**Protocolli di sicurezza:**

Ci spostiamo dalla LAN alla rete internet.
Protocolli di sicurezza sono:

1. SSH
2. TLS→offre sicurezza ai livelli superiori, così risulta trasparente rispetti al livello applicativo
3. IPsec→tutti i protocolli di livello più alto utilizzano i meccanismi di sicurezza che offre

Tre requisiti:

1. Authentication→digital signature
2. Confidentiality→encryption
3. Integrity→MAC basato su funzioni hash

Abbiamo visto che i meccanismi di base di IP non sono sicuri.
TELNET→ Non è stato sviluppato pensando alla sicurezza. Non ha cifratura e autenticazione. Ci sono falle di sicurezza nell'implementazione stessa.
E' inutile cercare di renderlo sicuro, bisogna ripartire da capo e creare un nuovo protocollo.

**SSH:**

"Secure shell".
Costituito da 3 RFC:

1. Transport layer protocol
2. User authentication protocol
3. Connection layer protocol

Si possono usare diversi algoritmi di cifratura:

- ❏ EdDSA ECDSA, RSA e DSA per la crittografia a chiave pubblica
- ❏ ECDH e Diffie-Hellman per lo scambio di chiavi
- ❏ HMAC, AEAD e UMAC per MAC
- ❏ AES (e deprecato RC4, 3DES, DES[30]) per la crittografia simmetrica
- ❏ AES-GCM[31] e ChaCha20-Poly1305 per la crittografia AEAD
- ❏ SHA (e MD5 deprecato) per key fingerprint

Ci sono diverse implementazioni di questo protocollo, la più famosa è OpenSSH.

*Transport layer protocol:*

Eseguito su TCP/IP, questo layer gestisce lo scambio iniziale di chiavi, l'autenticazione del server (con public key), abilita la cifratura, la compressione e la verifica dell'integrità.
Espone ai livelli superiori un'interfaccia per inviare e ricevere pacchetti di testo normale.
Il livello di trasporto organizza anche il ri-scambio delle chiavi, di solito dopo che sono stati trasferiti 1 GB di dati o dopo che è trascorsa 1 ora.

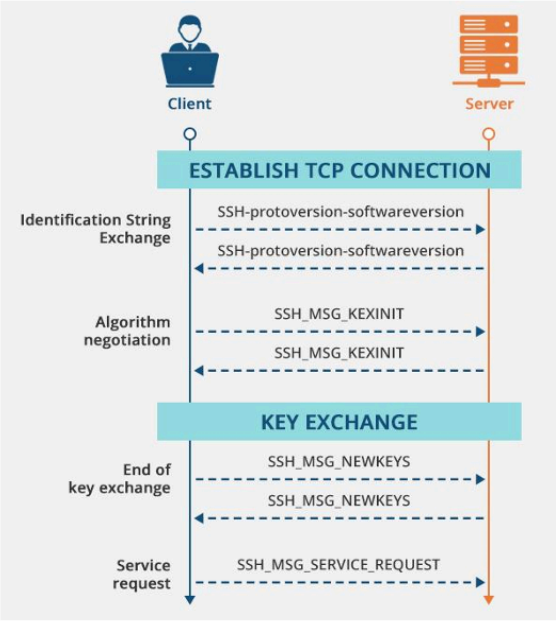
*User authentication protocol:*

Questo strato gestisce l'autenticazione del client e fornisce diversi metodi.
L'autenticazione è guidata dal client:è il client a richiedere una password, Il server risponde semplicemente alle richieste di autenticazione del client.
I metodi di autenticazione dell'utente ampiamente utilizzati includono i seguenti:

- ❏ password: un metodo per l'autenticazione diretta tramite password, inclusa una funzionalità che consente di cambiare la password. Non tutti i programmi implementano questo metodo.

❏ publickey: un metodo per l'autenticazione basata su chiave pubblica, di solito supporta almeno coppie di chiavi DSA, ECDSA o RSA, con altre implementazioni che supportano anche certificati X.509.
❏ keyboard-interactive (RFC 4256): un metodo versatile in cui il server invia uno o più prompt per inserire informazioni.
❏ metodi di autenticazione GSSAPI che forniscono uno schema estensibile per eseguire l'autenticazione SSH utilizzando meccanismi esterni come Kerberos 5 o NTLM.

