Introduzione al Web

Nozioni di base utili per comprendere e approfondire le tecnologie e le caratteristiche del web

Giuseppe Della Penna
Università degli Studi di L'Aquila

giuseppe.dellapenna@univaq.it
http://people.disim.univaq.it/dellapenna

Versione documento: 221102

This work is licensed under CC BY-NC-SA 4.0. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-sa/4.0/

A cosa servono queste slide?

- Il corso di **Web Engineering** è un corso *avanzato* che tratta lo sviluppo web su particolari *piattaforme* e spesso con un *punto di vista* specifico.
- Sebbene nel corso vengano richiamate molte nozioni di base sulle tecnologie del Web (ad esempio i linguaggi HTML e CSS), per poi approfondirne aspetti meno noti, è utile che tutti coloro i quali affrontano questo corso abbiano una *base comune*, seppur esigua, di conoscenze al riguardo, che eviteremo così di ripetere durante le lezioni.
- Inoltre, per capire veramente il Web bisogna conoscere elementi di base inerenti molti *argomenti trasversali*, quali le reti. In queste slide troverete una serie di definizioni che, sebbene molto basilari, vi permetteranno di colmare anche questo tipo di lacuna.
- Infine, nel corso parleremo ampiamente di *privacy* e *sicurezza*, e in queste slide ho voluto raccogliere in maniera più organizzata molte delle osservazioni in merito che faremo durante le lezioni.

Nozioni di base

Quelle che quasi tutti conoscono, ma quasi mai correttamente

This work is licensed under CC BY-NC-SA 4.0. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-sa/4.0/

Gli elementi di base del Web

- Il web moderno, inteso come l'insieme di applicazioni o siti accessibili tramite un web browser (che però, va ricordato, non è l'unico tipo di servizio presente su internet) è costruito su una serie di pilastri fondamentali
  - La trasmissione di dati tramite il protocollo applicativo HTTP sullo stack TCP/IP (proprio di tutta internet)
  - La codifica dei dati testuali tramite UNICODE
  - La strutturazione dell'informazione tramite L'Hypertext Markup Language (HTML)
  - La rappresentazione visiva dell'informazione ottenuta tramite i Cascading Style Sheets (CSS)
  - L'interazione e la dinamica programmata nei browser (client) tramite il linguaggio Javascript
  - La manipolazione dell'informazione grezza e la sua conservazione, operata sui server con una varietà di linguaggi server side (Java, PHP, Python, ma anche lo stesso Javascript,…) e di DBMS (MySQL, Oracle, SQLServer,…)

Nozioni di base
Vulnerabilità

- Una **vulnerabilità** è un problema del software che può essere sfruttato per effettuare operazioni (accedere o manipolare dati, ecc.) che normalmente non sarebbero possibili, solitamente con scopi malevoli.
- Il sistema CVE (**Common Vulnerabilities and Exposures**) è il riferimento pubblico per tutte le vulnerabilità software note. Gli identificatori CVE assegnano a ciascuna vulnerabilità un nome specifico, in modo da portevi fare correttamente riferimento.
  - CVE è mantenuto dal National Cybersecurity FFRDC degli Stati Uniti.

Nozioni di base
Rete

- Una **rete** (di dispositivi) è un sistema di collegamento che consente lo scambio di dati e la collaborazione fra i dispositivi.
  - Le **reti locali** (LAN, Local Area Network) interconnettono dispositivi in un ambito limitato (stanza, edificio, ecc.)
  - Le **reti geografiche** (WAN, Wide Area Network) interconnettono dispositivi posti in luoghi geograficamente anche molto distanti
- Una **internet** (con la «i» minuscola) è costituita da un insieme di reti connesse tra loro, che stabiliscono un metodo comune di comunicazione (protocollo) ed eventualmente dei sistemi per tradurre i protocolli usati nelle singole sotto-reti in quello comune.
  - Le **reti private virtuali** (VPN, Virtual Private Network) possono essere viste come un'estensione a livello geografico delle reti locali, che utilizzano una internet pubblica, e non una costosa linea privata, per connettere LAN fisicamente distanti. I sistemi crittografici della VPN garantiscono che i dati, nel tratto in cui attraversano reti pubbliche, siano sicuri e indecifrabili.

Nozioni di base
Protocollo

- Per poter comunicare in rete è necessario stabilire delle regole e delle convenzioni, nonché uno o più linguaggi comuni.
- Un protocollo (di rete) è **un insieme di regole e convenzioni che devono essere seguite per connettersi e comunicare in rete**.

Nozioni di base
Internet e Intranet

- **Internet** (con la “I” maiuscola) è l’insieme *globale* di tutte le reti connesse nel mondo, che usa il protocollo TCP/IP per la comunicazione.
- La **Intranet** di una certa organizzazione è l’insieme di tutte le reti appartenenti a tale organizzazione. Le Intranet usano gli stessi protocolli di Internet, ma adottano speciali misure per esporre all’esterno (cioè su Internet) solo una parte specifica delle loro risorse, mentre le restanti sono utilizzabili solo dall’interno della intranet stessa.

Nozioni di base
Internet Service Provider

- Mantenere un nodo connesso a Internet è complesso e costoso e solo grandi enti possono farlo.
- Gli utenti comuni si collegano invece a Internet attraverso gli **Internet Service Providers** (ISP).
- Un provider permette agli utenti di connettersi alla propria rete tramite apparati poco costosi (modem, router, ecc.) e linee dati facilmente disponibili ovunque (linee fonia e ora linee in fibra ottica).
- La rete del provider è opportunamente aggregata a Internet, quindi tramite questa l'utente può essere anch'esso connesso a Internet.
- Gli ISP fungono quindi da passaggio obbligato per tutte le informazioni scambiate dall'utente su Internet e per questo possono, ad esempio, **limitarne le capacità di connessione** (filtraggio di particolari siti o protocolli, limitazione della velocità di connessione) e **analizzarne il traffico** a vario scopo, oltre che **tenere dei registri** più o meno dettagliati di tutta l'attività dell'utente sulla rete.

Nozioni di base
Identificatori di risorse, URI e URL

- Per identificare univocamente le risorse, anche al di fuori della rete, vengono comunemente utilizzati gli "**Uniform Resource Identifier**" (URI):

  `<schema>://<autorità>/<path>?<query>#<frammento>`

  - Lo *schema*, obbligatorio, fornisce informazioni sul formato e il significato del resto della URI
  - L'*autorità*, se presente, indica l'organizzazione a cui appartiene il nome definito dal resto della URI
  - Il *path*, obbligatorio, fornisce la prima parte del nome univoco (eventualmente rispetto all'autorità) della risorsa, rappresentato in maniera gerarchica
  - La *query*, se presente, fornisce un'ulteriore parte del nome, questa volta sotto forma di dati non gerarchici
  - Il *frammento*, se presente, permette di puntare a una sotto-risorsa della risorsa principale
- Un "**Uniform Resource Locator**" (URL) è un tipo specifico di URI che non solo identifica la risorsa, ma indica dove si trova e/o come accedervi.

  `<protocollo>://<host>:<porta>/<path>?query#<frammento>`

  - In una URL, lo schema identifica il **protocollo** di accesso alla risorsa (http, ftp, …)
  - L'autorità è costituita dal FQDN dell'**host** che ospita la risorsa (o dal suo indirizzo IP) eventualmente seguito dal numero di **porta** che identifica il servizio logico a cui connettersi sull'host (se omessa, ha un default dipendente dal protocollo)
  - Il **path** è una sequenza di stringhe separate da slash (/) che, come in un filesystem, identifica una risorsa attraverso il percorso logico necessario a raggiungerla (nel caso del filesystem, come raggiungere un file attraversando la gerarchia cartelle).

