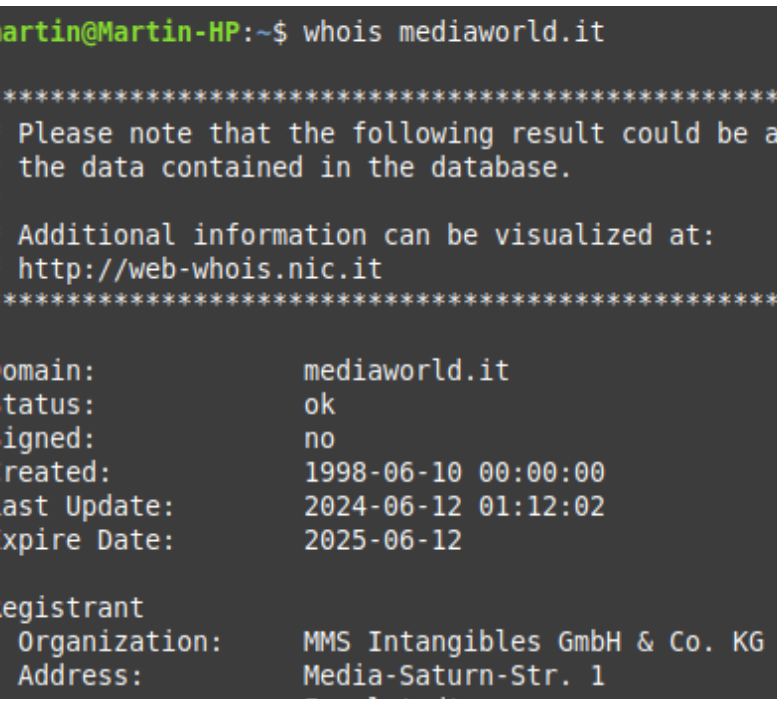
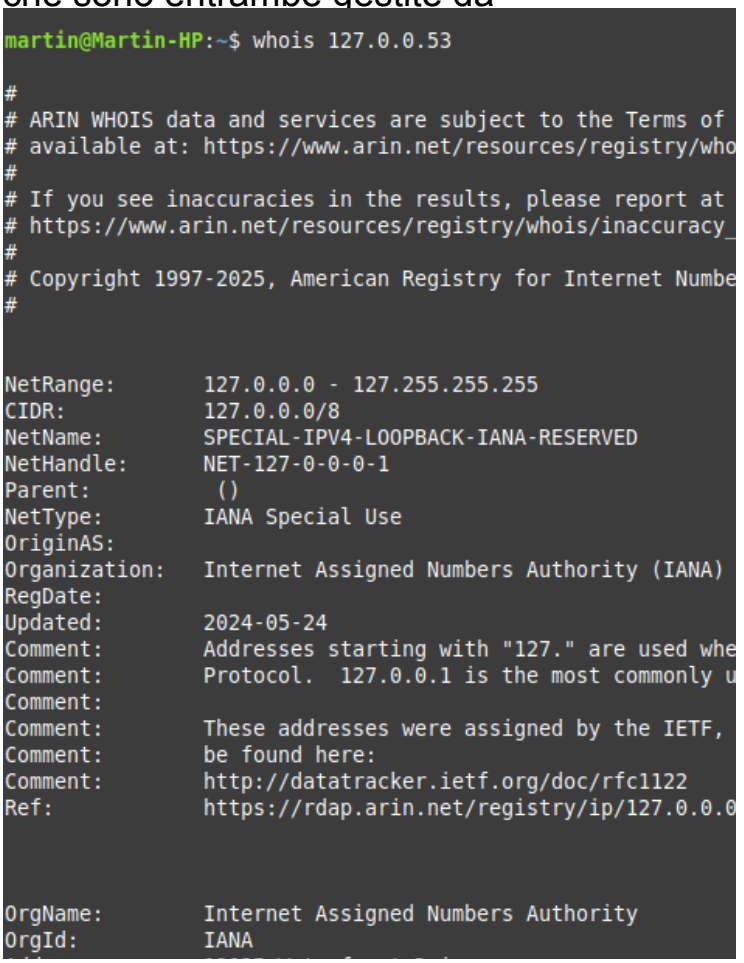
DOMANDE NOTE:

Wireshark:

- dimmi se il sito (es. di mediaworld) ha il server del proprio sito gestito da loro o da una terza azienda.
    a. RISPOSTA:
        - lancia nslookup www.mediaworld.it per scoprire l'ip del server;
        - lancia whois mediaworld.it per scoprire le persone e le aziende che gestiscono mediaworld;
        - lancia whois sull'IP ricavato dal punto 1 per vedere se coincide con quanto ottenuto dal punto 2 (una cosa è a chi è affidato il controllo del sito, un altro è chi effettivamente gestisce il lato IT del sistema);
        - Osserviamo dalle foto che sono entrambe gestite da IANA.





