Telematica II Parziale
08 World Wide Web (WWW) Concetti generali
World Wide Web (WWW) Concetti generali
World Wide Web (WWW)
Complesso di documenti ipertestuali risiedenti su vari host e collegati fra loro mediante hyperlink. Gli utenti navigano attraverso questi documenti utilizzando un browser Web. Oltre al testo, i documenti possono contenere immagini, video e altre risorse di tipo multimediale. Il World Wide Web fu iniziato al CERN di Ginevra nel 1989 da Tim Berners-Lee, attualmente è il direttore generale del World Wide Web Consortium (W3C).
Testi e ipertesti
- Ipertesto: modello di interfaccia utente che permette al fruitore di un documento di «navigare dietro richiesta» mediante un PC (generalmente)
- Tail anchor: origine dell'hyperlink all'interno di un documento
- Destination anchor: destinazione dell'hyperlink (tipicamente un documento)
- Di norma: la tail anchor viene associata ad una zona del documento origine che ha attinenza logica con la destinazione
- Un testo si legge (sequenzialmente), un ipertesto si naviga
- Il documento di origine e quello di destinazione tipicamente risiedono su host diversi: Occorre che tra i due host sia attivo un protocollo di comunicazione capace di trasferire il documento di destinazione sull'host A. Trasferimento dietro richiesta dell'utente: Host A client, Host B server;
- Nota: il documento di destinazione non è necessariamente un ipertesto • Potrebbe non avere link o potrebbe non essere nemmeno un testo (es. un'immagine)

Standard per gli ipertesti
- HyperText Markup Language (HTML): Standardizzato dal W3C, è il più diffuso standard per la definizione del formato degli ipertesti. Basato su «plain text" con inserimento di tag. Linguaggio di markup: unisce il testo «utile» e l'informazione per la gestione dello stesso inframezzandoli in un unico stream. Altri esempi di linguaggi di markup: RTF, LaTeX. Permette di definire gli hyperlink
- HyperText Transfer Protocol (HTTP): Definito dall'IETF in una serie di RFC. Principale protocollo applicativo del Web. Usato dal client per instaurare una connessione con il server ed accedere ad un documento residente su un «Information Server»
- World Wide Web (WWW): insieme degli Information Server

Web: tipi di Information Server
Più tipi di servizi nel Web: www (protocollo HTTP, HTTPS), ftp (protocollo FTP), nntp (protocollo NNTP), coap (protocollo CoAP), mailto (servizio di posta elettronica), …. Ogni servizio servizio richiede il proprio protocollo. Il Web è un ambiente multiprotocollo. Un browser (Internet Explorer, Microsoft Edge, Mozilla Firefoz, Google Chrome, Apple Safari, ecc.) è un applicativo che in genere contiene molteplici client (client HTTP, client FTP, ecc.)
Indirizzamento nel Web URI, URL e URN
Indirizzamento: URI, URL e URN
Un servizio fornisce l'accesso ad una risorsa: Uniform Resource Identifier (URI), Uniform Resource Locator (URL): identifica dove si trovi la risorsa, Uniform Resource Name (URN): identifica la risorsa in base ad un identificatore univoco e persistente (anche in caso di indisponibilità futura). Il termine URI è meno restrittivo, usato nelle specifiche (RFC). Nella pratica: l'URL è il tipo di URI utilizzato più comunemente. L'URL è la stringa che il browser utilizza per localizzare una risorsa nel Web
Sintassi
- scheme:hierarchical-part -> scheme: identifica il metodo d'accesso (protocollo), Stringa alfabetica in formato minuscolo, Informa il client su come accedere alla risorsa specificata dalla parte hierarchical-part;
- hierarchical-part: Struttura generale sostanzialmente identica per tutti gli schemi, con qualche particolarità specifica.

URI – Codifica
Solo caratteri ASCII grafici (no control chars). Alcuni caratteri (delimiter) sono riservati / . # ? ; : $ … (altri). SPACE e caratteri riservati. Codifica con sequenza di escape «%hex-code». Es.: a casa à a%20casa. RFC3986 classifica i caratteri secondo una suddivisione ben definita:
- unreserved = ALPHA / DIGIT / "-" / "." / "_" / "~"
- gen-delims = ":" / "/" / "?" / "#" / "[" / "]" / "@"
- sub-delims = "!" / "$" / "&" / "'" / "(" / ")» / "*" / "+" / "," / ";" / "="

Caratteri unreserved: servono per comporre i nomi dei campi. Caratteri delimiter: servono per delimitarli

URL con schema file

file://host/path : host è un domain name (o un indirizzo). La risorsa è accessibile tramite file system. In questo caso il campo scheme non indica un protocollo. Il campo path segue le convenzioni del file system corrente (Windows, Unix, ecc.), mentre la sintassi è quella tipica di Unix. I nomi delle directory sono separati dal carattere «/» (slash) e non dal carattere «\» (backslash), come in Windows.

URL con schema http

http://host:port/path : host è un domain name oppure un indirizzo IP (IPv4 o IPv6). port identifica la porta su cui è in ascolto il server http. Porta di default del protocollo HTTP: TCP 80. path corrisponde ad un http selector che identifica una risorsa (di solito un file) nell'albero delle directory. La radice di tale albero non è la radice del file system sul server HTTP, ma una directory a valle di tale radice denominata Document Root. Il server HTTP non ha accesso a directory esterne alla Document Root. La Document Root ha come path "/" e può essere omessa. Il campo scheme «http» non può essere omesso, anche se sono comuni indirizzi Web in cui sia omesso. In questi casi provvede il browser Web a completare l'URL. Tale risorsa è il file di nome index.html che risiede nella Document Root. Il file index.html è definito come Default Document sul server e, in quanto tale, non ha bisogno di essere specificato nell'URL.

URL con schema ftp

ftp://user:password@host:port/path : user: nome utente, Valore di default: anonymous. port: porta del server ftp, Valore di default: 21. path: percorso della directory a cui si vuole puntare, Navigato tramite il comando CWD.
I quattro URL equivalenti dell'esempio identificano la risorsa "Elenco della Document Root". A tale risorsa si accede in modo anonimo. Come per HTTP: la Document Root ha come path «/» e può essere omessa. Nota: il carattere «@» facente parte della password deve essere codificato tramite la sequenza di escape «%40» per non essere interpretato come separatore fra password e host.

Query String

Valida con il metodo HTTP, è una sequenza di caratteri appesa all'URL. Valida anche con il protocollo CoAP. Scopo: trasferire informazioni dal client al server. Informazioni tipicamente ottenute mediante un modulo (form HTML) che il server ha presentato all'utente e in cui l'utente ha inserito dei valori. Ogni valore è associato ad un parametro. Le coppie valore/parametro sono assemblate per comporre la query string. Le coppie sono comunemente separate dal carattere "&". Tuttavia esisteva una raccomandazione del W3C per sostituire tale carattere con ";", Raccomandazione mai adottata. Non esiste un limite specifico al numero o alle dimensioni dei campi, alla lunghezza della query string, ed alla lunghezza totale dell'URL. Unico limite: capacità dell'host che deve elaborare l'URL.

Fragment

Un hyperlink può avere come destination anchor non l'inizio di un documento HTML, ma un suo punto intermedio. In tal caso, all'URL che indirizza all'interno del documento si appende una sequenza chiamata fragment identifier. Nell'esempio, la risorsa a cui accede l'URL è il "Default Document della directory apidocs del server www.unipr.it". In entrambi i casi il client (browser) acquisisce tale risorsa. Primo URL: il browser si posiziona all'inizio del testo. Secondo URL: il browser si posiziona nel punto intermedio identificato (nel codice HTML) dal tag. Nel caso in cui tale tag mancasse, il browser si posizionerebbe all'inizio del documento HTML.

URL relativo e URL assoluto

URL assoluto: URL in cui sono presenti i campi scheme e host. Necessario per identificare in modo univoco una risorsa nel Web. URL relativo: URL in cui mancano i campi scheme e host. Nel caso in cui dal campo path dell'URL relativo siano rimosse alcune componenti più significativi (partendo da sinistra), si ha ancora un URL relativo. Concatenando scheme, host e la parte di path rimossa con l'URL relativo è possibile ricostruire l'URL assoluto ed identificare così una una risorsa. Base URL: URL assoluto posto davanti all'URL relativo. In un documento HTML che contiene link relativi ad altri documenti, la base URL può essere dichiarata esplicitamente. Altrimenti si usa come base URL l'URL della encapsulating resource (ossia del documento stesso). L'utilizzo di URL relativi in documenti HTML è utile perché consente la portabilità dei documenti stessi. Consente di trasferire un intero sito Web da un host ad un altro senza dover modificare tutti i link interni. Gli URL assoluti cambiano, mentre gli URL relativi rimangono immutati.

Esempi di scheme URL

La procedura per registrare nuovi schemi è specificata nella RFC4395. Il registro degli schemi è reperibile all'URL.