Nozioni di base
Rappresentazione delle risorse

- Le risorse scambiate su una rete devono essere adeguatamente **rappresentate** per poter essere comprensibili dai dispositivi che stanno dialogando.
  - Ad esempio, quando si trasmette un'immagine non si sta trasmettendo veramente l'immagine, ma una rappresentazione di quest'ultima, codificata ad esempio tramite il formato JPEG.
- Perché una comunicazione possa essere efficace, le due parti devono concordare, oltre che sul protocollo, anche sui formati da utilizzare per scambiare le risorse.
- Su Internet sono presenti una gran quantità di **formati standard** per la rappresentazione di numerosi classi di risorse, come ad esempio l'HTML per gli ipertesti, JPEG, PNG, GIF, BMP, ecc. per le immagini, vari derivati dell'MPEG per audio e video, ecc.
- Questi tipi sono indicati tramite i cosiddetti **media types** (noti anche come tipi **MIME**), che hanno la forma generica «*tipo/sottotipo*», ad esempio text/html, image/jpeg, audio/mpeg.

Nozioni di base
World Wide Web

- Il World Wide Web, insieme alla posta elettronica, è il modo più diffuso di utilizzare Internet (ma non è l'unica applicazione costruita su Internet!)
- Il Web
  - Si basa sul protocollo **HTTP**.
  - Permette di accedere e manipolare risorse tramite la loro **URL** o URI.
  - Permette lo scambio di risorse codificate con i più comuni media types, tra cui il principale è l'**HTML** (text/html).
- L'accesso al web si effettua utilizzando particolari software chiamati **web server** e **web browser**.
- Tim Berners-Lee propose l'architettura del Web e realizzò i primi server e browser, nonché la prima pagina HTML, nel 1991, presso il laboratorio di fisica del CERN nel 1990.
- Tim Berners-Lee ha anche fondato il W3C (World Wide Web Consortium), che si occupa tuttora di standardizzare e sviluppare ulteriormente il Web.

Nozioni di base
Codifica del testo

- **Il Web è costituito soprattutto da testo**, per cui è importante capire come questo può essere rappresentato su una macchina.
- Un **set di caratteri** (*character set*, *charset*) definisce l'insieme di caratteri (lettere, numeri, punteggiatura, simboli,…) necessari per un particolare scopo (non necessariamente legato alla trasmissione e conservazione digitale).
- Un **set di caratteri codificato** è un set di caratteri a ciascuno dei quali è stato assegnato un numero univoco. Gli elementi di un set di caratteri codificato sono anche noti come **code points**. Un code point, quindi, rappresenta la posizione di un carattere nel set di caratteri codificato: ad esempio, il code point per la lettera «á» nel set di caratteri codificati Unicode è 225.
- La **codifica dei caratteri** (*character encoding*) definisce infine il modo in cui il set di caratteri codificato (o meglio i suoi code point numerici) verranno mappati su bytes per essere salvati su supporti digitali e trasmessi in rete.
  - Molti standard storici di codifica dei caratteri (ad esempio gli ISO 8859, ancora largamente diffusi) utilizzano un singolo byte per ogni code point che rappresenta semplicemente la posizione del carattere nel set. Ad esempio, la «A» nel set codificato ISO 8859-1 è il 65° carattere, ed è quindi codificata per la rappresentazione un byte con il valore di 65.
- Quando si comunicano informazioni testuali è quindi necessario specificare il set in uso e il corrispondente encoding, in modo che tali caratteri possano essere rappresentati in modo affidabile.

Nozioni di base
Unicode

- Lo **Unicode Consortium** ha definito un ampio set di caratteri che include tutti quelli necessari a qualsiasi sistema di scrittura nel mondo.
  - Lo standard Unicode è fondamentale per l'architettura del Web e dei sistemi operativi, e sta gradualmente sostituendo i set di caratteri specifici creati in passato da diverse organizzazioni, quali l'ISO.
  - I primi 65536 code points nel set di caratteri Unicode costituiscono il **Basic Multilingual Plane** (BMP), che include la maggior parte dei caratteri più comunemente usati. Sono inoltre disponibili circa un milione di code points aggiuntivi per **caratteri supplementari**, comprese le emoji.
- Sebbene il set di caratteri Unicode sia unico, le codifiche disponibili sono più di una: **UTF-8**, **UTF-16** e **UTF-32**.
  - UTF-8 utilizza **un byte** per rappresentare i caratteri nel set **ASCII**, **due byte** per i caratteri più comuni in altri alfabeti, **tre byte** per il resto del BMP e **quattro byte** per i caratteri supplementari.
  - UTF-16 utilizza 2 byte per qualsiasi carattere nel BMP e 4 byte per i caratteri supplementari.
  - UTF-32 utilizza 4 byte per tutti i caratteri.
  - Quindi, sebbene il code point per la lettera «á» nel set di caratteri Unicode sia sempre 225, in UTF-8 è rappresentato da due byte.
- In HTML, e in generale su Internet, attualmente si considera come **standard lo Unicode con codifica UTF-8.**
- Incredibilmente, persino l'uso malizioso dei *character encoding* Unicode può evidenziare **vulnerabilità** e permettere, ad esempio, il furto di informazioni.
  - **UTF-7, originariamente definito per codificare il solo BMP, permetteva codifiche alternative di caratteri ASCII come «<« e «>». Le vecchie versioni di Internet Explorer** potevano essere indotte a interpretare una pagina come UTF-7, e in questo caso le sequenze +ADw- e +AD4-, che le maggior parte dei validatori tratta come come testo semplice, venivano trasformate in «<» e «>» permettendo attacchi **XSS**.
  - Poiché **molti caratteri di lingue diverse possono somigliarsi visivamente**, è possibile indurre un utente a navigare su un sito il cui indirizzo è visivamente simile a quello di un sito sicuro, anche se in realtà i caratteri in esso contenuti sono diversi, e quindi porteranno a un sito malevolo (attacchi di **omografia**), ad esempio usando il carattere cirillico "a" (Unicode 0430) invece di quello ASCII (Unicode 0041)

Generalità sulla rete Internet

Protocolli, indirizzi, servizi

This work is licensed under CC BY-NC-SA 4.0. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-sa/4.0/

Come funziona la comunicazione in rete?
Il modello OSI

- Quando inviamo un pacchetto di dati sulla rete, questo viene manipolato e trasportato fino a destinazione da una serie di machine, o nodi della rete, tramite particolari protocolli, che si occupano di gestire, ad esempio, l'integrità del messaggio o il suo corretto istradamento.
- Per capire quindi veramente come funziona la comunicazione sulla rete, partiamo dal modello di riferimento, il modello OSI (*Open Systems Interconnection*), adottato fin dagli anni '80, e basato su sette livelli. L'Internet moderno non si basa sul modello OSI, ma su una sua semplificazione, il modello TCP/IP, che vedremo di seguito.
  - 7. Livello di **applicazione**. E' rappresentato dal software dell'utente che comunica attraverso la rete, come il browser web o il client ti posta elettronica. Si basa protocolli specifici quali HTTP (Hypertext Transfer Protocol), FTP (File Transfer Protocol), POP (Post Office Protocol), SMTP (Simple Mail Transfer Protocol), DNS (Domain Name System).
  - 6. Livello di **presentazione**. Prepara i dati per il livello di applicazione, definendo come codificare, crittografare e comprimere i dati in modo che vengano ricevuti correttamente dall'altra parte.
  - 5. Livello di **sessione**. Crea canali di comunicazione, chiamate sessioni, tra i dispositivi.
  - 4. Livello di **trasporto**. Esegue il controllo del flusso, inviando i dati a una velocità che corrisponde alla velocità di connessione del dispositivo ricevente, e il controllo degli errori, controllando se i dati sono stati ricevuti correttamente e, in caso contrario, reinviandoli. Inoltre, suddivide segmenti in pacchetti di rete all'origine e li riassembla alla destinazione.
  - 3. Livello di **rete**. Il suo compito principale è il routing, cioè la scelta del percorso migliore dalla sorgente alla destinazione (identificate tramite i loro indirizzi di rete) attraverso la rete fisica. Inoltre, distribuisce i pacchetti (generati dal livello precedente) in segmenti all'origine e li riassembla alla destinazione.
  - 2. Livello di **collegamento**. Gestisce il collegamento tra due nodi fisicamente. Suddivide i pacchetti dati in frame e li invia dall'origine alla destinazione, gestendo anche le situazioni di collisione.
  - 1. Livello **fisico**. Gestisce l'effettiva trasmissione «fisica» dei dati «grezzi» (binari) via cavo (ed esempio Ethernet) o via radio (WIFI)