- perchè WSH ha bisogno di sudo?

    a. RISPOSTA: Perchè WSH deve ordinare alla scheda di rete, quindi interfacciandosi con l'hardware, di entrare in modalità promiscua. In questa modalità, la scheda "ascolta" e inoltra al sistema operativo tutti i pacchetti che vede, indipendentemente dal loro destinatario.
- cos'è il protocollo ARP?
    a. RISPOSTA: ARP (Address Resolution Protocol) è un protocollo che mappa gli indirizzi IP (logici) agli indirizzi MAC (fisici). Serve a un dispositivo per scoprire l'indirizzo MAC di un altro dispositivo che si trova sulla stessa rete locale. Il dispositivo invia un ARP Request (domanda broadcast: "Chi ha questo IP? Dammi il tuo MAC"); il destinatario con quell'IP risponde con un ARP Reply (invio del suo MAC). Il mittente salva la coppia IP-MAC nella sua cache ARP.
- Perché i contenuti della maggior parte dei pacchetti sono illeggibili? Come si può comprenderli?
    a. RISPOSTA: I payload non si possono leggere perché sono cifrati. No, non è possibile comprendere direttamente il contenuto cifrato di un messaggio catturato su Wireshark. È possibile decifrarlo e leggerlo solo se si possiede la chiave di decifratura (es. la chiave simmetrica di sessione per TLS, o la chiave pre-condivisa per WPA2), e la si fornisce a Wireshark che è in grado di eseguire la decifratura in automatico.
- Quali protocolli (es. HTTP, FTP) trasmettono dati in chiaro? Che implicazioni di sicurezza ha questo per le informazioni sensibili catturate con Wireshark?
    a. RISPOSTA: HTTP e FTP trasmettono in chiaro. Implicazione: credenziali e dati sensibili sono visibili a chi intercetta il traffico.
- Confronta la visibilità del contenuto dei dati catturati in una sessione FTP rispetto a una SSH. Perché questa differenza è cruciale dal punto di vista della sicurezza?
    1. RISPOSTA: FTP: dati visibili in chiaro. SSH: dati cifrati e illeggibili. Cruciale perché SSH garantisce riservatezza, FTP no.

- Come possono comandi come ping e traceroute (analizzabili via Wireshark) o nslookup essere utilizzati per la ricognizione di rete da parte di un attaccante? Quali informazioni potrebbero ottenere?
    - RISPOSTA: usati per ricognizione. Ottengono host attivi (ping), topologia/router (traceroute), mappatura IP-dominio (nslookup).

Interfaccia Socket:

1. provare a vedere scambio pacchetti su WireShark durante connessione client-server di questa esercitazione;
    a. RISPOSTA: mettere filtro ip.src e ip.dst == 127.0.0.1, ossia localhost;
2. dove avvengono nei codici l'inizializzazione delle connessioni? le send e le receve? le interfacce socket dove sono?
    a. RISPOSTA: in createTCPconnection o createUDPInterface vengono create le socket per le connessioni. Con TCP/UDPsend e TCP/UDPreceive facciamo il resto.
3. cos'è il socket?
    a. RISPOSTA: sono le interfacce che usano i vari processi, sia di client che di server, per creare un flusso di connessione endpoint. L'unione dell'indirizzo IP e della porta coinvolta costituiscono la socket;
4. criterio per scegliere TCP o UDP? Vantaggi e svantaggi?
    a. RISPOSTA: i TCP è come una spedizione raccomandata, garantisce affidabilità, ordine e controllo del flusso, ideale per dati critici come file e pagine web, ma con un certo overhead. Il UDP, invece, è più simile a un'emissione radio, veloce e leggero, perfetto per streaming o gaming dove la tempestività prevale sulla garanzia di ogni singolo pacchetto. La decisione finale si basa sulla priorità tra la certezza della consegna e la pura velocità.