09 HyperText Transfer Protocol (HTTP)

HyperText Transfer Protocol (HTTP) Concetti generali

HTTP – Concetti fondamentali

HTTP si basa su un protocollo di trasporto di tipo «connection oriented» (TCP). Il server rimane in attesa di richieste sulla well-known port 80 (porta di default). Il client effettua una richiesta TCP e richiede una risorsa (tipicamente un file HTML). Il server trasmette il file e svincola la connessione (chiude la connessione TCP).

HTTP – Struttura dell'interazione

HTTP si basa su un paradigma di tipo request/response. NON esiste il concetto di connessione http. Non c'è una fase di inizializzazione. Ogni richiesta è indipendente dalle precedenti. HTTP è stateless. Una connessione TCP per ogni elemento. Se un ipertesto include delle immagini, per ogni immagine viene istanziata una connessione TCP. Il dialogo avviene in modalità «text-oriented». Compatibile con TELNET: telnet host 80.

HTTP – Transazione di base

L'URL contenuto nel messaggio di richiesta GET può essere assoluto o relativo. Nel caso in cui sia relativo, come base URL si utilizza la stringa http://hostname, dove hostname indica il domain name o l'indirizzo IP su cui è in esecuzione server http. Appena restituita la risorsa, il server HTTP chiude la connessione. Nel caso in cui la richiesta non possa essere soddisfatta, il server restituisce sempre qualcosa. Pagina di errore che possa essere mostrata (rendering) da parte del client (browser o applicativo CLI).

Esempio di pagina di errore

In caso di errore, il server genera a run-time (in base alla richiesta) una pagina HTML dinamica come risposta, che illustra discorsivamente il problema. Il contenuto di questa pagina non è definito dal protocollo http. Può essere differente da server a server, anche se il suo significato è sempre il medesimo.

HTTP – Versioni del protocollo

Ogni versione è compatibile con le precedenti. Ogni richiesta HTTP (tranne con la v0.9) deve dichiarare che versione del protocollo sta utilizzando. GET URL (richiesta in formato 0.9). GET URL HTTP/1.0 (richiesta in formato 1.0) • GET URL HTTP/1.1 (richiesta in formato 1.1).

HTTP/1.0

In ogni richiesta viene inserita la versione del protocollo HTTP utilizzata. La risposta contiene uno status code, che consente al client di comprendere il successo o il fallimento della richiesta e di adattare il proprio comportamento. È stato introdotto il concetto di header HTTP sia per le richieste che per le risposte, consentendo il trasferimento di metadati, e di fatto rendendo il protocollo HTTP flessibile ed estensibile. L'introduzione degli header HTTP consente il trasferimento di altri tipi di documenti, in aggiunta a file HTML plain. Header Content-Type. La presenza di elementi da caricare a run-time forza il client ad effettuare ulteriori connessioni per il recupero di tali elementi.

HTTP – HTTP/1.1

Consente di riutilizzare una connessione, in modo da risparmiare sul tempo richiesto per l'apertura di una nuova connessione (specialmente per il caricamento di elementi inclusi nel documento relativo alla richiesta originale). Introdotto il concetto di pipelining, che consente di eseguire una seconda richiesta prima che la risposta alla prima richiesta sia stata completamente trasmessa. Riduzione della latenza della comunicazione. Introdotti meccanismi di cache control e di content negotiation (inclusi lingua, encoding), che consentono al client ed al server di accordarsi sul formato più adeguato da scambiarsi. Header host: consente la comunicazione con server che ospitano molteplici domini con lo stesso indirizzo IP.

HTTP – HTTP/2.0

Protocollo binario, non testuale. Non può più essere letto e creato manualmente, ma consente di implementare tecniche di ottimizzazione. Protocollo multiplexed, consente di istanziare richieste parallele sulla medesima connessione. Prevede header compressi, con particolare vantaggio sulla rimozione di duplicati ed overhead dei dati trasmessi (spesso gli header sono simili su un set di richieste conseguenti). Consente al server di popolare la cache del client con i propri dati, attraverso un meccanismo definito «server push». Non richiede un adattamento particolare lato server, siccome l'utilizzo di HTTP/1.1 o HTTP/2.0 è abbastanza trasparente.

Struttura dei messaggi HTTP/1.0-1.1

Richiesta: Request Line, [Header], Linea vuota (CRLF), [Entity Body]. Risposta: Status Line, [Header], Linea vuota (CRLF), [Entity Body]. Request Line/Status Line è sempre presente. Gli header possono essere assenti (tranne l'header Host in HTTP/1.1). La linea vuota deve sempre essere presente (separa l'header dal body). Entity Body può essere assente.

HTTP – Codici di stato

Il codice comunemente ottenuto in caso di accesso regolare a una risorsa è 200. Se la risorsa è stata spostata, il codice di stato è 301 o 302. L'header della risposta dovrebbe contenere il nuovo URL ed il client (browser) si collega al nuovo URL à Operazione «trasparente» per l'utente. Se la risorsa non esiste il codice è 404. Se la risorsa esiste ma è protetta il codice è 401. Alla ricezione di questo codice, un browser cerca di ricollegarsi alla risorsa presentando delle credenziali (username e password). All'utente viene presentata una finestra con la richiesta di username e password. Se la risorsa esiste ma è vietata il codice è 403. In genere questo codice viene restituito nel caso in cui la risorsa corrisponda al listing di una directory (ossia l'URL identifica una directory ed in questa directory non esiste un default document) ed il server sia configurato per non rendere disponibile l'elenco dei file di una directory. Solitamente i server Web sono configurati in questo modo.

HTTP – Header generali

«Informational headers»: Connection:, Date:, MIME-Version:, Transfer-Encoding:, Trailer: (usato per chunked encoding), Upgrade: nuovo protocollo, Via: lista intermediari.
«Caching headers»: Cache:, Pragma: (Pragma: no-cache).

- MIME-Version: utilizzato da alcuni server che costruiscono risposte MIME-compliant. Non previsto dalla specifica RFC2616
- Chunked encoding: metodo di transfer encoding in cui il body viene spezzato in blocchi («chunk») ognuno dei quali preceduto da un suo header che ne specifica la lunghezza
- Trailer: identifica il blocco finale e permette al ricevente di controllare la corretta ricezione dell'intera risorsa •
- Upgrade: utilizzato per verificare se la controparte supporta una nuova versione
- Via: analogo al campo Received di RFC2822, serve per tracciare il percorso della richiesta
- Pragma: no-cache: utilizzato da un client quando l'utente ricarica una pagina. Il client richiede ad un proxy intermedio di non ottenere la copia della risorsa in cache, ma di richiederla nuovamente all'origin server. La direttiva Pragma è concepita per un uso più generico, ma solitamente è usata solo in questa occasione.

HTTP – Header nei messaggi di richiesta

«Informational»: Client-IP:, From:, Host:, Referer:, User-Agent:, UA-Color:, UA-Disp: dimensioni display, UA-...
«Accept headers»: Accept: media types, Accept-Charset:, Accept-Encoding:, Accept-Language:, TE: accepted transfer encoding

- Host: informa il server su quali siano l'hostname e la porta a cui il client sta facendo la richiesta. Obbligatorio in HTTP/1.1
- Accept-Charset: informa il server su quali siano i charset accettati dal client (utile nel caso in cui il server abbia più copie di un documento, ognuna con caratteri differenti)
- Accept-Encoding: informa il server sulla capacità del client di accettare risorse codificate (es. in formato gzip)
- Accept-Language: utilizzato dal client per specificare una o più lingue (con un ordine di preferenza) tra cui il server può scegliere per restituire una risposta
- TE: Transfer Encoding, simile all'header Accept-Encoding, ma riguarda l'encoding di trasferimento • Se il parametro è vuoto, l'unico transfer encoding accettato è «chunked»

«Conditional request»: Expect: expected server behavior. If-Match: Entity Tag. If-Modified-Since:. If-Unmodified-Since:. If-None-Match: Entity Tags. If-Range: (richiesta condiz. per range di documenti). Range: (richiede un chuck di risorsa)
«Security»: Authorization:, Cookie:, Cookie2: (dichiara versione supportata).
«Proxy request»: Max-Forwards: (usato solo col metodo TRACE, limita il numero massimo di sistemi intermedi attraversati), Proxy-Authorization: (per autenticarsi ad un proxy), Proxy-Connection: (come il campo Connection)

HTTP – Header nei messaggi di risposta

«Informational»: Age: età della risorsa, Public: (request methods - obsoleto), Retry-After:, Server:, Warning:.
«Negotiation»: Accept-Ranges:, Vary: (lista headers che possono essere coinvolti nella scelta della risposta da dare al client)
«Security»: WWW-Authenticate:, Proxy-Authenticate:, Set-Cookie, Set-Cookie2:.

- Age: età stimata in secondi • Obbligatorio per cache HTTP/1.1
- Public: elenca i metodi supportati dal server
- Retry-After: utilizzato con il codice 503 (Service Unavailable), specifica una data o un numero di secondi
- Warning: fornisce alcune notizie su ciò che è accaduto durante la richiesta • HTTP/1.1 definisce vari codici di warning (3 cifre)
- Accept-Ranges: informa il client sul fatto che il server HTTP accetti o meno di trasferire un chunk di risorsa.

