Lecture 4.1:

Extensible Authentication Protocol

Recommended reading: RFC 3748, June 2004

EAP

- Initially developed for PPP
  - But now highly successful in LANs & WLANs
- Directly runs over data-link
  - Does not require IP
- Developed to provide a generic framework to support multiple authentication protocols
  - And proprietary ones!!
    - Otherwise PPP should be customized to support new AUTH
  - Particularly effective when used with backend AUTH server
    - PPP authenticator = pass-through to e.g. RADIUS/Diameter server
    - No need to update PPP authenticator when ISP change AUTH

EAP (partially) works directly from client to server
**EAP advantages**

- **EAP can support multiple authentication mechanisms without having to pre-negotiate a particular one.**
- **NAS** (e.g., a switch or access point) do not have to understand each authentication method
  - MAY act as a pass-through agent for a backend authentication server.
- **Separation of the authenticator from the backend authentication server simplifies credentials management and policy decision making.**

---

**A note on EAP encapsulation in RADIUS (RFC 2869/3579)**

Whole EAP packet encapsulated into a RADIUS attribute (EAP-Message attribute)

RADIUS message authenticator: mandatory to protect integrity of EAP message included

Otherwise MITM attack may tamper EAP packet/session
EAP over PPP

- **Support of EAP added in PPP:**
  - PPP Authentication protocol option (03): 0xc227
  - EAP authentication protocol value in PPP: 0xc227

  ```
<table>
<thead>
<tr>
<th>Type</th>
<th>Length</th>
<th>Auth-protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>03</td>
<td>04</td>
<td>c227=EAP</td>
</tr>
</tbody>
</table>
  ```

- Though EAP negotiated during LCP as “auth protocol”…
- … IT IS NOT an authentication protocol!!
  - AUTH will be chosen later on during the Auth Phase
  - Only when further info from user will be obtained

---

EAP packet format

- **Code:**
  - 1: request
  - 2: response
  - 3: Success
  - 4: Failure

- **Identifier:**
  - 1 byte
  - To match requests with responses

- **Length:**
  - 2 byte
  - Length including header
  - At least 4 (code+id+len)

<table>
<thead>
<tr>
<th>Request/Response Packet Format</th>
<th>Code</th>
<th>Identifier</th>
<th>Length</th>
<th>Type</th>
<th>Type-Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 byte</td>
<td>1 byte</td>
<td>2 bytes</td>
<td>1 byte</td>
<td>***</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Success/Failure Packet Format</th>
<th>Code</th>
<th>Identifier</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 byte</td>
<td>1 byte</td>
<td>2 bytes</td>
</tr>
</tbody>
</table>

=len=4, no data

---

--- Giuseppe Bianchi ---

--- Giuseppe Bianchi ---
### “standard” basic Types

(specified in original RFC; about 40 EAP Types @ 2004)

<table>
<thead>
<tr>
<th>Type #</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Identity</td>
<td>Mandatory, special case type</td>
</tr>
<tr>
<td>2</td>
<td>Notification</td>
<td>Mandatory, special case type</td>
</tr>
<tr>
<td>3</td>
<td>Nak</td>
<td>Mandatory, special case type, response only</td>
</tr>
<tr>
<td>4</td>
<td>MD5-Challenge</td>
<td>Mandatory</td>
</tr>
<tr>
<td>5</td>
<td>One Time Password (OTP)</td>
<td>Optional</td>
</tr>
<tr>
<td>6</td>
<td>Generic Token Card (GTC)</td>
<td>Optional</td>
</tr>
<tr>
<td>254</td>
<td>Expanded Type</td>
<td>Support recommended</td>
</tr>
<tr>
<td>255</td>
<td>experimental</td>
<td>Unspecified, used for trials &amp; tests</td>
</tr>
</tbody>
</table>

Special case types: do not specify an AUTH method but are special EAP Protocol packets

