1 Lezione del 18-11-25

1.1 Riassunto di Internet

Cerchiamo di mettere insieme quanto visto sull'infrastruttura Internet osservando la vita di un messaggio relativo ad un applicazione che conosciamo bene, cioè una *richiesta Web*.

Poniamo di avere un host, come ad esempio un portatile o un telefono cellulare, collegato alla rete locale (magari una LAN istituzionale su Ethernet).

1. Appena arrivato sulla rete, l'host non avrà indirizzo IP, per cui dovrà inviare un messaggio di *Discovery* **DHCP** sulla sua rete locale. Ricordiamo che DHCP è un protocollo di livello application, per cui questo messaggio verrà incapsulato in un segmento UDP, incapsulato in un datagramma IP, incapsulato in un frame Ethernet 802.3 con indrizzo di *broadcast*.

   Il server DHCP riceve il messaggio, demultiplexando il datagramma IP e il segmento UDP, e risponde. Questo processo si ripete per il cosiddetto *4-way handshake* DHCP (Discovery, Offer, Request, Acknowledge).

   Al termine della trasmissione di *Acknowledge*, quindi, l'host conosce l'indirizzo del server DHCP, del DNS locale, il gateway predefinito e il suo indirizzo IP all'interno della rete;

2. Il secondo passaggio è quello della risoluzione del nome di dominio attraverso il **DNS**. Verrà quindi creata una query DNS (che è nuovamente un protocollo di livello application), che verrà incapsulata in un segmento UDP, incapsulato in un datagramma IP, incapsulato nuovamente in un frame Ethernet 802.3 inviato al server DNS della rete locale.

   Il server DNS riceve tale messaggio, e in maniera iterativa o ricorsiva interroga i server DNS dal top-level (o più in basso se ha cache), fino al server autoritativo associato al dominio cercato. Da questo ottiene un record contenente l'indirizzo IP del Web server con cui l'host desidera comunicare. Questo indirizzo viene inserito in una risposta DNS inviata all'host.

   A questo punto l'host riceve la richiesta: conosce l'indirizzo IP del Web server con cui deve parlare;

3. Si stabilisce una connessione **TCP** col Web server, attraverso il classico *3-way handshake* TCP sul socket di ascolto.

   Il Web server riceve l'handshake, e da lato suo provvede ad aprire una nuova connessione TCP con relativo socket. Le modalità con cui gestisce questo nuovo socket potrebero essere variegate (processi, thread, server iterativi, ecc...).

   Al termine di questa fase sia host che Web server hanno dei socket TCP attivi e sono pronti a scambiarsi dati;

4. Infine, l'host, che a questo punto si comporterà da client **HTTP**, inviando una richiesta HTTP ad esempio per la pagina `index.html` del Web server contattato, sulla porta 80. La richiesta verrà incapsulata su TCP, e quindi in datagrammi IP e frame Ethernet che raggiungeranno il gateway di rete. Da lì in poi attraverso il protocollo IP la richiesta raggiungerà il server.

Il server riceverà la richiesta, che verrà demultiplexata da TCP sulla porta 80, cercherà la risorsa desiderata e invierà indietro una risposta HTTP che la contiene. Questo processo si svolge analogamente a prima, ma nella direzione opposta.

Quando il client riceve la pagina, il browser ivi installato potrà accedere al messaggio di risposta HTTP inviato dal server, accedere ai dati HTML che contiene, e visualizzare la pagina sullo schermo.

1.2 Sicurezza in rete

Un problema che abbiamo finora ignorato delle reti è la **sicurezza**. La sicurezza di rete si pone di assicurare:

- **Confidenzialità**: solo sorgente e destinatario dei messaggi in rete dovrebbero essere in grado di decodificare il contenuto del messaggio. In questo si vuole che il *trasmettitore* **cifri** il messaggio, e il *ricevitore* lo **decifri**.

  Questo permette di evitare attachi del tipo *man-in-the-middle* o di *sniffing*, cioè a eventuali malintenzionati di *"spiare"* la conversazione fra 2 o più host;

- **Autenticazione**: si vuole che ricevitore e trasmettitore possano confrmare l'identità l'uno dell'altro.

  Questo è chiaramente importante nel caso di sistemi che devono verificare l'identità degli utenti (pagine di login, ecc...);

- **Integrità dei messaggi**: vogliamo assicurarci che i messaggi in transito non vengano modificati, o comunque che non vengano modificati senza la possibilità di rilevare le modifiche.

  Anche questo torna utile nel caso di attacchi del tipo *man-in-the-middle* (quando non *"spiamo"* soltanto i messaggi, ma li manomettiamo), o semplicemente per combattere le imprecisioni date dalle caratteristiche di inaffidabilità dei mezzi di comunicazione stessi;

- **Accesso e disponibilità**: vogliamo assicurare che i servizi Internet che offriamo siano accessibili e disponibili agli utenti, anche di fronte a elementi di disturbo o carico consistente.