Come funziona la comunicazione in rete?
Il modello TCP/IP

- Il modello TCP/IP (Transfer Control Protocol/Internet Protocol) è un modello reale (mentre quello OSI è solo di riferimento) creato dal Dipartimento della Difesa degli Stati Uniti ancor prima del modello OSI.
- *Si tratta del modello effettivamente utilizzato su Internet.*
- Rispetto ad OSI, questo modello è più semplice in quanto comprime diversi livelli in uno:
  - I livelli di applicazione, presentazione e sessione sono combinati in un unico livello *applicazione*. Questo significa che l'applicazione utente (ad esempio il browser web) dovrà occuparsi anche della corretta codifica dei dati (tipicamente usando **MIME** e sistemi di crittografia come **SSL/TLS**) e del mantenimento delle sessioni (solitamente con le cosiddette **Socket**).
  - Il livello di rete è gestito tipicamente dall'Internet Protocol (**IP**), anche se è possibile trovare a questo livello anche protocolli più specifici come Ipsec, ICMP, IGMP, OSPF, RIP.
  - Il livello di trasporto è gestito tipicamente dal Transfer Control Protocol (**TCP**) o dall'User Datagram Protocol (**UDP**).
  - I livelli fisico e di collegamento sono combinati in un unico livello di *accesso alla rete*. Il modello TCP/IP non si interessa del funzionamento fisico (è sufficiente che ci siano primitive per la trasmissione e ricezione di bit sulla rete fisica), mentre per la parte di collegamento viene utilizzato tipicamente il protocollo **PPP** (Point-to-Point protocol).

Come si identificano le macchine in rete?
Gli indirizzi IP

- L'Internet Protocol (IP), in quanto protocollo di livello rete, gestisce l'instradamento dei pacchetti dalla sorgente alla destinazione.
- A questo scopo, **a ogni dispositivo esposto direttamente sulla rete viene assegnato un indirizzo IP** che lo identifica in modo univoco.
  - Gli indirizzi IP sono spesso dinamici, nel senso che ad ogni riconnessione alla rete a un dispositivo può essere assegnato un indirizzo IP diverso (ad esempio tramite il DHCP).
  - Tuttavia, **l'internet provider conosce sempre l'associazione tra l'indirizzo IP e l'utente dell'intestatario della linea a cui tale IP è assegnato**.
  - Tramite l'IP è spesso **possibile risalire anche alla posizione geografica approssimativa del dispositivo stesso**.
  - Tuttavia, se si utilizza una **VPN**, l'indirizzo IP collegato ai propri pacchetti sulla rete sarà quello generico di un gateway della VPN, mentre il reale indirizzo della macchina sorgente sarà noto solo alla VPN stessa.

Come si identificano le macchine in rete?
IPv4 e IPv6

- Esistono attualmente due tipi di indirizzi IP:
  - **L'IPv4**, quello più noto, utilizza un indirizzo a 32 bit, potendo indirizzare circa **4 miliardi** di dispositivi (alcuni blocchi IP sono riservati ad usi speciali). Viene solitamente rappresentato come una sequenza di quattro byte separati da punti, ad esempio **192.168.130.4**
  - **L'IPv6** è le versione più moderna dell'IP. E' formato da 128 bit, quindi fornisce circa **$3{,}4 \times 10^{38}$** indirizzi distinti. Viene rappresentato usando otto gruppi di quattro cifre esadecimali separate da due punti, ad esempio **2001:0db8:85a3:0000:1319:8a2e:0370:7344**
- L'IPv6 si è reso effettivamente necessario perché gli IPv4 liberi sono ormai esauriti. Infatti, attualmente la maggior parte dei dispositivi connessi in rete utilizza la Network Address Translation (NAT), grazie alla quale dei sottoinsiemi di dispositivi vengono indentificati sulla rete con un unico indirizzo IP, che viene poi mappato su un indirizzo IP «privato» interno alla sottorete, impedendo la vera connessione end-to-end tra macchine e rendendone in certi casi più difficile l'identificazione. IPv6 elimina la necessità di NAT e permette a tutti i dispositivi di essere effettivamente presenti sulla rete in modo diretto.
- Inoltre, IPv6 integra
  - Meccanismi di Quality of Service (QoS)
  - Meccanismi di sicurezza di rete (IPsec).
  - Meccanismi di multicasting (trasmissione di un pacchetto a più destinazioni in un'unica operazione), che era opzionale in IPv4.

Come si assegna un nome a una macchina in rete?
Il servizio DNS

- Nella pratica quotidiana, ad esempio quando si vuole contattare un sito web installato su una macchina in rete, è poco pratico usare il suo indirizzo IP. Per questo sono stati introdotti i **fully qualified doman names** (FQDN, ad esempio www.univaq.it), o nomi simbolici per i dispositivi connessi alla rete. Questi nomi sono univoci e composti da un *hostname* (www) e da un *nome di dominio* (univaq.it).
- Il **servizio DNS** costituisce la "guida telefonica" della rete: gli utenti identificano le macchine con un FQDN e l'applicazione, ad esempio il browser web, utilizza il DNS per **risolvere l'indirizzo IP** associato, col quale stabilire la connessione effettiva.
- Il DNS è realizzato come un *database distribuito*, costituito dai *server* DNS, con una precisa struttura gerarchica.
  - Ogni macchina presente in rete deve «conoscere» l'indirizzo IP di almeno un server DNS detto *resolver* o *recursor*, che ha lo scopo di istradare le ricerche nel modo corretto nella gerarchia dei DNS.
  - La ricerca viene avviata a partire dai *root nameserver*, che la smistano a un *server di dominio di primo livello* (TLD) sulla base dell'ultimo segmento del nome di dominio (ad esempio .it). I root nameserver sono 13 in tutto il mondo, di cui 10 negli Stati Uniti, due in Europa (Inghilterra e Svezia) e uno in Giappone.
  - Il resolver passa la ricerca al TLD indicato dal root nameserver, che a sua volta restituisce l'indirizzo del *server autoritativo* specifico per il dominio richiesto (univaq.it).
  - Il server autoritativo di dominio, infine, interrogato dal resolver, restituisce l'indirizzo IP effettivo relativo al FQDN richiesto, che viene poi passato dal resolver all'applicativo che ne ha fatto richiesta.
- A vari livelli (resolver, sistema operativo dell'utente) è disponibile inoltre di una cache che memorizza i risultati delle ultime ricerche e permette di restituire un risultato immediato senza dover ogni volta percorrere l'intera gerarchia.