HTTP – Header nei messaggi di risposta – Entity Headers

«Informational»: Allow: (request methods ammissibili), Location:
«Content»: Content-Base: (URL di base), Content-Encoding:, Content-Language:, Content-Length:, Content-MD5:, Content-Range:, Content-Type:
«Caching»: Etag:, Expires:, Last-Modified:.

- Content-Base: specifica un URL assoluto da utilizzare come base URL per gli URL relativi contenuti nella risorsa
- Content-Length: specifica la lunghezza del body in byte • Se è presente l'header Transfer-Encoding significa che il file originario è stato sottoposto a codifica • à La lunghezza originaria (entity length) e la lunghezza trasferita nel messaggio sono diverse • à Content-Length non deve esserci; nel caso in cui sia presente, deve essere ignorato

- Content-Range: utilizzato nelle «range request» quando si chiede solo un chunk di un documento (es. dopo un'interruzione); identificato dove si trovi range trasmesso rispetto all'intero documento
- Content-Type: ha la medesima funzione che ha in un messaggio di posta MIME
- Etag: ha la medesima funzione di Last-Modified ma è più potente (risoluzione inferiore al secondo) • Il server può decidere di usare degli Etag "deboli" (iniziano con la lettera W), in tal caso modifiche cosmetiche al documento non cambiano l'Etag e il documento può essere ricaricato solo quando ci siano modifiche semantiche

Metodi HTTP

HTTP – Messaggio di Richiesta

HTTP – Metodo GET

Serve per acquisire una risorsa. GET condizionale: richiede al server HTTP una risorsa solamente se questa è aggiornata. Nel caso in cui l'URL identifichi un file normale: la risorsa è il file (caso comune). Nel caso in cui l'URL identifichi un file eseguibile (opportunamente configurato): il file è mandato in esecuzione e la risorsa è l'output del processo (form con metodo GET).

HTTP – Metodo HEAD

Agisce come il metodo GET, con la differenza che il server restituisce solamente l'header come risposta. L'header deve essere uguale a quello restituito dal metodo GET. Consente di ispezionare una risorsa senza doverla scaricare • Determinare il tipo (o la data, o altro). Determinare se la risorsa esiste. Verificare se la risorsa è stata aggiornata. GET e HEAD sono gli unici metodi obbligatori in HTTP/1.1.

HTTP – Metodo POST

Serve per mandare dati in input al server. L'URL specifica la risorsa che deve elaborare i dati. I dati sono inseriti nel body della richiesta. Solitamente serve per mandare i valori ottenuti dai campi di un modulo (form). In base al valore del campo Content-Type, il body può contenere i dati in formato query string, oppure come campi separati in formato form-data

HTTP – Metodo PUT

Serve per memorizzare dati sul server HTTP di destinazione (upload). L'URL specifica la risorsa (file) che si vuole generare/sostituire sul server http. Il file trasmesso è il body del messaggio. La lunghezza del body è quella «visibile» inclusiva dei terminatori di fine linea. Nell'esempio c'è solamente un fine-linea ed è il solo carattere LF (ammissibile in questo caso, siccome si tratta di un file generico, non interpretato dal protocollo ma trasferito as-is). Nota: differenza tra il metodo PUT ed il metodo POST: PUT: il body del messaggio di richiesta è un file che viene depositato sul server, l'URL identifica tale file e POST: il body del messaggio di richiesta è un input che viene passato al server, l'URL identifica il file che elabora la richiesta.

HTTP – Metodo DELETE

Serve per rimuovere dati dal server. L'URL specifica la risorsa che si vuole cancellare. Anche nel caso in cui il server dia risposta affermativa, potrebbe non effettuare l'operazione (HTTP/1.1). PUT e DELETE dovrebbero richiedere autenticazione.

HTTP – Metodo TRACE

Serve per verificare come una richiesta arrivi al server HTTP, effettuando un loopback. Eventuali sistemi intermedi possono modificare la richiesta originale. Spesso inseriscono degli header aggiuntivi. A volte modificano l'URL. Quando il server HTTP di destinazione riceve una richiesta TRACE. Manda un messaggio di risposta con codice 200. Mette nel body del messaggio il messaggio ricevuto. Utilizzato per diagnostica. Inconveniente: a volte i sistemi intermedi effettuano modifiche dipendenti dal metodo

HTTP – Metodo OPTIONS

Serve per verificare quali capacità abbia il server http. In questo modo il client HTTP può conoscere quali operazioni può compiere, senza dover effettivamente accedere alle risorse. La richiesta può essere di varia natura. Generica: relativa a tutto il server. Specifica: relativa ad una singola risorsa. URL convenzionale * identifica il server in generale. Se applicata ad un URL specifico, la stessa richiesta potrebbe ricevere risposte differenti. Nota: «Connection closed by foreign host» non è parte del protocollo http. È dovuta al client usato per effettuare la richiesta ed indica che il server ha chiuso la connessione dopo aver restituito la risposta

10 HTTP

HyperText Transfer Protocol (HTTP) Gestione delle connessioni

HTTP – Connessioni in serie

Schema originario ereditato da HTTP/0.9. Esempio: pagina HTML che include altri due file (es. due immagini). In totale: per visualizzare completamente la pagina occorre acquisire tre risorse mediante tre transazioni, ognuna delle quali utilizza una connessione TCP indipendente dalle altre.

HTTP – Connessioni in parallelo

Rispetto al caso con connessioni in serie, in questo caso il client HTTP apre una nuova connessione TCP non appena scopre che deve acquisire una nuova risorsa. Tre transazioni che utilizzano tre connessioni TCP, ma due connessioni sono parallele e non sequenziali. Il tempo totale per visualizzare la pagina diminuisce rispetto al caso precedente.

HTTP – Connessioni sequenziali/parallele

Nel caso in cui la connessione di rete sia lenta, le connessioni parallele risultano essere di scarsa utilità. Obiettivo: eliminazione dei tempi «morti». Troppe connessioni contemporanee utilizzano memoria e possono provocare problemi. Non è risolto il problema del "TCP slow start". Algoritmo di controllo della congestione utilizzato nella fase iniziale di una connessione TCP, che provoca in genere una diminuzione del throughput (RFC2581). Un server HTTP può chiudere connessioni multiple provenienti da un medesimo client http.

HTTP – Connessione persistente

La connessione persistente è il vero punto di rottura rispetto allo schema di HTTP/0.9. Non è più utilizzabile il concetto di chiusura della connessione TCP per segnalare al client HTTP la fine della risorsa. Nuovo metodo: basato sull'uso dell'header Content-Length.

HTTP – Connessione persistente: keep-alive

1. Il client HTTP è a conoscenza del fatto che dovrà richiedere altre risorse (es. elementi grafici per completare la pagina HTML) • Richiede al server HTTP di non chiudere la connessione, inviando l'header Connection con valore keep-alive
2. Il server HTTP conferma il comportamento utilizzando il medesimo header field 1. Inoltre: inserisce l'header field Content-Length per informare il client HTTP sulla lunghezza attesa della risposta (necessaria per la fine del messaggio di risposta).

HTTP – Connessioni persistenti in HTTP/1.0 e HTTP/1.1

- HTTP/1.0 Di default la connessione è transitoria. Connessione keep-alive attivata su richiesta, il server può anche rifiutare. In caso di rifiuto: il server non restituisce l'header Connection: keep-alive. Connessione keep-alive soggetta a numero massimo di transazioni/timeout. Connessione keep-alive può dare problemi con dumb proxy.
- HTTP/1.1 Default: la connessione è persistente. Un client HTTP deve aprire al massimo due connessioni. Quando il client ha concluso le richieste, deve mandare l'header Connection: close.

HTTP – keep-alive e dumb proxy

Il problema deriva dalla presenza di proxy intermedi che non comprendono il significato dell'header Connection: keep-alive e non lo rimuovono dalla richiesta prima di inoltrarla al server. Funzionamento come blind relay. Il server non chiude la connessione completata l'interazione perché ha ricevuto una richiesta di keep-alive Il proxy attende la chiusura della connessione, ignorando ogni nuova richiesta sulla connessione corrente. La seconda richiesta emessa dal client sulla connessione in keep-alive si blocca, siccome il proxy non la processa mai Next request.