*Connection layer protocol:*
Questo strato definisce il concetto di canali, richieste di canale e richieste globali tramite cui vengono forniti i servizi SSH.
Una singola connessione SSH può ospitare contemporaneamente più canali, ognuno trasferendo dati in entrambe le direzioni.
Le richieste di canale sono utilizzate per trasmettere dati specifici del canale fuori banda, come la dimensione cambiata di una finestra terminale o il codice di uscita di un processo lato server.
Inoltre, ogni canale esegue il proprio controllo del flusso utilizzando la dimensione della finestra di ricezione. Il client SSH richiede l'inoltro di una porta lato server utilizzando una richiesta globale.
I tipi di canale standard includono:
❏ shell per le shell terminali, richieste SFTP ed exec (incluse le trasferimenti SCP)
❏ direct-tcpip per connessioni inoltrate da client a server
❏ forwarded-tcpip per connessioni inoltrate da server a client



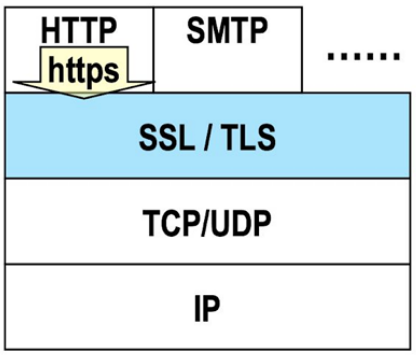

Il server si autentica con una chiave pubblica.
SSH non utilizza certificati X.509
L'implementazione del client richiede all'utente di autenticare "manualmente" la chiave pubblica del server; dopo questo l'identità del server è vincolata alla chiave pubblica inviata dal server, i successivi accessi non richiederanno questa autenticazione manuale.
La coppia (id, chiave pubblica) è memorizzata in ~/.ssh/known_host
Come verifichiamo l'impronta digitale della chiave del server?

❏ ssh-keygen -l -f /etc/ssh/ssh_host_rsa_key
Quando questa fingerprint cambia ci viene impedito l'accesso perchè potrebbe essere un attacco.
Per raggirare questo comportamento devo eliminare manualmente l'associazione tra l'id del server e la chiave.

*Mutual public key authentication:*
Di default il client si autentica con username:password (quelli dell'utente del SO utilizzato per l'accesso).
Per ragioni di sicurezza potremmo volere che anche client SSH siano obbligati a eseguire l'autenticazione a chiave pubblica.
Come facciamo?
❏ generare una coppia di chiavi privata/pubblica
❏ "installare" in modo sicuro la chiave pubblica del client nel server (questo può essere fatto con il comando ssh-copy-id o con chiave pubblica inviata per email o copiata con scp)

*Conclusioni:*
SSH implementa meccanismi di sicurezza a livello applicativo
❏ Anche se è possibile incapsulare traffico IP generico in canali SSH, SSH è un protocollo applicativo specifico
❏ E se volessimo proteggere altri protocolli applicativi? (ad esempio: HTTP, FTP, SIP, ecc...)
    O incapsuliamo questi all'interno di tunnel SSH (non è la soluzione migliore)
    Oppure estendiamo questi altri protocolli applicativi non sicuri per implementare i loro meccanismi di sicurezza
    O implementiamo i meccanismi di sicurezza richiesti a livelli inferiori (questo è effettivamente l'approccio attuale)

**TLS:**
Transport layer security.
Fornisce sicurezza per comunicazione end to end.
Opera a livello 4 dello stack.
Stabilisce una sessione con meccanismi di sicurezza.
Oggi dovremmo usare TLS1.3, l'unico senza falle di sicurezza nell'implementazione.



transport layer security

Cosa fa TLS?

1. Stabilire una sessione (fase di Handshake di TLS)
    a. Concordare sugli algoritmi
    b. Condividere segreti
    c. Eseguire l'autenticazione
2. Trasferire dati dell'applicazione, garantendo:
    a. Privacy della comunicazione
    b. Crittografia simmetrica
    c. Integrità dei dati
    d. Codice di autenticazione dei messaggi chiave (HMAC)
3. L'approccio di TLS è due in uno: contiene sia il protocollo per stabilire la sessione che il protocollo che consegna dati e applica servizi di sicurezza.

Attacchi a TLS1.2:

1. Al protocollo→downgrade e renegotiation attack
2. Agli algoritmi→CRIME e BEAST
3. All'implementazione→HEARTBLEED

Cosa è stato fatto in TLS 1.3?

1. Pruning dei cifrari deboli, mantenuto solo AEAD, gli unici a garantire perfect forward secrecy
2. Handshake più veloce e più sicuro→ la comunicazione sicura inizia già dopo 1 RTT
3. PSK combinata con DHE
4. 0-RTT→ se conosco il server posso direttamente stabilire una connessione sicura

**OpenSSL:**

OpenSSL è un toolkit crittografico che implementa i protocolli di rete Secure Sockets Layer (SSL v2/v3) e Transport Layer Security (TLS) e gli standard crittografici correlati richiesti da essi.

I componenti principali sono:

1. Libreria crittografica: libcrypto
2. Libreria di protocollo SSL/TLS: libssl
3. Programma openssl → strumento a riga di comando per utilizzare le varie funzioni crittografiche della libreria crittografica di OpenSSL dalla shell.
   Può essere utilizzato per: creare e gestire chiavi private, chiavi pubbliche e parametri, fare operazioni crittografiche con chiave pubblica, creare certificati X.509, CSR e CRL, calcolare il digest dei messaggi, cifrare e decifare, testare client e server SSL/TLS, gestire posta firmata o crittografata con S/MIME e richiedere, generare e verificare timestamp.