WebServices:

1. cos'è HTTP e URL?
    a. hypertext transfer protocol e universal resource locator sono rispettivamente un linguaggio testuale di descrizione di una pagina web, composto da tag annidati, e una stringa che

permette di identificare in maniera univoca una risorsa http in qualsiasi parte del web;

2. cos'è CGI?
    a. common gateway interface, evoluluzione dell'internet classico basato su html statici. La risposta del server ora è dinamica in quanto può eseguire elaborazioni complesse, query a db e altro ancora dietro le quinte e senza che il client sappia niente (nasce il backend). Il server può quindi diventare a sua volta client per alcune richieste di dati (es. posta elettronica);
3. cosa sono web-socket?
    a. evoluzione successiva alla CGI, permette una connessione bidirezionale tra client e server. Nasce da una connessione http/https. E' persistente ed elimina la necessità di dover instaurare due connessione per una comunicazione bidirezionale, permettendo invece di comunicare da entrambe i lati nella stessa connessione. Flag di Protocol Upgrade.
4. cosa sono SOA?
    a. service oriented architecture. Prima le applicazioni giravano completamente sul client, ora con le SOA al client è delegato solo il 'client leggero' ossia GUI e funzionalità di base, mentre tutti i servizi restanti sono disponibili nel server come normalissime chiamate a funzione (API). Aumentano potenza di calcolo e distribuzione del lavoro, proprietà intellettuale programmi e algoritmi, semplifica manutenzione e rilascio aggiornamenti (client non deve scaricare nulla), introduce sistemi economici nuovi (pay per use). Tuttavia necessita connessione costante e stabile;
5. cosa sono API?
    a. interfaccia di funzione facente parte di una libreria messa a disposizione dal server remoto che offre il servizio effettivo. Sono tecnologicamente agnostici (no linguaggio fisso o architettura particolare necessaria). La funzione del client che usa l'API deve solo invocare la funzione in remoto e decodificare i dati (la fz è chiamata STUB). Anche la funzione del server che riceve la richiesta deve ipoteticamente codificare l'output della chiamata in una versione flessibile (la fz è chiamata SKELETON) e compatibile. La parte dei

programmi che effettivemente conducono codifica e decodifica si chiamano SERIALIZZAZIONE e DESERIALIZZAZIONE;

6. cosa sono Web-Services? Vantaggi svantaggi?
    a. i web-services sono una delle implementazioni più diffuse del sistema basato su SOA. Sfrutta sempre http/https per essere facilmente adattibile alle reti già esistenti (firewall ecc ecc). Aggiunge scalabilità orrizontale (ogni micro-servizio può essere assegnato a macchine diverse), facilità di manutenzione lavorando su singoli moduli, agnosticita tecnologica e totale indipendenza tra moduli, evitando cosi effetti collaterali di bug, crash o attacchi. Tuttavia aumenta carico complessivo rete e latenza, difficolta in monitaggio globale e gestione, così come eventuali join tra più db. Ha due approcci principali:
        i. XML SOAP;
        ii. REST: più diffusa, trasferisce i dati sfruttando le funzionalità di http come i suoi metodi (GET, POST, PUT, DELETE) e usando formati semplici per il parsing come json;
7. cos'è il load balancer? che tipi esistono?
    a. direttore d'orchestra del traffico di rete. Distribuisce omogeneamente il traffico delle richieste, allocando o disallocando le risorse quando necessario. Porta a più richieste soddisfatte contemporaneamente, più rapidità, non manda in crash nessuno e se succede non invia più richieste a quel specifico server. Può essere implementato con 'servizi distinti su istanze diverse', ossia base forse nemmeno considerabile balancer, 'balancer centrale', la più diffusa, ossia quando il balancer possiede l'IP/dominio del punto di accesso al server, e poi redistribuisce lui la richiesta ai vari sotto-server in base al servizio richiesto e ai server possibili (può essere configurato per gestire anche le session stickiness, ossia che le richieste di un certo client vanno mandate allo stesso server per migliorare alcuni servizi (in particolare legate alle sessione utenti), 'balancing legato a DNS': DNS risponde con IP diversi ogni volta che viene richiesto, distribuendo il carico omogeneamente (susciettibile a memoria cache DNS).