HTTP – keep-alive e dumb proxy

1. Un client HTTP invia un messaggio al proxy, inclusivo dell'header Connection: keep-alive (richiedendo una connessione keepalive, se possibile), ed attende una risposta.
2. Il dumb proxy prende la richiesta HTTP entrante, ma non sapendo come gestire l'header Connection, lo tratta come un semplice extension header, e pertanto inoltra il messaggio al successore. Problema: l'header Connection è hop-by-hop.
3. Quando il server riceve la richiesta, intende (erroneamente) che sia il proxy (che ai suoi «occhi» è il client) ad avere richiesto la connessione keep-alive. Quindi, il server risponderà confermando il keep-alive al proxy
4. Il dumb proxy riceverà la risposta dal server ma, non sapendo come interpretare l'header Connection: keep-alive, inoltrerà questa risposta al client reale, che purtroppo penserà che il proxy (che per il client è il server) acconsenta alla connessione keep-alive à Sia il client sia il server interagiranno la modalità keep-alive, non sapendo che il proxy intermedio non elabora l'header in modo corretto
5. Siccome il proxy non conosce il meccanismo del keep-alive, inoltrerà tutti i dati al client ed attenderà che sia l'origin server a chiudere la connessione à Problema: l'origin server non chiuderà mai la comunicazione, perché questa azione dovrebbe essere richiesta esplicitamente dal dumb proxy à Il proxy quindi si bloccherà in attesa che la connessione sia chiusa.
6. Dopo aver ricevuto la risposta alla prima richiesta, il client prova ad eseguire una seconda richiesta al proxy sulla medesima connessione keep-alive. Siccome il proxy non si aspetta ulteriori richieste sulla stessa connessione, ignorerà la richiesta
7. Questo errore di comunicazione causa il blocco del client fino a quando la connessione non scada e si chiuda per questo motivo.

HTTP/1.1: Connessione "pipelined"

Per ottenere un'ulteriore diminuzione dei tempi di latenza è possibile iniziare una transazione prima che una precedente sia conclusa. Naturalmente se si hanno le informazioni necessarie per iniziare le transazioni successive.

HyperText Transfer Protocol (HTTP) Sistemi intermedi

Proxy e Gateway

Origin Server: server HTTP che possiede la risorsa (o su cui la risorsa viene generata, nel caso di pagine dinamiche). Per ottenere la risorsa, il client può accedere all'origin server (accesso diretto) oppure ad un sistema intermedio. I sistemi intermedi attraversati possono essere molteplici à catena di nodi. Esempio: uno User Agent (UA) accede ad un Origin Server (O) attraverso tre sistemi intermedi A, B, C (il simbolo v indica una singola connessione TCP)

HTTP – Sistemi Intermedi

Tre tipi di sistemi intermedi (RFC 2616)

1. Tunnel: Non ha concezione dei messaggi, limitandosi ad inoltrarli
2. Proxy: È un forwarding agent. Riceve un URL (assoluto), ed è in grado di modificarlo. In grado di aggiungere header (modifica del messaggio). Inoltra la richiesta modificata
3. Gateway: È un receiving agent. Riceve una richiesta e la soddisfa. Inoltra una nuova richiesta usando un protocollo differente.

Proxy e gateway tengono copia delle risposte (cache) per velocizzare l'accesso alle risorse. Il tunnel agisce come un blind relay. Quando sia stato attivato, non viene più considerato parte della comunicazione HTTP (anche se il traffico fluisce attraverso di esso) e cessa di esistere quando le connessioni TCP sono chiuse. Non svolge funzioni di caching. Usato tipicamente per far transitare traffico non-HTTP su http. Es.: quando un firewall lascia passare solamente traffico di tipo HTTP: tramite il metodo CONNECT si ordina al nodo intermedio di comportarsi come un tunnel, ossia di stabilire una connessione ad un host/porta qualsiasi • RFC2616 prevede il metodo CONNECT ma non ne descrive il suo funzionamento. Transparent proxy: modifica i messaggi solamente per scopi di autenticazione o autorizzazione. Un proxy non transparent fornisce qualche servizio aggiuntivo (cambio versione protocollo, anonimato, ecc.). A differenza del proxy, un gateway si comporta come se fosse l'origin server.

Esempi di utilizzo di un proxy

- Controllo di accesso a documenti (ambito aziendale). Un proxy centralizzato controlla gli accessi a vari server aziendali e tiene traccia del traffico (log). Modifiche alle autorizzazioni richiedono interventi solamente sul proxy.
- Caching.
- Firewall (security). Soluzione: utilizzare un host (bastion machine) collegato sia ad Internet sia alla rete interna come unica via di collegamento. Connessioni TCP dirette tra ambito interno ed esterno sono impossibili e tale host funziona da proxy HTTP per consentire traffico Web.
- Anonymizer
- Surrogate proxy: Situato presso un server, Assume il medesimo indirizzo IP, Scarica da attività ripetitive
- Content router: Router che smista traffico verso server decentrati (mirror), Nel caso esistano più copie di uno stesso documento, distribuite in vari nodi, smista la richiesta al nodo più «vicino» (es. quello che ha minore ritardo), Utile in caso di condizioni di «flash crowds», Fenomeno per cui un sito Web riceve un numero molto elevato (e non previsto) di richieste, con conseguente picco di traffico inatteso.

Configurazione manuale dei parametri di un proxy nel browser

Vantaggi e limiti della configurazione manuale

È possibile definire più proxy a seconda del protocollo, L'accesso ad host interni può essere diretto, Inconvenienti. Un solo proxy per tutte le destinazioni. Modifica del proxy (es. indirizzo IP) richiede la modifica manuale in tutti i browser. La porta 8080 (che fa parte del blocco "registered ports") è dedicata al servizio HTTP, in alternativa alla well-known port 80. Molto spesso la porta 8080 e quelle consecutive (8081, 8082, ecc.) sono utilizzare come porte per i proxy. Da notare che 8081, 8082 e 8083 sono state registrate da IANA per altre funzioni.

Proxy Auto Configuration (PAC)

Tecnica introdotta da Netscape. Basato sull'utilizzo di un file in formato Javascript (PAC file). Estensione .pac. MIME type application/x-ns-proxy-autoconfig. Definisce come le richieste provenienti da un browser Web debbano essere gestite (sulla base del protocollo applicativo). Inviate direttamente all'URL di destinazione. Inoltrate ad un proxy server function FindProxyForURL(url, host) { ... }. La funzione restituisce una stringa testuale. Rispetto alla configurazione manuale: molto usata in ambito di reti aziendali, presenta il notevole vantaggio che, in occasione di un cambio di configurazione (variazione o introduzione di un nuovo proxy), è sufficiente un aggiornamento del PAC file centralizzato. Può contenere istruzioni anche elaborate per ottenere configurazioni condizionali.

Web Proxy Auto Discovery (WPAD)

L'URL del file PAC viene recuperato in modo automatico, attraverso una serie di tentativi: Richieste DHCP, Service Location Protocol (SLP) e Richieste DNS per un domain name prestabilito (es. wpad.example.com). Dynamic Host

Configuration Protocol (DHCP): RFC2131, è utilizzato per configurare dinamicamente un host che si collega ad una rete TCP/IP fornendogli indirizzo IP, gateway di default e server DNS • Tra le opzioni che può supportare, può fornire anche l'URL del file PAC.
Service Location Protocol (SLP): RFC2165, elimina la necessità di conoscere il nome di un host che fornisce un determinato servizio, supportando al contempo un meccanismo di configurazione dinamica. Sia DHCP sia SLP non sono destinati ad essere applicati sull'Internet pubblica, ma solamente all'interno di una LAN. Prima di recuperare la sua prima pagina, il browser Web (che implementa il metodo WPAD) inoltra una query di tipo DHCPINFORM al server DHCP locale, estraendo quindi l'URL del file PAC dalla option WPAD-Option nella risposta. Se il server DHCP non restituisce questa opzione, il browser ritenta con una richiesta DNS. Es.: nel caso in cui il network name del PC che sta emettendo la richiesta sia pc1.pal2.dia.unipr.it, il browser tenta una richiesta ad una gerarchia di URL (sino a trovare un PAC file).
Nota: su sistemi Windows, nel caso in cui la query DNS non avesse successo, sarebbero tentate anche delle richieste tramite i protocolli Link-Local Multicast Name Resolution (LLMNR) e NetBIOS.
Nota: DHCP ha priorità superiore a DNS, e WPAD funziona solamente con DHCPv4. In DHCPv6 non esiste una option di tipo WPAD-Option.
Nota: browser differenti possono avere comportamenti differenti.
Note relative alla sicurezza: Un attaccante interno alla rete locale potrebbe istanziare un server DHCP che restituisca un percorso dannoso per un PAC file. Siccome il browser difficilmente è in grado di comprendere quali siano i confini del dominio aziendale al cui interno effettuare le richieste, si corre il rischio di ottenere dei riferimenti completamente estranei al dominio locale (e dannosi). Nel caso un attaccante registrasse un dominio «valido» per una richiesta WPAD, si aprirebbe la possibilità di attacchi di tipo Man-in-the-Middle (MITM). Intercettazione del traffico utente • Soluzione: molti domini di tipo wpad.tld puntano all'indirizzo di loopback del client. Il file PAC, essendo un file in linguaggio Javascript, viene eseguito dal browser Web anche nel caso in cui fosse stata disabilitata l'esecuzione di tali file.
Nota per un amministratore di rete: utilizzare WPAD solamente se si prevede di istanziare un server in grado di rispondere a tali richieste.

