Overview

• 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.
• This draft proposes an optional automated key selection mechanism for the TCP-EA Option that improves both security and operational complexity.
History

• This work was first published in draft-weis-tcp-mac-option-00
• We subsequently agreed to make it an extension of draft-bonica-tcp-auth-04
Goals

• Improve the operational characteristics of MAC session keys.
  – Human generated keys (especially those based on passwords) are never as good as randomly generated keys.
  – Requiring an operational staff to continually add new keys is both an operational problem and a security risk.

• Do this without introducing a heavy-weight out-of-band negotiation protocol.
  – Automatic Key Selection must be light-weight, in terms of complexity.

• Enable use of better performing MAC algorithms not suitable for use with manual keying.
Our Proposal

• A light weight mechanism whereby one TCP endpoint pushes a MAC session key to its peer.
  – 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 changed frequently.
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 frequently.
  – The KEK can be rolled over when necessary using the key rollover scheme described in TCP-EA.

• Better MAC keys!
  – The generated MAC keys are of better quality than ones chosen by operations staff.
  – The MAC keys will be automatically rolled over based on a variety of policies
Fitting into TCP-EA

<table>
<thead>
<tr>
<th>Kind</th>
<th>Length</th>
<th>T</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Alg ID</td>
<td>Res</td>
<td>Key ID</td>
<td></td>
</tr>
</tbody>
</table>

| Authentication Data | // |

- 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.
Resulting Packet Format

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Kind | Length | T | Alg ID | Key ID |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Message Authentication Code |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Encrypted Key |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
Sender Processing

• When a TCP endpoint needs to choose a new MAC key it takes the following steps:
  – Randomly generates a MAC key using a strong RNG or PRNG algorithm and places it in a TCP-EA key chain
  – Encrypts the MAC key with the KEK, and places it in the TCP-EA payload
  – Creates the packet:
    • Sets the K bit to 1
    • Performs the MAC calculation described in Section 7 of TCP-EA
Receiver Processing

• Anytime a TCP endpoint receives a TCP-EA packet with the K bit set to 1:
  – Extract and decrypt the MAC key with the KEK matching the KEK Key ID in the segment
  – Performs the MAC calculation described in Section 7 of TCP-EA.
  – If the decrypted key authenticates the packet, places the new MAC key in a TCP-EA key chain.
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.
Example: Beginning of a TCP session

Router1

SYN
SYN, ACK
ACK
(Etc.)

MAC Option(AuthData|EncKey)
MAC Option(AuthData)
MAC Option(AuthData)
MAC Option(AuthData)

Router2

...
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.
Packet Format including a Sequence Number

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

`+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Kind      |     Length    |T|0| Alg ID |Res| Key ID   |
|+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Sequence Number |
|+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Message Authentication Code |

Of course, when K=1 then an encrypted key payload will also be included.
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
Questions?