8. cos'è cloud computing e quali sono le principali funzionalità?
    a. paghi per le risorse che ti servono. E' un servizio che si basa su una connessione costante, possibilità di virtualizzazione e sistema web-service:
        i. ONSITE: hai solo hw;
        ii. IaaS (infastructure as a Service): hai solo VM;
        iii. PaaS (Platform): VM con SO;
        iv. SaaS (Software): direttametne app/software pronti da usare (gmail, dropbox, ecc);
        v. FaaS (function): detto serverless (fuorviante), consente agli sviluppatori di scrivere ed eseguire piccole funzioni/codice in risposta a specifici eventi senza considerare nulla di tutto quello che sta sotto. Le funzione sono tipicamente event-driven e vengono pagate solo per il tempo di esecuzione effettiva (granularità al massimo).
9. viene richiesto di eseguire l'ultimo esercizio. sfruttare le architetture dei microservizi. Che problema esce, se c'è? calcolare il tempo di esecuzione (ottimizzato e non) e spiegare un po' tutto (VEDI ESERCITAZIONE);
10. (sempre relativo ad esercitazione, in particolare chat Web) cos'è SOCKET.io:
    a. libreria js che implementa pooling aggiuntivo nel caso in cui la connessione web-service dovesse cadere.

WebSocket:

1. eseguire tutti gli esercizi; -> vedi esercitazione;
2. come è fatta la web chat degli esercizi? -> vedi esercitazione;
3. cos'è una web socket? -> risposto prima;
4. cos'è/cosa fa HTTP UPGRADE?
    a. passa da HTTP/HTTPS a Web-Socket;
5. a che livello opera websocket? -> livello applciazione;
6. che funzione websocket non si può fare con HTTP? -> cpmunicazione bidirezionale, per esempio il server che invia per primo una richiesta al client (andrebbe instaurata altra connessione, vedere esempio coockies di ripresa connessione se chiudi server ma non client);

7. da chi parte la richiesta? -> sempre client perchè nasce da http;
8. cosa permette di fare il websocket? -> già risposto;
9. Domande sulla chat websocket:
    a. com'è fatta, lato client e lato server?
        i. spiegare codice e teoria web-socket immagino;
    b. perchè viene usato nodejs?
        i. per le librerie di facile importazione e adattabilità con JS, come socket.io o express;
    c. perchè è stata scritta in js e non in java o c?
        i. principalmente per la gestione semplice di funzioni event-driven;
    d. come lo scriveresti in java o c?
        i. togliendo javascript dal server (dal client non puoi perchè solo js è compatibile col browser), toglieresti anche quindi nodejs. Questo comporterebbe la perdita di tutte le librerie, per cui ne andrebbero trovate altre altrettanto valide per le connessioni di rete web-socket. In piu perdi la gestione della concorrenza e dell'inattività di nodejs, per cui dovresti farla tu da 0. Addirittura con C siamo al massimo di complessità perchè dovremmo guardare tutte queste funzionalità considerando un estermo basso livello di software, quindi tornare a lavorare con le socket e i vari componenti di rete come nell'esercitazione 2. Chiaramente in generale sono molto meno supportate per questo tipo di funzionalità;
    e. storia cifratura simmetrica?
        i. la cifratura simmetrica è la prima tipologia di sistema di crittografia. La sua peculiarità, in particolare che la distingue dal più avanzato modello a cifratura assimetrica, è che la chiave di cifratura e quella di decifratura sono uguali. Per questo chiaramente con l'avanzare del progresso tecnologico è stata inventata la cifratura assimetrica, ossia perchè bisognava passare durante l'avvio della comunicazione la chiave con cui sono stati cifrati i messaggi al destinatario, in modo che possa leggerli una volta decifrati i contenuti; questo porta chiaramente con se dei problemi di sicurezza, in

quanto bisogna rendere sicuro il canale di comunicazione della chiave altrimenti chiunque può decifrare il contenuto. Tra i primi algoritmi di cifratura simmetrica troviamo il cifrario di cesare (lettera con salto di n lettere), il cifrario monoalfabetico (l'alfabeto viene modificato senza uno schema preciso, ma è fisso, per cui si possono capire alcune combinazioni con l'analisi in frequenza, tuttavia possiede 21! (con lettere straniere 26!))...+ lettere aggiungiamo (caratteri speciali, maiuscole) più aumenta difficoltà di brute force o analisi frequenziale; per ultima la cifratura a blocchi che divide il testo in tot blocchi di dimensione fissa e li cripta tutti in modi diversi con permutazioni, XOR e op. logico/matematiche ecc ecc, poi rimette insieme e dà l'output. Per decriptare si fa processo inverso paro paro. Più famosa implementazione è AES con blocchi da 128 bit.

