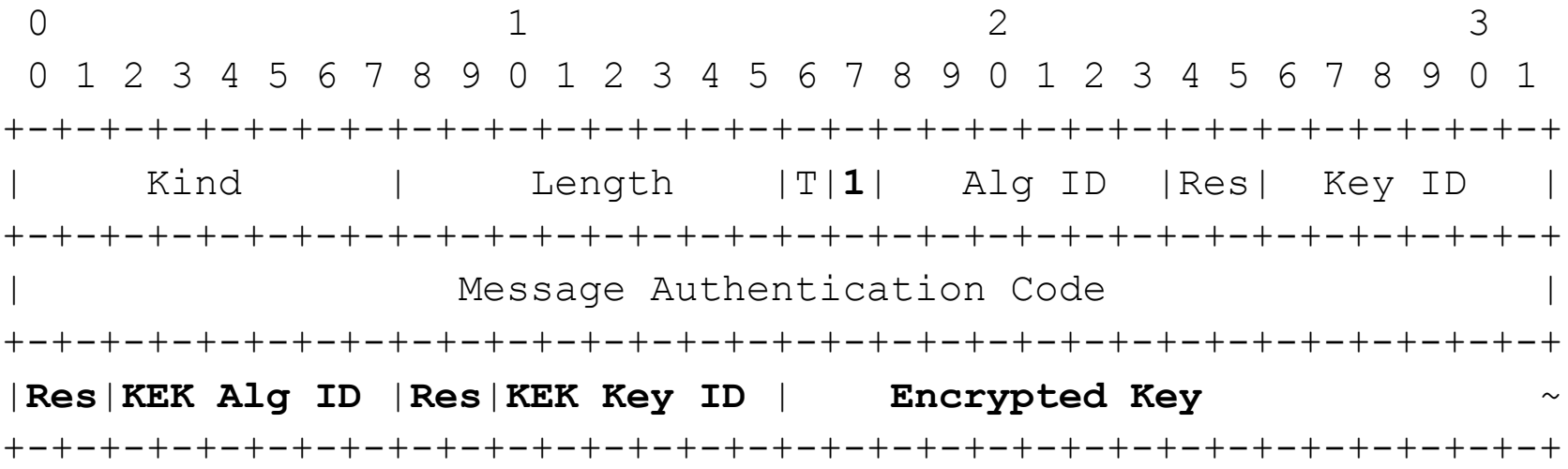
Our Proposal

- A light weight mechanism whereby one TCP endpoint pushes a MAC session key to its peer.
  - Keys are generated using a good random number generator
  - The SYN segment of an Active Open is an obvious time to push a key. Other events may require new keys as well.
- The MAC key is encrypted for confidentiality using a “Key Encrypting Key” (KEK)
  - This KEK is a strong key, and does not need to be frequently changed.

Still using a long term key! What's different?

- Less burden on the operations staff
  - Because the KEK is not a session key, it does not need to be changed as often as RFC 2385 keys.
  - The KEK can be rolled over when necessary using the key rollover scheme described in TCP-EA.
- Better MAC keys
  - Randomly generated MAC keys will be of better quality than ones chosen by operations staff.
  - The MAC keys can be automatically rolled over based on a variety of policies

Resulting Packet Format



- The “K” bit is set to 1
- The Authentication Data field definition is enhanced to include the encrypted key along with the output of the MAC algorithm.

When should a new MAC key be chosen?

- When no key is available, or when policy says a key is about to expire.
- Possible keying events:
  - At the beginning of the TCP session
  - When a TCP sequence number wraps
  - Due to time-based or volume-based policy.

Better performing MAC algorithms

- All MAC algorithms take as inputs a key and the data to be authenticated
- Some MAC algorithms add a third argument called a “nonce”. The nonce is a value that MUST be used only once with that particular key.
  - Using the same {key, nonce} twice can result in a catastrophic cryptographic weakness
  - But these algorithms are optimized in h/w or s/w and tend to be better performing

Nonces

- The most obvious means of generating a set of non-repeating nonces is to use a sequence number.
  - But it must be carried in the packet
  - Using the TCP Sequence Number may be tempting, but isn't sufficiently trustable.
    - I.e., it is a value not under the control of the TCP-EA Option code, so it can't guarantee non-repeatability.

MAC Algorithms using Nonces

The draft specifies the following algorithms that take a nonce as input:

- AES-128-GMAC-96
  - Optimized for implementation in h/w
- AES-128-UMAC-96
  - Optimized for implementation in s/w

Summary

This draft proposes:

- A secure operationally simple in-line means of distributing random TCP-EA session keys
- Making use of this automated method by making defining nonce-based MAC algorithms