---

### Expanded type

- \(\text{Type} = 254 (0xFE) = \text{Expanded type}\)
- Increases number of types from 255 to \(2^{32}-1\)
  - Type 0 = reserved
  - Vendor-ID = 0 → IETF (expanded Type space)
- Allows to deploy proprietary auth methods
  - Vendor-ID ≠ 0 → vendor-specific

---
Identity

- used to query the identity of the peer
  - Data may include optional displayable message to prompt the peer
    - If messages need to be notified later on, use the EAP Notification Type (#2)
    - Identity sent in clear text
- generally identity request starts the authentication process
- Typical approach:
  - Identity request starts from NAS
  - Identity response used to route to the proper backend AUTH server (e.g. via realm)
  - Specific authentication mechanism may be based on user identity
    - Novel issue! Selection of AUTH only AFTER having discovered user identity!
- Identity protection mechanisms (abbreviated identity response) may apply
  - For example send only realm and not user id if routing is the only purpose

--- Giuseppe Bianchi

Authentication method negotiation through Nak

- Reply to a Request when desired authentication Type unacceptable
- Reply to a Request which uses expanded type (254) but peer does not support it
- Nak contains 1+ authentication Types desired by the Peer.
  - Type 0 used to indicate that sender has no viable alternatives
  - Type 254 used to indicate that request may be reformulated as expanded type

--- Giuseppe Bianchi
Negotiation through Expanded Type Nak

- Same operation as Nak but extended format:
  - 4 bytes header
  - 8 bytes Expanded Nak
  - Nx6 bytes indicating acceptable authentication methods
  - mixed IETF/proprietary
  - N>=1  minimum response size = 20

### Example 1: no viable alternatives

<table>
<thead>
<tr>
<th>Code (resp)</th>
<th>Identifier (= request)</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type 254</td>
<td>Vendor ID 0 (IETF)</td>
<td>20</td>
</tr>
<tr>
<td>Type 254</td>
<td>Vendor Type 3 (Nak)</td>
<td></td>
</tr>
<tr>
<td>Type 254</td>
<td>Vendor ID 0 (no viable alternative)</td>
<td></td>
</tr>
</tbody>
</table>

### Example 2: two alternatives (one propr.)

<table>
<thead>
<tr>
<th>Code (resp)</th>
<th>Identifier (= request)</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type 254</td>
<td>Vendor ID 0 (IETF)</td>
<td>28</td>
</tr>
<tr>
<td>Type 254</td>
<td>Vendor Type 5 (One Time Password)</td>
<td></td>
</tr>
<tr>
<td>Type 254</td>
<td>Vendor ID 0 (IETF)</td>
<td>20 (MIT)</td>
</tr>
</tbody>
</table>

---

EAP-MD5-Challenge

- Analogous to CHAP-MD5, but “abstracted” over EAP
- Request/Response format: identical to CHAP, but encapsulated in EAP

<table>
<thead>
<tr>
<th>Code (req)</th>
<th>Identifier</th>
<th>Length</th>
<th>Type (MD5)</th>
<th>Val-size</th>
<th>Challenge</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>xxx</td>
<td>&gt;=6</td>
<td>4 (MD5)</td>
<td>1 byte</td>
<td>***</td>
<td>***</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Code (resp)</th>
<th>Identifier (= request)</th>
<th>Length</th>
<th>Type (MD5)</th>
<th>Val-size</th>
<th>Response</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>xxx</td>
<td>&gt;=6</td>
<td>4 (MD5)</td>
<td>1 byte</td>
<td>***</td>
<td>***</td>
</tr>
</tbody>
</table>
A few EAP tech details

- **Retransmission**
  - Explicitly supported
  - Link reliability not guaranteed (e.g. WLAN)
  - Retransmission limit not specified (but 3-5 is reasonable)
  - Retransmission timer: dynamically estimated
  - May significantly differ based on lower layer!
  - Retransmission rule: use same identifier
  - Different from PPP rules!

- **Duplicate detection**
  - Explicitly supported
  - Since a retransmission may occur before the response is received (e.g. due to LAN/WLAN delay)

- **Fragmentation & reassembly**
  - MUST be supported by methods which may generate packets greater than maximum threshold (1020 bytes)