La rete Internet
Considerazioni di sicurezza

- La struttura base di Internet espone tutta una serie di meta-informazioni che possono essere utili per attività lecite o malevole.
- Gli indirizzi IP possono essere usati per **identificare i soggetti connessi alla rete**, ma anche per attaccarli o **sostituirsi a loro** (*IP sproofing*) svolgendo poi attività illecite.
- I nomi simbolici attribuiti agli host, utili per poterli raggiungere più agevolmente, possono essere usati per indurre l'utente ad accedere a sistemi malevoli (ad esempio con attacchi di *omografia*).
- I server DNS, che sono il cardine del sistema di traduzione dei nomi simbolici in indirizzi IP, possono essere usati per **bloccare lecitamente l'accesso a certi siti** (rimuovendone l'IP dal DNS), **impedire la normale navigazione** bloccando i server (DNS DoS, DNS flooding) ma anche per **deviare richieste lecite su siti fraudolenti** (DNS hijacking),

I protocolli del Web

HTTP e HTTPS

This work is licensed under CC BY-NC-SA 4.0. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-sa/4.0/

L'Hypertext Transfer Protocol (HTTP)
Il protocollo del Web

- HTTP (Hypertext Transfer Protocol) è un protocollo testuale **usato per la comunicazione tra il browser e i server** che ospitano le applicazioni web.
- L'HTTP è un cosiddetto protocollo **client-server unidirezionale**. Questo significa che nella comunicazione c'è un lato che fornisce il servizio (il server) a seguito della richiesta di un altro lato (il client), e questi ruoli restano fissi nel tempo:
  - **Il server non può mai prendere l'iniziativa e parlare col client senza venir interpellato** da quest'ultimo
  - **I client non possono parlare tra di loro** (*peer-to-peer*), se non tramite un server intermediario
  - Attualmente, siamo in una fase di transizione in cui l'avvento di nuovi standard come le Web Sockets e WebRTC permetterà l'eliminazione di molti di questi vincoli, aprendo le porte a una vastità di nuove applicazioni eseguibili sui browser, ma anche a moltissimi nuovi pericoli per gli utenti.
- L'HTTP, per la sua semplicità e versatilità, è diventato negli ultimi anni **il protocollo più utilizzato anche per una vastità di altre applicazioni che usano il Web** come vettore, ma non sono propriamente esposte all'utente finale sotto forma di siti, come ad esempio le Web API. Per questo è molto importante conoscerlo.
- Attualmente sono note **quattro versioni** di HTTP:
  - **HTTP/1.0:** implementata da Berners-Lee nel 1991.
  - **HTTP/1.1**: estensione del protocollo base, sviluppata tra il 1997 e il 1999, attualmente supportata ed utilizzata da tutti i browser
  - **HTTP/2**: prima versione «moderna», pubblicata nel 2015 e ormai supportata da quasi tutti i browser e i server moderni.
  - **HTTP/3**: la versione più avanzata dell'HTTP pubblicata il 6 giugno 2022.

L'Hypertext Transfer Protocol (HTTP)
Connessione

- HTTP è un protocollo applicativo quindi, su Internet, è posto al di sopra di tutti gli altri livello dello stack TCP/IP.
- HTTP è un protocollo stateless, cioè non offre strumenti alle parti coinvolte per mettere in relazione diverse richieste/risposte memorizzando informazioni di stato. Questo vuol semplicemente dire che tutti i dati relativi a una richiesta devono essere contenuti nella richiesta stessa: non si possono inviare dati correlati in richieste successive.
- Quando un client invia una richiesta HTTP al server, viene per prima cosa aperta una connessione TCP/IP tra il client e il server. A differenza di molti altri protocolli (come FTP), tale connessione viene generalmente chiusa una volta che la relativa richiesta è stata soddisfatta.
  - Questo comportamento rende il protocollo HTTP ideale per il Web, in cui ogni pagina può richiedere il caricamento di decine di risorse anche da sorgenti diverse, in quanto limita il numero di connessioni attive limitandole a quelle effettivamente necessarie.
- Tuttavia, per ottimizzare l'uso delle connessione quando si devono scaricare più risorse correlate dallo stesso server, in HTTP 1.0 fu introdotta l'estensione non ufficiale «http keep-alive»:
  - Il client inserisce nella richiesta l'intestazione «connection: keep-alive». Un server che supporta la connessione keep-alive aggiungerà a sua volta alla risposta questa intestazione e non chiuderà la connessione, che potrà essere riutilizzata finchè una delle due parti deciderà di chiuderla segnalandolo all'altra con l'intestazione «connection: close».
  - A partire da HTTP 1.1, tutte le connessioni sono considerate persistenti se non diversamente dichiarato.
- I server stabiliscono sempre un timeout di connessione (solitamente poche decine di secondi), dopo il quale quest'ultima, se non è in uso, viene chiusa, per evitare problemi di sovraccarico sui server (anche intenzionali, ad esempio per attacchi DoS).

L'Hypertext Transfer Protocol (HTTP)
Richiesta

- Ogni messaggio di richiesta HTTP contiene almeno la versione del protocollo in uso, un metodo e la URL della risorsa richiesta (su cui effettuare l'operazione data dal metodo). Possono poi essere incluse una serie di intestazioni e un payload, cioè un insieme di dati da associare alla richiesta.
  - Tutte queste informazioni, compreso il payload, sono codificate in forma di testo: si usa ASCII come set di caratteri per tutti gli elementi, tranne il payload, che può basarsi su un qualsiasi charset UNICODE (adeguatamente dichiarato).

```
GET /section.php HTTP/1.1
Accept: text/html,application/xhtml+xml,application/xml
Accept-Encoding: gzip, deflate, br
Accept-Language: it-IT,it;q=0.8,en-US;q=0.5,en;q=0.3
Connection: keep-alive
Cookie: …
DNT: 1
Host: www.univaq.it
Referer: …
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:102.0)
Gecko/20100101 Firefox/102.0
```

- La prima riga della richiesta contiene la **URL** della risorsa richiesta (/section.php, relativa al server contattato, il cui nome completo è riportato nell'intestazione **Host**: www.univaq.it) , il **metodo** di accesso (**GET**) e la **versione** del protocollo (1.1)
- Le prime intestazioni definiscono le modalità di invio dell'eventuale payload (**Content-Type**, **Content-Length**, ecc., qui non presenti) e di ricezione della risposta: formati accettati (**Accept**), codifiche accettabili (**Accept-Encoding**), lingue accettabili (**Accept-Language**)
- Le altre intestazioni (che non sono, tuttavia, tutte obbligatorie) contengono informazioni utili (*spesso anche per scopi illeciti*!) quali i **Cookie** noti al client e relative al server contattato, l'indirizzo della pagina dalla quale si è avviata questa richiesta (**Referer**), le caratteristiche del browser in uso (**User-Agent**) e il metodo di autenticazione, nel caso di risorse protette tramite schemi di protezione HTTP (**Authorization**).

L'Hypertext Transfer Protocol (HTTP)
Risposta

› Ogni risposta HTTP contiene almeno la versione del protocollo e un codice di stato:

```
HTTP/1.1 200 OK
Date: Tue, 12 Jul 2022 09:16:45 GMT
Server: Apache
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Set-Cookie: lang=en; expires=Thu, 11-Aug-2022 09:16:45 GMT; path=/
Vary: Accept-Encoding
Content-Encoding: gzip
X-Frame-Options: SAMEORIGIN
Content-Length: 11793
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=ISO-8859-1
<!DOCTYPE html> <html> <head> <title> … </title> </head> <body> … </body>
</html>
```

› La prima riga della risposta indica la versione di HTTP in uso (1.1) e lo stato della risposta (200 OK). Esistono numerosi codici di stato utili a segnalare diverse situazioni di errore e modalità di risposta.

› Nel caso di risposta con payload (cioè contenuto, in questo caso la parte HTML che segue le intestazioni), le intestazioni **Content-Type**, **Content-Length** e **Content-Encoding** ne specificano le caratteristiche (formato text/html, codifica caratteri ISO-8859-1, compressione gzip, lunghezza totale 11793 bytes)

› Anche in questo caso, ci sono altre intestazioni utili a documentare meglio la risposta, che possono essere usate per trarre informazioni utili all'analisi del traffico, come i **Set-Cookie**, che imposta un cookie richiesto dal server sul client, il **Server** che ha generato la risposta, la data di ultima modifica della risorsa, quali intestazioni della richiesta sono state usate per selezionare la rappresentazione della risposta (**Vary**), come gestire il caching della risorsa nel browser (**Cache-Control**), ecc.