Gestione di richieste HTTP ad origin server e proxy server

Un client invia richieste contenenti URL relativi ad un origin server, e URL assoluti ad un proxy server. Un surrogate proxy (o reverse proxy) non è visto dal client e riceve un URL relativo. Un intercepting proxy (o transparent proxy) non è visto dal client e riceve un URL relativo. Nel caso in cui un proxy riceva un URL relativo, utilizza l'header Host per ricostruire l'URL assoluto.

HyperText Transfer Protocol (HTTP) Meccanismi di caching

Definizione di cache

Meccanismo per mantenere una copia della risorsa in una memoria (cache) disponibile presso il client. È effettuata comunemente dai browser (cache locale) e dai proxy/gateway. Tende ad eliminare trasferimenti di dati ridondanti (accessi multipli alla stessa risorsa). Favorisce una diminuzione del traffico/costo di rete e la latenza percepita dall'utente finale. Riduce gli effetti di latenza in case il server su cui la risorsa è disponibile sia localizzato fisicamente a grandi distanze. Essenziale in caso di «flash crowd».

Cache hits e misses

- Cache hit: una richiesta viene soddisfatta, in termini di risposta emessa dal ricevente, con una copia della risorsa contenuta all'interno della cache
- Cache miss: una richiesta viene inoltrata all'origin server, poiché non esiste una copia della risorsa richiesta all'interno della cache
- Hit ratio (hit rate): corrisponde alla frazione di richieste soddisfatte sul totale, e fornisce un'indicazione sulla bontà della cache
- Byte hit rate: rapporto fra byte recuperati dalla cache e byte trasferiti su rete. A differenza dell'hit rate, considera le dimensioni delle risorse trasferite.

Rivalidazione di una risorsa

Il contenuto dell'origin server (in termini di risorse mantenute) può cambiare nel tempo.
Consigliabile (se non necessario) controllare periodicamente che il contenuto della cache sia allineato con l'origin server. La cache può effettuare questa verifica in qualsiasi momento. Normalmente il controllo viene effettuato solamente un client richieda il documento e la copia nella cache è abbastanza datata. Rivalidazione: La cache manda una richiesta al server per la risorse specifica. Se il codice di risposta corrisponde a 304 (Not Modified), la copia della risorsa nella cache viene segnata nuovamente come «fresca» e restituita al client.

Rivalidazione con metodo condizionale

La cache emette una GET condizionale: If-Modified-Since:, If-None-Match:.

If-Modified-Since: Se il documento è stato modificato si ottiene una risposta normale, Altrimenti un codice 304 senza body, richiesti solo header che cambiano. Es. Content-Type non serve.
If-None-Match (tag revalidation): Se il documento è stato riscritto: cambia la data, il contenuto non è detto. Si genera un Etag (Entity Tag) che cambia solo se ci sono modifiche effettive.

Cache private e cache pubbliche

Cache privata: dedicata ad un singolo utente: I browser hanno una cache privata incorporata. Cache pubblica: proxy server condiviso da molteplici utenti. Una cache condivisa ha più possibilità di ridurre il traffico di rete. Si configura nel browser come si configura un proxy (è un proxy a tutti gli effetti). Un intercepting proxy può forzare le richieste HTTP attraverso una cache senza interventi nelle impostazioni del browser.

Controllo della cache tramite HTTP-EQUIV

«Meta-tag» nel codice sorgente HTML. Problemi: Pochi server/proxy lo supportano, Alcuni browser li riconoscono, comportamento in contrasto con cache. Un meta-tag di tipo HTTP-EQUIV dovrebbe avere un effetto equivalente a quello ottenuto agendo sul protocollo http. Con riferimento all'esempio: si richiede che vengano tenute copie cached del documento HTML. Dovrebbe equivalere all'header HTTP Cache-control: no-cache. Però: l'origin server oppure il proxy dovrebbero analizzare l'header del documento e generare gli header HTTP corrispondenti à Questo in genere non avviene. Normalmente un browser analizza il meta-tag. Tuttavia: anche se il browser rispetta il meta-tag e non tiene il documento in cache, quando effettua una nuova richiesta essa può essere fatta a una cache intermedia che gli serve una copia cached, anche se questa non era la volontà dell'autore del documento HTML. L'uso dei meta-tag non fornisce garanzie.

11 http

HyperText Transfer Protocol (HTTP) Virtual host, content negotiation, trasferimento parziale

Funzionalità di Virtual Hosting

Scopo: molteplici siti Web ospitati su uno medesimo host
Architettura: Document Root separate. Ogni virtual host ha la propria struttura di documenti
Come distinguere i virtual host: Path; Numero di porta; Indirizzo IP (IP-based virtual host, vanity address); Header Host (name-based virtual host, obbligatorio in HTTP/1.1). Ogni virtual host ha il proprio domain name.
La distinzione sulla base dell'indirizzo IP e dell'header Host sono preferibili rispetto alle altre poiché evitano di dover inserire nell'URL di un sito elementi come path o numero di porta: Indirizzi «puliti» stile www.mysite.it.
Assegnare un indirizzo IP ad ogni virtual host è troppo dispendioso in termini di consumo di indirizzi IP: Risorsa scarsa (specialmente IPv4), Soluzione generalmente abbandonata in favore dei name-based virtual host (utilizzo dell'header Host).

Name-based Virtual Host

Nel caso in cui manchi l'header Host (raro con i browser attuali, obbligatorio in HTTP/1.1), alcuni server HTTP restituiscono una home page convenzionale contenente, ad esempio, un elenco di link che portano alle varie home page dei virtual host. Attenzione alle problematiche di sicurezza. Apache httpd: come pagina di default, serve il default document del primo virtual host elencato nel file di configurazione dei name-based virtual host. Il primo virtual host viene scelto ogni volta che venga scelto un header Host non valido.

Content Negotiation – Formazione e interpretazione dell'header Host

- In emissione (request message): Nell'header Host deve essere inserita una replica del domain name dell'URL. Se nell'URL è presente un indirizzo IP, deve essere replicato l'indirizzo IP. Se è in uso un proxy, è necessario indicare il domain name dell'origin server, non del proxy
- In ricezione (response message). Se l'origin server non è organizzato a virtual host, il contenuto dell'header viene ignorato. Se l'URL del request message è assoluto, ha la precedenza sull'header. Se l'URL del request message è relativo (non ha il campo Host), il valore host/port si ottiene dall'header
- HTTP/1.1: anche se il server non è organizzato a virtual host, ma riceve una richiesta in versione HTTP/1.1, la mancanza dell'header Host provoca un errore.

Content Negotiation – Request Message e Message Entity

Message Entity: insieme degli header che portano informazioni relative al contenuto, e del body del messaggio stesso.
Nota: gli header relativo al contenuto non sono necessariamente sono contigui.

Content Negotiation – Contenuto dell'Entity

Testo e Altro: Immagini, Audio e Applicazione. Il tipo di contenuto è identificato dall'entity header Content-Type. La relativa codifica può essere specificata tramite il parametro charset dell'entity header Content-Type. I valori leciti per questi parametri sono i valori registrati per la codifica MIME

Content Negotiation – Codifica Entity, varianti e loro negoziazione

Il browser deve interpretare correttamente i byte del body, potrebbe non riconoscere alcune codifiche, potrebbe non essere in grado di rappresentare alcuni script. Es. testi in codifica orientale. Il server potrebbe disporre di molteplici

varianti (lingue differenti, codifiche differenti, risoluzione grafica, ecc.) della stessa risorsa. Gestione mediante un'interazione client-server definita Content Negotiation. Questa interazione sfrutta appositi header. Es.: Accept-Charset.

Transparent Content Negotiation (TCN)

La scelta della variante viene lasciata al client che, in base alla lista di risorse che riceve, emette una seconda request. La scelta può anche essere tentata dal server. Il server utilizza il response code 200 e, al posto di TCN: list, utilizza l'header TCN: choice seguito poi dalla eventuale lista. Il client decide quindi se accettare la scelta. Si può così evitare al client di effettuare la seconda request. Valori del parametro «source quality»: 1.000 perfect representation, 0.900 threshold of noticeable loss of quality, 0.800 noticeable, but acceptable quality reduction, 0.500 barely acceptable quality, 0.300 severely degraded quality, 0.000 completely degraded quality. Soluzione comunque poco diffusa.

Content Negotiation – Codifica testo: Entity Header e Meta Tag

Scopo del meta tag: il server HTTP dovrebbe analizzare il documento e generare gli header corrispondenti. In realtà pochi server lo fanno. Non tutti i browser considerano il meta tag. In definitiva è sconsigliato. In ogni caso se un client non riesce a inferire la codifica di un testo deve assumere ISO-8859-1.

Content Negotiation – Documento codificato: Content Encoding

HTTP prevede la possibilità di codificare l'entity body. Nel caso in cui il body sia stato codificato. Il server inserisce l'header Content-Encoding. L'header Content-Length specifica la lunghezza dopo la codifica. Il client che riceve un body codificato deve decodificarlo come prima attività. Tipi di codifica previsti: gzip (RFC1952), compress (Unix file compression), deflate (RFC1950, RFC1951), identity (default). Esempio: Content-Type: image/gif e Content-Encoding: gzip. N.B.: Questa codifica non va confusa con quella specificata dai parametri di Content-Type.

