Shared Key Authentication for the TLS Protocol

Barbara Fox

(Internet Draft by Daniel Simon)

Microsoft Corporation

Why Shared Key Authentication?

- Passwords still in wide use for client auth.
  - portability
  - backwards compatibility = customer need!
  - more familiar and understood than public-key
- Typically sent by client over (only-)server-authenticated SSL/PCT (TLS) connection (eg., FTP, Telnet)
- Sometimes use challenge-response

An Insecure Combination

- Many TLS connections are weakly encrypted
  - weak encryption by default for export reasons
  - encryption turned off for efficiency
- Hence a poorly-chosen password may be insecure, even over TLS connection
- Challenge-response protocol doesn't help!
  - Attacker first brute-force attacks TLS connection, then dictionary-attacks password

Solution

- **Standard** shared-key (including password) authentication protocol, incorporated into TLS for interoperability
- Shared key/password can be protected by strong MAC key, even if encryption is weakened or turned off
- Even weak passwords (eg., 4-digit PINs) invulnerable to (offline) attack

Our Proposal

- Three new handshake message types:
  - **SharedKeys**: client lists "auth. services" with whom a key is shared (if more than one)
  - **SharedKeyRequest**: server selects an auth. service and relays its challenge, if necessary
  - **SharedKeyVerify:** client's response
- Support indicated by new CipherSuite
- Provision for "passthrough authentication"

Protocol Flow

- Optional SharedKeys message appended to ClientHello
  - Contains list of supported auth. services
- Otherwise, mirrors signature-based auth.
  - SharedKeyRequest replaces CertificateRequest
    - Auth. service(s), optional extra challenge
  - SharedKeyVerify replaces CertificateVerify
    - Auth. service, identity, auth. response

Response Format

- Similar to signature-based response, with MAC instead of signature
- New single-purpose “auth_write_key” that can be sent to auth. service in pass-through case
- Exact format may be adjusted based on outcome of TLS process (final HMAC format, PRF/MAC primitives)

Conclusions

- Nobody who prefers public-key-based authentication ever needs to implement (let alone use) the shared-key variety
- But shared-key-based authentication will happen anyway…
- ...So let’s make it secure

Status

- Internet Draft (ietf-tls-passauth-00.txt)
- Test server up and running (see TLS mailing list archives for details)
- Independent implementations = 2
- Will ship with MS products via SCHANNEL.DLL in 1997
- Included in “TLS Shared Key Authentication Protocol “ Proposal

Comments/Questions

- bfox@microsoft.com
- dansimon@microsoft.com
- listserv = ietf-tls@w3.org