HTTP sicuro: HTTPS

- Tecnicamente, HTTPS non è un vero protocollo: **si tratta solo di HTTP agganciato al sistema di crittografia SSL/TLS**, come specificato nell'architettura TCP/IP.
- I siti web che installano e configurano un **certificato SSL/TLS** possono utilizzare il protocollo HTTPS per stabilire connessioni *sicure*.
- Nelle URL la prima parte, seguita dai due punti, è il nome del protocollo di comunicazione: se questo è HTTPS, allora la comunicazione è attualmente protetta tramite SSL/TLS.
- Molti siti permettono (ancora) **solo connessioni insicure HTTP**, oppure permettono di **connettersi indiscriminatamente con HTTP o HTTPS**, o ancora utilizzano **HTTP per scaricare parti delle dipendenze** (ad esempio script o immagini) **richieste da una risorsa attenuta via HTTPS** (una pagina HTML). **Tutte queste pratiche sono molto pericolose**, in quanto diminuiscono o annullano o la sicurezza della connessione, e l'utente non è spesso in grado di rendersene conto.

SSL e la protezione delle connessioni sul Web

- **SSL** è l'acronimo di **Secure Sockets Layer**. SSL è utilizzato per la comunicazione crittografata tra un sito Web e un browser. La tecnologia SSL è attualmente deprecata ed è stata interamente sostituita da **TLS (Transport Layer Security)**, che ne incorpora le caratteristiche. L'obiettivo di queste tecnologie è quello di rendere sicura la trasmissione di informazioni sensibili, inclusi dati personali su una connessione internet (**non necessariamente Web/HTTP**).
- Tecnicamente, TLS utilizza una combinazione di **crittografia** simmetrica e asimmetrica: con la crittografia **simmetrica**, i dati vengono crittografati e decrittografati con una chiave segreta nota sia al mittente che al destinatario, mentre con la crittografia **asimmetrica** si usano coppie di chiavi, una *pubblica* e una *privata*, che devono essere usate in maniera alternata per crittografare decrittografare: ad esempio il mittente può utilizzare la chiave pubblica del destinatario per crittografare i dati che gli invierà, e tali dati potranno essere decrittografati solo con la chiave privata del destinatario.
  - In particolare, per questioni di efficienza, TLS utilizza la crittografia asimmetrica in una fase iniziale, detta *handshaking*, per generare e scambiare in modo sicuro una **chiave simmetrica di sessione**, che viene poi usata per crittografare i dati trasmessi durante la connessione vera e propria.
- Il funzionamento di SSL/TLS si basa su **certificati di sicurezza** TLS (X.509) emessi da una terza parte nota come **autorità di certificazione** (CA), che contengono una chiave pubblica e informazioni sul dominio da essa protetto e sul soggetto che lo controlla.
- I browser solitamente mostrano **un'icona a forma di lucchetto** in presenza di un certificato SSL/TLS valido relativo al dominio a cui si è connessi.

SSL e la protezione delle connessioni sul Web
Certificati e Autorità di Certificazione

- I certificati di sicurezza contengono la chiave pubblica del proprietario e certificano almeno che quest'ultimo controlli il dominio protetto dal certificato (e quindi i siti web in esso ospitati).
- I certificati emessi dalle CA vengono a loro volta convalidati (firmati tramite chiave privata, quindi verificabili con la corrispondente chiave pubblica) da altre CA di livello superiore, formando una cosiddetta *chain of trust* che termina con una **CA radice**, considerata globalmente affidabile.
- I certificati (contenenti le chiavi pubbliche) delle CA radice sono normalmente **installati fisicamente, tramite canali sicuri, nei browser e nei sistemi operativi**.
- In alcuni casi, un server può utilizzare un certificato auto-emesso (**self-signed**) che deve essere esplicitamente considerato attendibile dal client, ma si tratta di una pratica sconsigliata.

SSL e la protezione delle connessioni sul Web
Il processo di handshaking

- La comunicazione tramite TLS si avvia con *l'handshake*.
- Il client invia un messaggio al server con le informazioni necessarie a determinare i parametri di crittografia appropriati per la connessione. Il server risponde quindi con un messaggio che indica i parametri di connessione negoziati.
- Il server invia quindi al client il proprio **certificato di sicurezza** (e qualsiasi altro certificato di supporto nella corrispondente *chain of trust*), seguito da una **firma** sul contenuto dell'intero handshake eseguita tramite la chiave privata corrispondente alla chiave pubblica contenuta nel certificato.
  - **Il server può richiedere al client di autenticarsi allo stesso modo**, e in tal caso anche il client dovrà inviare il proprio certificato e firmare questa comunicazione con la sua chiave privata. *Questo non avviene con la normale comunicazione tramite web browser.*
- A questo punto il **client genera una chiave di crittografia simmetrica (di sessione)** e la comunica al server cifrandola con la chiave pubblica presente nel certificato del server stesso.
- Il server decifrerà la chiave di sessione, e tutto il resto della comunicazione si svolgerà in maniera sicura **cifrando i dati con la chiave di sessione**.

I Protocolli del Web
Considerazioni di sicurezza

- La sicurezza offerta da HTTPS, cioè HTTP su SSL/TLS, è tutt'altro che assoluta.
- Un primo punto debole di questo sistema risiede nel fatto che un certificato CA malevolo considerato attendibile per errore può convalidare l'identità di un numero infinito di domini a loro volta malevoli, mettendo in pericolo l'utente.
  - E' quindi importante installare nel proprio browser e sistema operativo solo certificati veramente attendibili.
- Inoltre, sebbene SSL/TLS sia concettualmente inattaccabile, qualsiasi difetto nell'implementazione del su software di supporto può portare a rischi di sicurezza su larga scala.
  - Questo ad esempio è accaduto nel 2014 con il bug hearthbleed di OpenSSL (CVE-2014-0160)
- Infine, un altro punto debole del sistema è che un certificato base (DV, **Domain Validated**) assicura solo che il soggetto *controlli* il dominio specificato: la convalida viene in genere eseguita con tramite una risorsa di validazione che il soggetto è tenuto a rendere disponibile nel dominio, provando così di averne il controllo. Questo però non assicura nulla riguardo al fatto che il soggetto effettivamente sia il *proprietario* del dominio (potrebbe averne preso il controllo in maniera malevola).
  - Esistono tuttavia certificati cosiddetti OV (**Organisation Validated**) ed EV (**Extended Validation**), nei quali viene garantita anche l'identità effettiva del proprietario del dominio protetto dal certificato: con i certificati OV, l'entità richiedente è soggetta a controlli aggiuntivi come la conferma del nome dell'organizzazione, dell'indirizzo e del numero di telefono, utilizzando banche dati pubbliche. I certificati EV includono controlli ancora più dettagliati.
  - I browser normalmente visualizzano il nome dell'organizzazione in verde quando viene rilevato un certificato EV valido.