Trasferimento parziale: Chunked Encoding

Un messaggio viene consegnato dal server spezzato in molteplici chunk di dimensione nota. Motivazione: con le connessioni persistenti

1. Il server deve specificare l'header Content-Length
2. Se il documento è dinamico la lunghezza non è nota a priori.

Soluzione: Chunked-Encoding.

Con questa codifica non si usa Content-Length. Alcuni server usano sempre questa codifica, anche per documenti costituiti da un solo chunk. In tal caso viene servito un unico blocco contiguo.

Trasferimento parziale: Range Request

Messaggio GET in cui il client non richiede l'intera risorsa ma solamente una parte di essa (range). Caratterizzata dall'header Range e possibile solamente se il server ha preventivamente dichiarato, mediante l'header Accept-Ranges, la sua disponibilità ad accettare richieste di tale tipo • Utile in caso di crash del trasferimento di una risorsa di grandi dimensioni. Quando il client riesce a ricontattare il server, non richiede più l'intera risorsa, ma solamente la parte mancante. Il parametro bytes dell'header Accept-Ranges indica l'unità di misura usata per misurare l'ampiezza del trasferimento parziale.

12 HTTP

HyperText Transfer Protocol (HTTP) Autenticazione e Sicurezza

Autenticazione in http

Autenticazione: l'entità che si presenta deve dichiarare le proprie credenziali. Un server HTTP può restringere l'accesso ad alcune risorse mediante un meccanismo di autenticazione («accede solamente chi si qualifica»). Le risorse protette sono organizzate in «realm». Un realm è un insieme di risorse identificato da una stringa case insensitive. URL + realm = protection space (univoco nel Web). Per accedere ad uno spazio protetto, il client deve autenticarsi fornendo una coppia per ogni «realm».

Processo di autenticazione

1. L'utente vuole accedere a una risorsa protetta
2. Il client invia una richiesta HTTP di tipo GET (priva di autorizzazione) per la risorsa protetta
3. Il server risponde con un messaggio di challenge.

Quando lo UA riceve uno status code 401 (Unauthorized) richiede all'utente l'inserimento della coppia. La coppia viene codificata in formato Base64. L'UA ritenta l'accesso inviando una richiesta contenente le credenziali fornite. Tutti gli accessi successivi saranno con credenziali. Richiesta HTTP GET con credenziali. Messaggio con header field Authorization. Nel caso in cui le credenziali fornite non fossero corrette, il server HTTP restituisce uno status code 403 (Forbidden). Ogni server può decidere autonomamente se visualizzare una pagina HTML informativa sul tipo di situazione riscontrata.

Schemi di autenticazione

RFC2617 «HTTP Authentication: Basic and Digest Access Authentication».

- Basic: La coppia è trasferita sostanzialmente in chiaro (codificata solamente tramite codifica Base64). Anche se fosse crittografata, una volta catturata potrebbe essere utilizzata per accessi successivi. Schema accettabile per riservatezza in ambienti «amichevoli» (non critici);
- Digest: La password non viene mai trasferita in rete. Al posto della password, viene utilizzato un suo «digest» (es. MD5). Per evitare repliche, il client invia il digest della password combinata con un token variabile di volta in volta (nonce) ricevuto dal server in fase di riposta alla prima GET di accesso alla risorsa http.

Processo di autenticazione con schema Digest

La string opaque serve per correlare richieste successive relative allo stesso realm. Le credenziali sono contenute nella chiave response, e corrispondono all'hash (ossia il digest) MD5 di una stringa «complessa».

Problemi legati alla sicurezza

Privacy (un malintenzionato potrebbe vedere il contenuto della comunicazione) • Integrità (un malintenzionato potrebbe modificare il contenuto della comunicazione) • Autenticazione (non vi è la certezza di chi sia la controparte con cui si sta interloquendo). Nel gestire questi aspetti si usano comunemente tecniche di crittografia.

Algoritmi di crittografia

- Algoritmi simmetrici: Utilizzo della medesima chiave crittografica sia per la fase di codifica, sia per la fase di decodifica, Veloce (semplice realizzare un algoritmo anche lato HW), Problema: distribuzione della chiave tra le entità, «Esplosione» del numero delle chiavi;
- Algoritmi asimmetrici: Utilizzo di una chiave crittografica Kenc per la fase di codifica, e di un'altra chiave crittografica Kdec per la fase di decodifica. Vantaggio: solamente due chiavi, gestione semplice. Chiavi invertibili: si codifica con una qualunque delle due chiavi, si decodifica con l'altra. Problema: richiede molta elaborazione (lento). Algoritmo più noto: RSA (Rivest, Shamir, Adleman).

Sistema basato su chiavi pubbliche

Si basa sull'utilizzo di un algoritmo asimmetrico. Una chiave è resa pubblica (Kpub), l'altra rimane privata (Kpriv). L'utente A pubblica la propria chiave KPUB-A. L'utente B codifica un messaggio con la chiave pubblica dell'utente c=(m)Kpub-a. L'utente B manda il messaggio all'utente A. Solo A è in grado di decodificarlo correttamente (tramite la propria chiave privata Kpriv-a) m= (c)Kpriv-a

Nota: conoscendo le seguenti informazioni: Chiave pubblica, Algoritmo di crittografia (di solito è reso pubblico), Messaggio in chiaro (chiunque può inventarlo) e Messaggio cifrato (chiunque può ottenerlo) NON è possibile ricavare la chiave privata.

Equivalenza tra algoritmi a chiave simmetrica ed asimmetrica

A pari lunghezza in bit, la chiave pubblica viene scoperta prima. Due algoritmi sono considerati avere «equivalent strength» per le chiavi di dimensione X e Y se la quantità di lavoro richiesta per «rompere l'algoritmo» o per determinare le chiavi (data la dimensione delle chiavi) è approssimativamente la medesima utilizzando una certa risorsa.

Firma digitale

Metodo matematico applicato con l'obiettivo di dimostrare l'autenticità di un messaggio o di un documento digitale inviato tra mittente e destinatario attraverso un canale di comunicazione non sicuro. Obiettivo: Garantire al destinatario che il mittente del messaggio sia chi dice di essere (autenticazione), Garantire al destinatario che il mittente non possa negare di averlo inviato (non ripudio), Garantire al destinatario che il messaggio non sia stato alterato lungo il percorso. Le firme digitali si basano su schemi o protocolli crittografici a chiave pubblica. Solitamente la chiave privata viene mantenuta all'interno di smart card, dispositivi OTP, dongle USB, App mobile.

Public Key Infrastructure (PKI)

Struttura che associa una chiave pubblica all'identità di un soggetto, per mezzo di un certificato. Permette ad entità che non si siano mai contattate in precedenza di autenticarsi a vicenda e di utilizzare le proprie chiavi pubbliche per scambiarsi messaggi crittografati. In generale una PKI può comprendere SW lato client, SW lato server, hardware (es. smart card), contratti legali, procedure operative. Una PKI (ad esempio in un ambito aziendale) può comprendere più livelli gerarchici. Es. un soggetto garantito da un certificato emette a sua volta certificati per garantire altri soggetti.

Componenti di una PKI

- Una PKI è composta da diverse entità: Certificati digitali, Autorità di certificazione (Certification Authority, CA), Meccanismo di distribuzione dei certificati digitali, Certificate Revocation List (CRL), corrispondente ad una lista di Serial Number (SN) di certificati compromessi;
- Certificato digitale: Documento che associa all'identità una chiave pubblica ed informazioni aggiuntive, Generato da una CA, Protetto da una firma digitale della CA. L'utente ha il compito di verificare la validità di un certificato, Formato dei certificati: X.509 (ver. 3);
- Normalmente i browser contengono una lista pre-installata di CA note • Sono in grado di verificare la maggior parte dei certificati digitali in circolazione;

- Una CA emette certificati che sono usati da terzi. È un esempio di trusted third party.
- Molte CA commerciali fanno pagare per i loro servizi, ma esistono anche dei provider che emettono certificati gratuitamente. Istituzioni governative possono avere le loro proprie CA;
- Esistono due diversi possibili stati di revoca (RFC 3280). Revoked: un certificato è revocato in modo irreversibile se (ad esempio) la CA ha emesso un certificato improprio oppure se la chiave privata è stata compromesso. Caso più comune: perdita della chiave privata da parte dell'utente. Hold: stato reversibile, in cui il certificato è bloccato temporaneamente. Es.: l'utente non è sicuro di aver perso la chiave privata.

Certificato in formato X.509 – Campi Informativi

Versione (3) • Serial Number (SN), Algoritmo di firma digitale, Ente emittente (CA), Periodo di validità, Nome del soggetto, Chiave pubblica del soggetto, (… Informazioni aggiuntive …) e Firma digitale (digest).