  Questo è particolarmente importante per il paradigma client-server, dove vogliamo che il server sia disponibile in qualsiasi momento, e si può incorrere in attacchi **DoS** (*Denial of Service*) mirati appunto ad impedire l'accesso ai server sovraccaricandoli.

  Altri attacchi di questo tipo sono quelli di tipo **ransomware**, dove si mira a crittografare i dati degli utenti, chiedendo riscatto per la decifrazione (in questo caso l'accesso che sottraiamo è quello ai dati stessi). Secondo l'Anastasi questo è un buon modo per fare soldi facili.

1.2.1 Un semplice modello

Riprendiamo i personaggi di Alice e Biagio, assunto che fra i due sia presente un canale di comunicazione non sicuro, e introduciamo un nuovo personaggio, *Gastone* (da *Intruder*), che ha come obiettivo quello di spiare e manomettere la conversazione fra Alice e Biagio.

Chiaramente, Alice e Biagio saranno un modello per due host che trasmettono informazioni sensibili, come client e server di un servizio Web di transazioni online, server DNS o router che si scambiano tabelle di routing, ecc...

Ci sono molte azioni dannose che Trudy potrebbe intraprendere sulla comunicazione fra Alice e Biagio, come ad esempio:

- **Sniffing** dei messaggi;
- **Inserimento** di nuovi messaggi sul canale;
- **Impersonazione** di Alice o Biagio, replicando il loro indirizzi IP;
- **Manomissione** della comunicazione fra Alice e Biagio, ad esempio rimuovendo uno dei due dal canale di comunicazione;
- **Denial of service**, cioè prevenire che il servizio venga usato (eliminando i messaggi sul canale o sovraccaricando le risorse di Alice o Biagio).

1.2.2 Introduzione alla crittografia

Il primo meccanismo di difesa che abbiamo a disposizione è la **crittografia**.

Definiamo un tipo di crittografia generale, basata su un dato *algoritmo crittografico*. Un algoritmo crittografico è un algoritmo che prende in ingresso una stringa comprensibile all'uomo (cioè in *chiaro*) e restituisce una stringa crittografata, cioè incomprensibile.

L'algoritmo crittografico prende come argomento una chiave, che usa per crittografare il messaggio in chiaro. Fissiamo l'algoritmo e diciamo che la sua applicazione al messaggio in chiaro m con chiave K si può semplicemente indicare come $K(m)$.

Tornando ad Alice e Biagio, diciamo che Alice possiede la chiave crittografica K_A e Biagio la chiave K_B. Vogliamo come condizione che per qualsiasi m:

$$m = K_B(K_A(m))$$

cioè K_B *decifra* i messaggi cifrati con K_A.

A questo punto quello che Alice invierà sulla linea sarà $K_A(m)$, e Biagio potrà successivamente usare la sua chiave K_B per ottenere il messaggio m secondo la stessa equazione di prima.

Diciamo anche che K_A è la chiave *pubblica*, cioè quella che può essere nota a tutti (e lo sarà ad Alice come a Trudy), in quanto viene usata per crittografare i messaggi destinati a Biagio. K_B sarà invece la chiave *privata*, cioè quella che solo Biagio deve avere, e che egli può usare per decifrare i messaggi rivolti a sé. Questo è sicuro in quanto, una volta crittografati, nessuno (né Alice ne Trudy) può decifrare i messaggi rivolti a Biagio, se non Biagio stesso.

1.2.3 Approcci alla decifratura

Vediamo quindi come Trudy, entrata in possesso di un messaggio crittografato da Alice per Biagio, potrebbe procedere alla decodifica del messaggio.

- Un primo approccio è chiaramente quello di ottenere in qualche modo la chiave di Biagio. Questo non è un approccio attuabile nella pratica (ci aspettiamo che Biagio custodisca attentamente la sua chiave), ma se non altro ci dimostra che non esistono approcci *"a prova di idiota"*, e cioè che non facendo attenzione con le chiavi si possono invalidare anche gli schemi crittografici più complessi;

- Un altro approccio è quello a **forza bruta**: si può provare a decrittografare raccogliendo a priori un insieme di chiavi possibili, ecc... il messaggio di Alice applicando l'algoritmo crittografico (che abbiamo detto è prefissato e noto) con tutte le chiavi possibili: questo però richiede grandi capacità computazionali e non è quasi mai conveniente;
- Un approccio più intelligente è quello di tipo statistico, cioè si può pensare di fare un'**analisi statistica** prima di applicare la forza bruta, magari raccogliendo a priori un insieme di chiavi possibili, ecc...