Il Web lato client

HTML, CSS e Javascript

This work is licensed under CC BY-NC-SA 4.0. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-sa/4.0/

HyperText Markup Language
La parte strutturale del web

- L'HyperText Markup Language (HTML) è l'elemento costitutivo più basilare del Web, e viene usato **per definire la struttura dei suoi contenuti**. L'HTML permette di definire **ipertesti**, cioè testi che sono connessi tra loro tramite collegamenti, o link.
- Tecnicamente, HTML è un **linguaggio di markup**, in cui cioè il cui contenuto testuale è strutturato tramite particolari delimitatori detti **tag**, aventi la generica sintassi «<tag>». Esistono tag di *apertura* (<tag>) e *chiusura* (</tag>), esattamente come le parentesi aperte «(» e chiuse «)» nel linguaggio comune.
- Il testo delimitato da un tag viene chiamato **elemento**. Il tag di ogni elemento fornisce la **semantica** del suo contenuto, indicando come questo debba essere interpretato (ma non rappresentato visivamente: a questo pensano i fogli di stile CSS). I tag quindi non vengono quindi visualizzati nei browser, ma li istruiscono su come mostrare il contenuto della pagina web.
- HTML (come la maggior parte dei linguaggi di markup) è facilmente manipolabile dalle macchine ma anche leggibile dagli esseri umani.
- HTML è inoltre totalmente **indipendente da architetture hardware, sistemi operativi e protocolli**, ed è addirittura **utilizzato al di fuori del Web** come formato alternativo per testo strutturato simile a quello generato dai word processor, ad esempio nelle email e persino per la definizione delle interfacce utente del software.

HyperText Markup Language
Evoluzione

- Esistono numerose versioni di HTML, che si è evoluto seguendo le esigenze del web ma anche di tutte le altre applicazioni che ne fanno uso:
  - 1991 Prima versione (HTML 1.0) definita da Tim Berners-Lee
  - 1995 HTML 2.0 (definita dall'HTML Working Group)
  - 1997 HTML 3.2 (W3C)
  - 1999 HTML 4.01 (W3C)
  - 2000 XHTML 1.0 (HTML 4.01 con sintassi XML)
  - 2008 Draft HTML5 (Web Hypertext Application Technology Working Group)
  - 2012 HTML5 Living Standard (WHATWG)
  - 2014 HTML5 (W3C)
  - 2017 HTML5.2 (W3C)
  - …

HyperText Markup Language
Elementi di base

- Un documento HTML ha in generale la forma che segue:

```
<!doctype html>
<html>
 <head>
  <meta charset="UTF-8">
  <title>…</title>
 </head>
 <body> … </body>
</html>
```

- La prima riga (<!doctype html>) **dichiara che il documento è scritto con HTML** e in particolare con la versione 5 del linguaggio.
- L'elemento <html> **contiene l'intero documento**: al suo interno
  - L'elemento <head> contiene informazioni di configurazione (**meta informazioni**) per il documento, quali ad esempio
    - Il **titolo** del documento (<title>, obbligatorio)
    - Il **charater set e il relativo encoding** usati nel documento (in questo caso UTF-8)
    - Lo **stile** da usare per la rappresentazione visiva del contenuto del documento (<style>)
    - La **logica** di controllo del documento, programmata tramite script Javascript (<script>)
  - L'elemento <body>, infine, **racchiude il contenuto vero e proprio** del documento.

HyperText Markup Language
Elementi di base

- Il contenuto racchiuso nel <body> è strutturato logicamente usando numerosi altri tag:
  - Contenuto *Phrasing* (elementi di base): , <img>,, ecc.
  - Contenuto *Heading* (intestazione): <h1>, <h2>, ecc.
  - Contenuto *Sectioning* (sezionamento): <aside>, <section>, ecc.
  - Contenuto *Flow* (flusso): , , ecc.
  - Contenuto *Embedded* (incorporato): <img>, <iframe>, <svg>, ecc.
  - Contenuto *Interactive* (interattivo): <a>, <button>, <label>, ecc.

```
<body>
<section><h1>Titolo sezione</h1>
<p>Paragrafo <em>enfatizzato</em></p>
<p><img src="http://www.univaq.it/img/logo.png"/></p>
</section>
</body>
```

- Alcuni di questi tag permettono di **includere nella pagina elementi non testuali**, quali immagini (<img>), controlli di input (<input>), bottoni (<button>), ecc.

Cascading Style Sheets
La parte visuale del web (e non solo)

- I Cascading Style Sheets (CSS) sono un linguaggio utilizzato per **definire l'aspetto visivo/percettivo (*presentazione*) dei documenti HTML** .
- CSS descrive come gli elementi definiti da HTML debbano essere rappresentati su **schermo**, su **carta**, nel **parlato** o su altri supporti.
  - Su media visivi, come gli schermi o la carta, CSS può essere usato ad esempio per definire il tipo di carattere, il colore, le dimensioni e la spaziatura del contenuto, dividerlo in colonne o aggiungere animazioni e altre caratteristiche decorative.
  - Su media diversi, come il parlato, CSS può essere usato per definire il tono e la velocità della voce usata per pronunciare un contenuto.
- A partire dalla versione 3, i CSS sono diventati molto più potenti e il loro sviluppo è continuo.
- I CSS (o dei loro derivati) possono essere usati anche per definire aspetti visivi al di fuori del web, ad esempio per le interfacce grafiche delle applicazioni desktop.

Cascading Style Sheets
Elementi di base

- CSS è un linguaggio basato su **regole**.
  - Ogni regola di stile inizia con un **selettore** che identifica gli elementi ai quali i corrispondenti stili devono essere applicati.
  - Il contenuto della regola è composto da una semplice specifica che assegna valori alle **proprietà doi stile** degli elementi selezionati.
- Esistono estensioni CSS che permettono di dichiarare **variabili** ed effettuare **calcoli** nelle regole di stile, nonché di attivare selettivamente alcune regole in base alle caratteristiche del dispositivo di output (**media queries**), per adattare la visualizzazione dinamicamente ai dispositivi.

```
p {color: black; font-size: 12pt; }
```

- La regola di esempio colora il testo tutti i paragrafi (elementi p di HTML) in nero (color: black) e imposta a 12 punti la dimensione dei loro caratteri (font-size: 12pt).

Programmazione sul web
Server e Client

- Nel contesto web, si parla spesso di programmazione **lato server** e **lato client**.
- Il codice lato server viene eseguito **sull'host che ospita l'applicazione web** e dalla quale il browser scarica le pagine HTML.
  - Tale codice viene in genere utilizzato per generare le pagine HTML dinamicamente, eseguire operazioni lunghe e complesse e gestire la persistenza dei dati dell'applicazione.
  - Esempi di linguaggi web lato server sono Java, PHP, Python, Ruby, ASP.NET e anche Javascript
- Il codice lato client viene eseguito **sul browser dell'utente**: quando viene scaricata una pagina HTML, il codice lato client della pagina viene scaricato insieme ad essa ed eseguito.
  - Tale codice viene in genere utilizzato per gestire l'interazione con l'utente sull'interfaccia, modificare dinamicamente la pagina visualizzata, e spesso anche per eseguire elaborazioni di media complessità senza delegarle al server.
  - L'unico linguaggio lato client attualmente in uso è Javascript.

Javascript
Il linguaggio di programmazione per i browser (e non solo)

- Javascript (JS) è un **linguaggio di programmazione** usato comunemente per fornire dinamica alle pagine web.
  - Tecnicamente, si tratta di un linguaggio *interpretato* basato su *prototipi* che supporta vari stili di programmazione: *object oriented*, *imperativo* e *dichiarativo*.
- Javascript è definito sulla base della specifica del linguaggio **ECMAScript** (ultima edizione: ECMA-262)
- **Javascript non è Java**: i due linguaggi hanno sintassi, semantica e uso molto diversi.