Transport Layer Security (TLS)

Strato intermedio tra layer di trasporto (TCP) e layer applicativo. Consente di risolvere vari problemi: Privacy, Integrità e Autenticazione.

TLS – Fasi operative

1. Negoziazione. Fase in cui viene deciso il meccanismo di crittografia (cipher) da adottare: Crittografia asimmetrica: RSA, Diffie-Hellmann, DSA, Fortezza. Crittografia simmetrica: RC2, RC4, IDEA, DES, Triple DES, AES. Hash: MD5, SHA
2. Autenticazione. Basata su algoritmo asimmetrico (utilizzo di public key) • Scambio di certificati • Scambio di chiave simmetrica
3. Traffico. Crittografato mediante la chiave simmetrica

HTTP tunneling

- Metodo CONNECT: Interpretato dal proxy, Richiede al proxy di modificare il proprio comportamento, Il proxy non si comporta più come tale (ossia non interpreta più i comandi), Il proxy diventa trasparente (diventa un "tunnel");
- Altri tipi di tunnel: Trasporto di protocollo generico su HTTP, Serve a superare quei firewall che lasciano passare solo traffico HTTP.

Autenticazione Kerberos

1. Richiesta di un Ticket-Granting Ticket (TGT) all'Authentication Server
2. L'Authentication Server verifica i diritti di accesso dell'utente e, in caso positivo, crea un TGT ed una session key, li cifra entrambi con una chiave derivata dalla password dell'utente, e li restituisce al client à il client richiede all'utente l'inserimento della propria password per decifrare il messaggio in ingresso e, in caso positivo, potrà utilizzare il TGT per richiedere un service ticket
3. Nel momento in cui l'utente voglia accedere al servizio, il client invia una richiesta al Ticket- Granting Server (TGS) contenente il client name, il realm name ed un timestamp à l'utente dimostra la propria identità inviando un autenticatore cifrato con la session key ottenuta al passo 2
4. Il TGS decifra il ticket e l'autenticatore, verifica la richiesta, e crea un ticket per il server richiesto, contenente il client name (e l'indirizzo IP, opzionale), il realm name ed il tempo di validità del ticket à il TGS restituisce il ticket al client, che conterrà anche due copie di una server session key – una cifrata con la password dell'utente, ed una cifrata con la password del servizio
5. Il client invia una service request al server contenente il ticket ricevuto al passo 4 ed un autenticatore à il servizio autentica la richiesta decifrando la session key, verifica che il ticket e l'autenticatore coincidano e, in caso positivo, concede l'accesso al servizio. Replay attack: per prevenirli, Kerberos utilizza dei timestamp à Il clock dei client e del server devono essere il più possibile sincronizzati, ma sapendo che questo è complicato, è possibile definire delle policy sul disallineamento massimo ammissibile (normalmente è pari a 5 minuti).

Autenticazione OAuth2

- Ruoli previst i in OAuth2.
  - Resource Owner: entità che può concedere ad un'applicazione client un accesso ben definito ad una risorsa
  - Client Application: applicazione che accede a delle risorse protette in vece del Resource Owner • Resource Server (API Server): gestisce l'accesso ad una risorsa protetta, consentendo l'accesso sulla base di Access Token
  - Authorization Server (OAuth2 Server): emette gli Access Token a favore delle Client Application autenticate nel momento in cui i permessi di accesso siano concessi dal Resource Owner
- Token utilizzati in OAuth2

- Access Token: ha una durata limitata nel tempo, definisce le politiche di accesso consentite ad una risorsa protetta da parte di un'applicazione client, ed è fornito al Resource Server come header HTTP à solitamente è consigliabile avere Access Token con durata breve
- Refresh Token: utile per richiedere un nuovo Access Token nel momento della sua scadenza • Authorization Code: restituito ai clienti dall'Authorization Server ed utilizzato successivamente per ottenere un Access Token.

Autenticazione JSON Web Token (JWT)

Definisce un meccanismo compatto e self-contained per trasferire informazioni in modo sicuro tra entità differenti mediante un oggetto JSON. Informazioni verificate e trusted poiché firmate digitalmente mediante crittografia simmetrica (algoritmo HMAC) ovvero crittografia asimmetrica (algoritmo RSA o ECDSA). Composto da 3 parti: header, payload, signature. Utilizzi principali: Autenticazione e autorizzazione: a valle del login, ogni richiesta successiva includerà il JWT (header Authorization: bearer), consentendo un controllo puntuale sulle route, i servizi e le risorse consentite per l'utente corrente. Scambio di informazioni: certezza del mittente e verifica della non manomissione del payload (siccome la firma del JWT è calcolata sull'header e sul payload).

13 HTTP

HyperText Transfer Protocol (HTTP) Web Cookies

Sessione nel Web

Il mantenimento di una sessione nel Web richiede dei costrutti aggiuntivi rispetto ad HTTP di base. «Fat URL»: Corrispondente ad un URL che contiene informazioni specifiche riguardanti l'utente. Cookie: Corrispondente ad un chunk di informazione che viene scambiato tra client e server per tenere traccia dell'interazione. Fat URL e cookie possono essere utilizzati insieme.

Sessione nel Web tramite cookie

Descrive un metodo per instaurare una sessione mediante messaggi HTTP. Metodo basato su blocchi di informazione chiamati cookie. Chunk informativi scambiati mediante appositi header field dei messaggi HTTP.

Cookie – Funzionamento di base

Mediante i cookie, il server deposita delle informazioni presso il client per poi recuperarle in un secondo momento. Stringa (senza caratteri speciali) restituita al client (all'interno del messaggio di risposta) tramite l'header Set-Cookie. Se accetta il cookie, il client quando accede al dominio www.abc.it inserisce all'interno della richiesta l'header field Cookie: Customer="ab32". La decisione sull'utilizzo dei cookie (e quindi sull'inizio di una sessione) spetta sempre al server. Il contenuto del cookie ha significato solamente per il server.

Cookie – Esempio di sessione

1. L'utente si identifica sul sito Web dello store online
2. L'utente sceglie gli articoli da acquistare
3. L'utente sceglie il metodo di pagamento
4. L'utente conferma l'acquisto

Fase #1

Il cookie con chiave Customer serve per identificare l'utente. Il cookie con chiave session serve per identificare la sessione. Gli attributi negli header Set-Cookie sono opzionali.

Fase #2

In risposta alla richiesta HTTP di tipo POST, il server restituisce un nuovo cookie che conferma che il carrello contiene un nuovo articolo.

Fase #3

Il nuovo cookie indica il metodo di pagamento scelto dall'utente tramite la richiesta HTTP di tipo POST precedente.

Fase #4

Transazione completata.

Attributi dei cookie: Domain e Path

Il primo campo (<cookie_name> = <cookie_value>) è obbligatorio. Tutti gli attributi che seguono sono opzionali.

- Attributo domain: Specifica un dominio nel quale ha validità il cookie. Nel caso in cui l'attributo domain non sia presente: vale il contenuto dell'header Host (default);
- Attributo path: Specifica un sottoinsieme di URL per i quali il cookie è valido. Nel caso in cui l'attributo path non sia presente: vale l'URL che ha provocato la risposta con Set-Cookie.

Attributi dei cookie: Expires e Max-Age

- Attributo Expires: Specifica la scadenza in formato Wdy, Superato il riferimento di scadenza, il client elimina il cookie. Nota: l'utilizzo dell'attributo Expires è stato sconsigliato da RFC2965;
- Attributo Max-Age: Specifica la validità del cookie in secondi. Trascorso un periodo pari a Max-Age, il client elimina il cookie. Un valore di Max-Age pari a zero richiede al client di scartare immediatamente il cookie

(tecnica solitamente utilizzata per chiudere una sessione). Nel caso in cui Max-Age/Expires non siano presenti, il cookie viene eliminato con la chiusura dell'istanza del browser.

Attributi dei cookie: Secure

Nel caso in cui l'attributo secure sia presente, il cookie deve essere accettato solamente se la comunicazione avviene tramite il protocollo HTTPS. HTTP crittografato mediante SSL/TLS su porta TCP 443.

Tipologie di cookie – Ciclo di vita

- Cookie di sessione: Non vengono memorizzati in modo persistente sul dispositivo dell'utente e vengono cancellati alla chiusura del browser. A differenza di altri cookie, non hanno una data di scadenza, ed in base a questo il browser riesce ad identificarli come tali;
- Cookie persistenti: Scadono ad una specifica data ovvero passato un determinato periodo di tempo. Per l'intera durata di vita del cookie, le sue informazioni verranno trasmesse al server ogni volta che l'utente visita il sito Web, ovvero ogni volta che l'utente visualizza una risorsa appartenente a tale dominio da un altro sito Web (es. annuncio pubblicitario). Solitamente possono essere utilizzati dagli inserzionisti per registrare le informazioni sulle abitudini di navigazione Web di un utente per un periodo prolungato di tempo. Nota: essi sono utilizzati anche per motivi «legittimi» (es. riconoscere gli utenti evitando di dover inserire, ad ogni visita, le credenziali per l'accesso ai siti Web).