Modello PUB/SUB:

1. cos'è PUB e SUB? Perché si usano? Vantaggi e svantaggi?
    a. modello di messaggistica alternativo a client/server con PUB/SUB/BROKER/TOPIC. Si differenzia principalmente perchè client/server è uno a molti, PUB/SUB è molti a molti; broker fa anche da balancer. Hai disaccoppiamento temporale (tempo invio != tempo ricezione), spaziale (PUB e SUB non si conoscono), agnosticità tecnologica completa, scalabilità enorme (puoi aggiungere quanti PUB e SUB vuoi) + affidabilità flessibile sulla sicura consegna della pubblicazione in topic (QoS). Svantaggi sono maggiore complessità del sistema, difficoltà di debugging se qualcosa non va e maggiore latenza perché introdotto componente intermedio. Protocollo più diffuso è MQTT implementato per come lo abbiamo usato da Mosquito;
2. cos'è e cosa fa il broker?
    a. componente intermedio che gestisce le pubblicazioni dei publisher, le sottoscrizioni dei subscriber e il corretto flusso di aggiornamento tra pubblicazioni/ricezioni nei vari topic;

3. quali sono i wizard pattern? -> # e +;
4. Domande sull'esercitazione:
    a. come funziona, come è fatto? -> già risposto;
    b. quale il protocollo di riferimento? -> MQTT;
    c. questione dell'affidabilità. Come è garantita?
        i. attraverso il QoS, ossia il quality of Service (campo presente e impostabile nelle sottoscrizioni). Se 0 significa che non ci interessa avere ACK dal broker di avvenuta ricezione, 1 se si usa (per il publisher questo). Per il subscriber invece, 0 significa che non ci sono conferme di ricezione da parte del ricevente (come per pub), 1 che si deve ricevere il messaggio almeno una volta, 2 esattamente una volta.
    d. perché sono stati introdotti i QoS sebbene esistesse già TCP?
        i. semplicemente perchè sono di due livelli diversi, applicazione e trasporto. TCP garantisce l'arrivo dei PDU a livello di trasporto, ma non ne garantisce l'interpretazione corretta semantica dal punti di vista applicativo. Mettiamo che via sia un crash del broker prima di interpretare il messaggio di un publisher, ma dopo aver ricevuto effettivamente tutti i PDU che lo compongono, oppure che la connessione TCP venga interrotta da parte di uno degli host, mentre invece MQTT rimane persistente (temporalmente disaccoppiato) TCP non può garantire che il 'messaggio' (inteso proprio come a livello applicazione MQTT) arrivi tutto a posto. QoS si occupa di verificare, coerentemente con il proprio livello di dominio, il successo della communicazione a livello applicazione.
    e. storia cifratura assimetrica?
        i. nasce come evoluzione di quella simmetrica. Ogni utente possiede due chiavi, una pubblica e una privata (correlate matematicamente). Qui il dato viene cifrato con la chiave pubblica del destinatario (appunto ottenibile perché pubblica), che quindi potrà decifrarlo solo con la propria chiave privata (nemmeno il mittente potrebbe decifrarlo una volta ciftrato). Non vi è più quindi

uno scambio di chiavi, e la stessa chiave pubblica può essere usata da più utenti. Il più importante è RSA, che si basa sulla difficoltà di scomporre un numero in fattori primi.

f. casa domotica:
    i. PUB sono i sensori della casa;
    ii. SUB sono le app mobile per ricevere aggiornamenti e attuatori della casa;
    iii. esempio: sensore luce rileva che il livello di lumen nella stanza padronale (registrata come stanza1) è sceso sotto tot perchè il sole sta calando allora pubblica tramite broker il dato LUCE_UNDER_LEVEL; a questo punto il controllore della lampadina della stanza, iscritta al topic stanza1 riceve il messaggio LUCE_UNDER_LEVEL, e di conseguenza accende la luce della lampadina;