Javascript
Elementi di base

- JavaScript dispone di tutte le funzionalità e i costrutti comuni ai più diffusi **linguaggi di programmazione imperativi** (variabili, cicli, istruzioni condizionali, istruzioni di assegnamento, operazioni matematiche di base, funzioni e procedure, ecc.). **In questo senso, ha un potere universale**.
- Javascript è progettato per essere eseguito all'interno di un **oggetto host**, o *ambiente*, che può estenderlo fornendo funzionalità aggiuntive specifiche per quell'ambiente.
  - Ad esempio, in una pagina web l'ambiente, che è il browser, fornisce a Javascript i mezzi per **leggere e modificare la struttura HTML della pagina** e i suoi fogli di stile (**DOM**), nonché per **interagire con alcune parti del browser** stesso, **intercettare tutte le azioni compiute dall'utente** sulla pagina ed eseguire chiamate verso altri server (**AJAX**)
    - Tuttavia, il browser costruisce una «**sandbox**» intorno alla pagina web, in modo che gli script in essa contenuti non possano interferire con le altre pagine aperte né con il resto della macchina.
  - I browser forniscono a Javascript anche l'accesso a una serie di funzionalità estese, le cosiddette **Web API**. Con queste ultime il «potere» di Javascript, seppur eseguito nella sandbox, si estende notevolmente, e quindi aumentano le possibilità di sfruttarlo in maniera malevola.
    - Alcuni esempi di Web API, il cui nome è piuttosto auto esplicativo: Battery API, Bluetooth API, Canvas API, Clipboard API, Console API, Credential Management API, Fetch API, File System Access API, Fullscreen API, Geolocation API, History API, Payment Request API, Picture-in-Picture API, Screen Capture API, Sensor API, Storage Access API, Web Audio API, Web Authentication API, Web Crypto API, Web Speech API, Web Storage API, WebGL, WebRTC.

Javascript
Dove si usa?

- Javascript è un linguaggio molto popolare tra i programmatori, per la sua versatilità e semplicità, e per questo si è esteso ben oltre la programmazione web lato client. In effetti, oggi Javascript viene usato per programmare una moltitudine di applicazioni ampiamente diffuse.
  - **Sviluppo web lato client**: come sappiamo, JavaScript viene usato insieme a HTML e CSS per creare le parti di una pagina Web che gli utenti vedono e con cui interagiscono nei loro browser.
  - **Sviluppo web lato server**: grazie all'avvento di **Node.js**, un framework Javascript comunemente usato per lo sviluppo back-end, Javascript può essere usato anche per la programmazione lato server.
  - **Sviluppo di giochi**: questa categoria, sebbene ricada propriamente in quella dello sviluppo lato client, è molto importante e va evidenziata. In precedenza, moltissimi giochi online erano sviluppati con tecnologie diverse e/o proprietarie come Flash. Oggi, Javascript è usato per programmare tutti giochi 2D e 3D che si trovano online.
  - **Sviluppo di applicazioni mobili**: le applicazioni per dispositivi mobili inizialmente erano sviluppate in maniera nativa, cioè interfacciandosi direttamente con il sistema operativo (Android, iOS) del dispositivo ed usando il linguaggio di programmazione da esso preferito. Attualmente, framework come **React Native** e **Ionic**, permettono di realizzare applicazioni mobili multipiattaforma usando gli stessi linguaggi del web: HTML, CSS e naturalmente Javascript.
  - **Sviluppo di applicazioni desktop**: seguendo l'esempio delle applicazioni mobili, si stanno affermando numerosi framework, come **Electron**, che permettono di sviluppare applicazioni per computer desktop usando Javascript (e spesso HTML e CSS per definirne l'interfaccia) invece dei più comuni linguaggi come Java o C++.
  - Javascript può essere usato per aggiungere dinamica anche ai documenti **PDF**!

Javascript
Nelle pagine web

- Javascript può essere inserito in una pagina HTML in vari modi.
  - Gli script possono essere **incorporati** all'interno dell'HTML usando l'elemento <script>

```
<script>
  …codice…
</script>
```

  - Gli script possono essere agganciati direttamente a particolari **eventi** inserendoli all'interno di specifici attributi di alcuni elementi HTML:

```
<button onclick="…codice…">Clicca qui</button>
```

  - Script esterni alla pagina possono essere **scaricati** ed eseguiti usando una variante dell'elemento <script>

```
<script src="programma.js"></script>
```

- Tutte queste possibilità offrono una versatilità estrema ma, come al solito, anche molti possibili vettori di attacco, come ad esempio il **Cross-Site Scripting**, in cui uno script malevolo viene iniettato nell'HTML e quindi eseguito dal browser dell'utente.

Il Web Lato Client
Considerazioni di sicurezza

- **HTML, di per sé, non pone particolari rischi di sicurezza**, in quanto è solo un modo statico per rappresentare informazione strutturata.
- **I fogli di stile CSS** sono anch'essi relativamente innocui, sebbene siano stati definiti attacchi che li sfruttano per violare la privacy degli utenti:
  - Ad esempio, è possibile **dedurre se l'utente ha visitato certi siti e comunicarlo a un host remoto** definendo una speciale regola di stile che marca visivamente i link che l'utente ha già visitato. Se tale regola, a sua volta, riesce a contattare un host esterno quando viene invocata, l'attaccante potrà sapere se l'utente ha visitato o no il sito a cui la regola di stile è stata applicata, oppure (usando uno script Javascript) analizzare tutti i siti visitati dall'utente (identificando i link a cui la regola è stata applicata):
    ```
    a#linkdamonitorare:visited{background-image:url(sitomalevolo)}
    ```
- Javascript, invece, **è alla base della maggior parte delle vulnerabilità del web lato client**, nonché di una serie di vulnerabilità note in tutti i sistemi che lo utilizzano come linguaggio di scripting.
- Poiché HTML è progettato per ospitare al suo interno Javascript, **un documento HTML non opportunamente filtrato può essere origine di attacchi** anche in contesti in cui ci si aspetta che questo venga usato per veicolare informazione pura (ad esempio le email), ma viene interpretato da un motore di *rendering* standard *script-aware*.

Surface Web, Deep Web e Dark Web

Le tre parti dell'iceberg Web

This work is licensed under CC BY-NC-SA 4.0. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-sa/4.0/

Surface Web

- Le persone solitamente si affidano ai motori di ricerca per ottenere le informazioni (e trovare i siti) di cui hanno bisogno.
- Le risorse così individuabili costituiscono il «surface web».
- Tuttavia, è noto che i motori di ricerca indicizzano solo l'1% delle risorse disponibili effettivamente sul web, quindi il surface web è una frazione minima di quanto esiste in rete.

Deep Web

- Il *deep web* costituisce il 99% delle risorse online che rimangono nascoste (anche se non necessariamente inaccessibili) al normale utente che utilizza i motori di ricerca.
- Tipicamente, **sono necessari permessi, credenziali o software particolari per accedere al deep web**, che per questo non può venire indicizzato dai motori di ricerca.
- Il deep web, in effetti, non è di principio nulla di strano o minaccioso: le informazioni bancarie di un utente (non i siti pubblici delle banche), i documenti della PA (non quelli pubblici) relativi ai cittadini, e anche le sezioni amministrative dei siti (per accedere alle quali è necessario effettuare una login) sono tutti parte del deep web in quanto sono disponibili sul web ma, ovviamente, non sono pubblicamente accessibili dai motori di ricerca!
- La dimensione del deep web è difficile da misurare. Si parla di decine di *petabytes*.
- Molte persone considerano il dark web come sinonimo di deep web. In realtà, il dark web è una parte (stavolta più minacciosa) del deep web.