Tipologie di cookie – Provenienza

- First-party cookie • Contengono un attributo di dominio corrispondente al dominio che viene visualizzato nella barra degli indirizzi del browser Web • Possono essere sia persistenti sia di sessione (solitamente utilizzati per tenere traccia di preferenze espresse in merito all'uso del sito Web stesso)
- Third-party cookie: Appartengono a domini differenti da quello mostrato nella barra degli indirizzi. Appaiono in genere quando le pagine Web sono dotate di contenuti (es. banner pubblicitari) da siti Web esterni. Implicano la possibilità di monitoraggio della cronologia di navigazione dell'utente. Personalizzazione della navigazione dell'utente. Tracciamento della navigazione e costruzione di una cronologia di navigazione degli utenti in tutti i siti che hanno contenuti del fornitore di cookie esterni. Nota: la maggior parte dei browser Web dispongono di impostazioni di privacy in grado di bloccare i cookie di terze parti.

Tipologie di cookie – Finalità di utilizzo

- Cookie tecnici: Utili per la navigazione (in quanto necessari la consultazione dei contenuti e l'erogazione del servizio) e per facilitare l'accesso e la fruizione del sito da parte dell'utente. Essenziali anche per operazioni delicate quali home banking o pagamento tramite moneta elettronica
- Cookie statistici: Utilizzati a fini di ottimizzazione del sito mediante una raccolta di informazioni in forma aggregata sul numero degli utenti e sulle modalità di fruizione del sito Web. Valgono le stesse regole, in tema di informativa e consenso, previste per i cookie tecnici
- Cookie per la memorizzazione delle preferenze: Utili per favorire l'utilizzo corretto del sito Web da parte dell'utente (es. per tenere traccia della lingua scelta)
- Cookie di marketing e profilazione (pubblicitari): Servono per riconoscere i singoli messaggi pubblicitari e conoscere quali sono stati selezionati e quando. Vengono utilizzati anche per una profilazione mirata (ed auspicabilmente anonima) dell'esperienza di navigazione dell'utente all'interno del sito Web
- Cookie di social network: Consentono di condividere anche con altri utenti i contenuti del sito che si sta visitando. Tipicamente utilizzati per attivare funzioni di condivisione quali «Like», «Follow», o «Share» dei social network. Queste funzioni consentono alle reti sociali di identificare i propri utenti e raccogliere informazioni anche mentre navigano su altri siti.

Sicurezza legata ai cookie

- Intercettazioni di rete: Un cookie può essere rubato da un altro computer intercettando il traffico di rete (es. rete Wi-Fi non criptata). Apre scenari di attacchi di tipo MITM. Un attaccante potrebbe utilizzare i cookie intercettati per impersonare la vittima proprietaria dei cookie, ed effettuare operazioni dannose. Soluzione: utilizzare una comunicazione cifrata utilizzando HTTPS;
- Falsi sotto-domini & DNS cache poisoning: Un attaccante in grado di far utilizzare alla vittima un server DNS fittizio/fasullo (DNS cache poisoning) potrebbe ricevere richieste per le risorse di interesse per la vittima, contenenti i cookie inviati dal browser della vittima stessa. La gravità di questo attacco può essere ridotta se il sito destinatario utilizza cookie protetti. L'attaccante dovrebbe essere in grado di ottenere il certificato SSL del sito Web di destinazione da una CA à In assenza di un certificato SSL, i browser delle vittime visualizzerebbero un warning relativo al certificato non valido del sito web dell'attaccante.
- Cross-site scripting: furto dei cookie: Si verifica quando un attaccante sfrutta un sito Web che permette agli utenti di inviare contenuto non filtrato HTML e JavaScript Click here!. Se la connessione avviene mediante

HTTPS, i secure cookie saranno inviati all'attaccante anch'essi in formato testo. È responsabilità degli sviluppatori dei siti Web filtrare tale codice dannoso

- Cross-site scripting: richiesta proxy: La vittima sta leggendo un post di un attaccante su un sito fidato (trusted), e lo script dell'attaccante viene eseguito nel browser della vittima, generando una richiesta per l'URL fidato con l'URL malevolo come server proxy. Siccome la richiesta è per il sito trusted, tutti i cookie di quel dominio verranno inviati insieme alla richiesta, tuttavia venendo instradati attraverso il server proxy dell'attaccante. Soluzione: utilizzo di secure cookie e di connessioni HTTPS, provviste di una crittografia end-to-end (informazioni crittografate sul browser della vittima e decifrate sul server di destinazione) à Il server proxy vedrebbe solamente i primi byte cifrati della richiesta HTTP.

Aspetti giuridici legati ai cookie

I cookie devono essere trattati come dati personali. I dati personali non possono essere tracciati o usati prima che l'utente abbia dato il consenso esplicito. Devono essere specificati tutti i tracciamenti dei dati personali su tutte le pagine/URL coinvolti del sito. Gli utenti del sito devono essere informati in un linguaggio semplice su vari aspetti. Chi riceve i loro dati e come sono usati. La data di scadenza dei cookie. Ogni autorizzazione deve essere salvata/registrata per provare alle autorità che è stata concessa (formato della prova). La revoca del consenso deve essere facile da fare, anche in un secondo momento. La cookie policy deve contenere la lista di tutti i cookie presenti su tutte le pagine del sito e, per ognuno di essi, chi riceve i dati, per che scopo sono utilizzati, e la data di scadenza. Nel caso di un sito Web con la presenza di soli cookie tecnici/funzionali e/o di cookie di statistica di prima parte. Non è necessario avere un cookie banner, il blocco preventivo dei cookie prima del consenso ed il salvataggio dei consensi (formato della prova). In questo caso rientrano anche i third-party cookie per statistica. Es. Google Analytics con IP anonimizzato, in modo tale che non possa essere ricostruito l'indirizzo IP con tecniche di reverse engineering. Necessario però disattivare la condivisione dei dati con gli altri prodotti/servizi Google, altrimenti si rientra nel caso successivo. Nel caso siano presenti altri tipi di cookie, e quindi quelli pubblicitari (es. Google Adsense, DoubleClick, ecc.) e tutti gli altri third-party (social network, ecc.), oltre al cookie banner sono richiesti anche altre azioni. Consenso esplicito (opt-in) e quindi non più implicito (es. «scrollare» la pagina). Blocco preventivo dei cookie prima del consenso. Registrazione dei log del consenso da fornire come «prova». Descrizione per ogni cookie che comprende chi riceve i dati, per quale scopo e la data di scadenza. Possibilità di revocare il consenso in modo semplice anche in un secondo momento.

14 HTTP

HyperText Transfer Protocol (HTTP) Web Robot

Web Robot

Applicativo SW che automatizza l'esecuzione di una serie di transazioni (spesso ripetitive) senza la necessità di un intervento umano. Nella pratica, è un client HTTP con differenti task: Aggiornamento del valori dei titoli azionari (stock graphing), Calcolo di statistiche relative al Web, Servizi di comparison shopping e Motori di ricerca (indexing e Web crawling)

Robot – Evitare i cicli

Il robot rischia di non uscire dal ciclo. Il server può risultare sovraccaricato, fino a provocare un fenomeno di Denial of Service (DoS) = Azione illegale! Informazioni ripetute e inutili. Soluzione: tenere sempre traccia degli URL visitati. Attenzione all'ingombro degli URL. URL medio: 70 caratteri. 55 miliardi di URL. 70 x 55 x 109 = 3850 x 109 = 3.85 TB! Partitioning: assegnare ad un robot un sottoinsieme tra tutti gli URL possibili.

Robot - Cicli ed alias degli URL

Uno stesso documento può avere molteplici URL. Occorre rendere «canonici» gli URL. Aggiungere la porta :80 al campo Host. Convertire i caratteri escaped (%7e à ~). Eliminare i fragment identifier (URL col simbolo #). Accorgimenti importanti ma non sufficienti.

Robot – Cicli di File System

Un nome di directory può essere reale (OK). Un nome di directory può essere un link simbolico (loop!).

Robot – Come evitare loop ed altri problemi

Rendere gli URL canonici. Navigare in modalità breadth first. «Strozzare» richieste troppo vicine (throttling). Limitare la dimensione degli URL. Evitare URL inseriti in blacklist. Pattern detection. Monitoraggio umano (non lasciare andare il robot solamente in modalità solitaria).

Robot exclusion

Convenzione utilizzata per controllare l'accesso ai siti Web da parte dei robot. Si basa sull'utilizzo di un file di nome robots.txt contenuto nella document root. Se questo file manca, si suppone che non ci siano limitazioni alla scansione del sito Web da parte del robot. Non è prevista la possibilità di avere file robots.txt in altre directory. Come prima azione, un robot che accede ad un server HTTP deve verificare l'esistenza di questo file.

Formato del file robots.txt

User-Agent: MyRobotPR
Disallow: /private
User-Agent: *
Disallow:
MESSAGE QUEUE TELEMETRY TRANSPORT (MQTT) pg46