Sicurezza:

1. come si fa a fare riservatezza con un messaggio di posta elettronica;
    a. si cifrano i messaggi con una cifratura a chiave assimmetrica, con un algoritmo come AES;
2. come si fa a fare firma digitale con un messaggio di posta elettronica;
    a. Si calcola l'hash univoco del messaggio. Questo hash viene cifrato con la chiave privata del mittente. La firma cifrata viene allegata al messaggio. Il destinatario può poi verificare la firma usando la chiave pubblica del mittente per decifrare l'hash e confrontarlo con un hash ricalcolato del messaggio ricevuto. Questo garantisce autenticità e integrità;
    b. oggi giorno si usa protocollo PGP che unisce quanto descritto nelle ultime due domande!
3. Come avviene la registrazione sicura di un utente?
    a. l'utente fornisce un nome utente (o email) e una password. La password non viene mai salvata in chiaro. Viene aggiunta una stringa casuale unica (salt) alla password, e poi l'intera stringa (password + salt) viene sottoposta a una funzione di hashing

crittografico forte. Il risultato (l'hash) e il salt vengono salvati nel database. Viene inviato un link di verifica (con un token monouso) all'email/numero fornito. L'utente deve cliccare il link per attivare l'account, confermando la proprietà dell'indirizzo. L'hash della password e il salt vengono archiviati in un database sicuro, con adeguate misure di protezione (crittografia del database, controlli di accesso). Tutte le comunicazioni durante il processo di registrazione (es. invio di password o token) avvengono tramite connessioni cifrate (HTTPS/TLS);

4. Backend/Frontend dove avviene lo store e la cifratura (come?) delle psw?
    a. lo store e la cifratura (hashing) delle password avvengono sempre sul Backend. Il frontend invia la password in chiaro al backend (sempre via connessione cifrata, es. HTTPS). Il backend riceve la password in chiaro, le aggiunge un salt e la sottopone a una funzione di hashing crittografico forte (es. bcrypt, Argon2). L'hash risultante e il salt vengono poi memorizzati nel database del backend;
5. l’app di web-chat websocket, che tipo di sicurezza usa?
    a. nessuna!
6. l’esercizio non è sicuro, come potremmo fare per renderlo tale?
    a. TLS, robustezza sugli input e i dati, richiesta autenticazione utente;
7. che proprietà possiede il TLS?
    a. possiede confidenzialità, tramite la crittografia simmetrica (dopo un handshake iniziale che stabilisce una chiave segreta condivisa, tutti i dati vengono cifrati e decifrati usando questa chiave), integrità, tramite l'uso di codici di autenticazione dei messaggi (MAC) (un'impronta crittografica viene calcolata e inviata con i dati; se i dati vengono alterati, l'impronta non corrisponde) e autenticazione, tramite certificati digitali (basati su crittografia asimmetrica) e una Certificate Authority (CA) (il server presenta un certificato che viene verificato dal client per confermare l'identità e la validità, garantendo che si stia parlando con il server legittimo;
8. come fa a scambiarsi la chiave simmetrica?

    a. usano le chiavi assimetriche (modello publiche private spiegato prima) per scambiarsi una chiave simmetrica (quindi risolve il problema delle chiavi simmetriche), per poi usarla per tutto il resto della connessione;
9. come posso rendere MQTT più sicuro con poco sforzo?
    a. togli --insecure per i motivi spiegati nell'esercitazione e prima; controlla ci sia ACL nel broker.
10. problema che TLS non può risolvere?
    a. non protegge da attacchi DoS o MitM (injection), ossia se il contenuto stesso è malevolo). Inoltre non controlla la proprietà di autorizzazione, disposta invece al destinatario; PLUS che non ho saputo dire per la lode ---> Problema TLS relativo specificatamente al modello PUB/SUB: se il broker viene attaccato durante la decodifica e l'elaborazione di un messaggio, il sistema è vulnerabile perchè i dati sono scoperti. TLS non protegge infatti da questo, si occupa solo di far arrivare criptati i dati (+firma), ma non controlla cosa succede una volta arrivati.