The Onion Router

- TOR, The Onion Router, è un software che garantisce un **accesso a Internet completamente anonimo**. TOR è essenzialmente uno strumento per chi cerca il massimo anonimato possibile su Internet, ad esempio per aggirare la censura, ma anche per svolgere ogni tipo di operazione illegale.
- TOR permette di accedere qualsiasi sito **mascherando l'identità dei client e crittografando i dati finché questi non arrivano alla macchina di destinazione**. In questo senso, TOR offre un servizio simile a quello di una VPN in termini di crittografia e sicurezza dei contenuti, ma con delle differenze sostanziali per quel che concerne l'anonimato:
  - Sebbene le macchine a cui ci si connette tramite una VPN non possano individuare la sorgente effettiva della connessione, i provider delle VPN conoscono l'indirizzo IP dei client connessi, quindi la rete non è del tutto anonima, se il provider VPN può essere costretto a rivelare tale informazione.
  - Nella rete TOR invece l'indirizzo IP reale dei client connessi non è ricostruibile, ma il sistema che gestisce questo anonimato ha un forte costo in termini di velocità di connessione.
- Tecnicamente, TOR agisce in due modi:
  - **Fa rimbalzare le connessioni della sorgente alla destinazione finale su diversi server proxy** casuali intermedi e sparsi in tutto il mendo chiamati «nodi», in modo tale che nessuno possa ricostruire l'intero percorso dei dati e conoscerne il vero mittente.
  - **Sottopone i dati a diversi strati di crittografia**, uno per ciascun nodo attraversato, in modo che nessuno dei nodi possa effettivamente leggere i dati in transito.
- Tuttavia, se ci si connette a un sito sul surface web, l'ultimo tratto della connessione (dall'ultimo nodo TOR al server destinazione) non sarà protetto dalla crittografia TOR (in quanto la destinazione non è parte TOR), e quindi i dati (soprattutto se non protetti da sistemi di crittografia «del surface web» come SSL) saranno espugnabili, anche se il loro mittente rimarrà ignoto.
  - Per questo motivo sono esistono siti specifici, indentificati dal TLD .onion, che risiedono direttamente sulla rete TOR (onion network), spesso indicati anche come *TOR hidden services*, per i quali l'intera connessione, fino al servizio finale, è protetta dalla crittografia TOR. **Da tali siti nasce il cosiddetto dark web**.

Dark Web

- Il deep web non è raggiungibile dai motori di ricerca ma di solito può essere esplorato dai normali browser usati nel surface web: è semplicemente necessario conoscere le informazioni necessarie ad accedervi (hostname a cui connettersi, credenziali per login, ecc.)
- Per accedere ai siti del dark web, invece, è necessario usare un software specifico (TOR), poiché sono pubblicati sulla **onion network**, una sotto-rete virtuale di internet (che utilizza *le stesse tecnologie di base, come il TCP/IP, e la stessa infrastruttura fisica, ma ad esempio non gli stessi DNS*) protetta da specifici **sistemi crittografici**, **inaccessibile e invisibile per i normali browser**.
- I siti del dark web hanno anch'essi un FQDN, che però non corrisponde a un TLD standard come .com o .it, ma appartengono tutti allo speciale TLD **.onion**.
- Tuttavia, in questo caso è **il contenuto del sito che ne determina la classificazione**: viene considerato *dark web* solo il sottoinsieme dei siti sulla onion network che effettivamente opera nell'illegalità. Le altre parti della onion network restano semplicemente parte del deep web.
- A cosa serve un sito .onion se si agisce legalmente? Bisogna considerare che TOR nasce per ragioni di anonimato e privacy, che possono essere del tutto legittime, tant'è vero che sulla onion network esistono versioni .onion di siti notissimi quali Facebook (https://facebookcorewwwi.onion), tramite cui collegarsi al social network aggirando le restrizioni delle libertà personali presenti in alcuni Paesi.

Come funziona TOR
Anonimato e istradamento dei messaggi

- La rete TOR è composta da macchine, dette nodi, che vengono utilizzate per effettuare richieste per conto dell'utente.
- Le richieste istradate sulla onion network **percorrono tre nodi** (casuali) prima di arrivare al sito di destinazione:
  - **Guard (entry) node**: è il primo nodo dopo l'effettivo originatore del messaggio. Un piccolo numero di guard nodes sono precaricati nel TOR browser scegliendoli casualmente da una lista di nodi *considerati sicuri e non compromessi*. Infatti, se il guard node non è stato violato, è estremamente improbabile essere identificati, persino se gli altri lo sono.
  - **Middle node**: è scelto casualmente dalla lista pubblica di nodi TOR.
  - **Exit node**: comunica con il sito web di destinazione, ed è anch'esso scelto casualmente dalla lista pubblica di nodi TOR.
- Ogni nodo **conosce solo l'indirizzo IP del precedente e del successivo nella catena**, quindi solo il guard node conosce l'indirizzo sorgente reale del messaggio. E' praticamente impossibile collegare un messaggio in uscita da un exit node con uno in entrata a un guard node, e quindi conoscere il reale mittente di un pacchetto di dati.
- Inoltre il nodo usato da una connessione può essere usato anche da altre con un ruolo diverso nella catena, generando ancora più "confusione" e mascherando meglio il percorso, soprattutto la sua sorgente.

Come funziona TOR
Crittografia e sicurezza dei messaggi

- Quando si richiede al TOR browser l'accesso a un sito, per prima cosa il browser seleziona un guard node nella lista a sua disposizione.
- I dati da inviare al guard node **vengono crittografati dal browser tre volte** (uno strato per ogni nodo che sarà attraversato) e inviati al guard node.
- Ogni nodo, compreso il guard node, elimina uno strato di crittografia e invia i dati al successivo (scelto casualmente). *Questa crittografia «a strati» è all'origine del nome onion network*.
- L'exit node invierà i dati al sito di destinazione:
  - Se questo è parte del surface web, tali dati saranno in chiaro, a meno che non sia stato usato il protocollo HTTPS.
  - Se invece il sito è nella onion network, la crittografia TOR proteggerà anche quest'ultimo tratto.
- Il sito web vedrà la richiesta come proveniente dall'exit node, e a questo invierà la risposta.
- A questo punto, la risposta percorrerà a ritroso il cammino dall'exit node al guard node e poi al browser TOR dell'utente. Ad ogni passo il nodo raggiunto aggiungerà uno strato di crittografia ai dati, e sarà quindi il TOR browser dell'utente che «pelerà» via tutti i tre strati, accedendo infine al contento della risposta.

TOR garantisce sempre sicurezza e privacy?

- L'infrastruttura TOR (prima di tutto il browser TOR) necessaria per accedere alla onion network **deve spesso essere scaricata dal surface web**: in questo modo chi la scarica può venir identificato e sottoposto a controlli, anche se le sue comunicazioni effettive rimarranno anonime e sicure.
- L'efficienza, sicurezza e velocità della rete TOR sono direttamente legate al numero e alla distribuzione casuale di nodi esistenti. Per questo la rete TOR è gestita su base volontaria, e qualsiasi utente TOR può configurare il proprio computer come nodo TOR se lo desidera. Tuttavia, **i nodi TOR sono visibili in una directory pubblica**, quindi chi li ospita può essere individuato e controllato, anche se di principio non compie alcuna azione illegale.
- Proprio perché pubblici, **i nodi TOR sono facilmente soggetti ad attacchi hacker**. Se uno di questi riesce ad inserirsi su un nodo, può conoscere gli indirizzi IP dei nodi ad esso adiacenti. **Riuscendo a compromettere molti nodi, l'hacker potrebbe controllare l'intero percorso di istradamento di un messaggio** (i tre nodi), e quindi conoscere l'indirizzo IP della reale sorgente, eliminando l'anonimato.

La privacy sul Web

Quali informazioni permangono durante la nostra navigazione?

This work is licensed under CC BY-NC-SA 4.0. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-sa/4.0/

I Web browser

- Un browser web è di base **un client per la connessione Internet con protocollo HTTP(S), un visualizzatore HTML/CSS e un interprete Javascript**.
- Tuttavia, i browser moderni sono applicazioni **multiprotocollo**, cioè possono spesso fungere da client per altri protocolli in uso su Internet (ad esempio FTP) e anche visualizzare/interpretare **risorse locali** (tramite il cosiddetto protocollo FILE).
- Inoltre molte della applicazioni moderne si sono spostate in **cloud**, e vengono quindi eseguite come applicazioni web (*rich internet applications* o **RIA**) tramite un web browser.
- Infine, altre applicazioni, sia in ambito desktop che (soprattutto) in ambito mobile, anche se appaiono «native», sono invece applicazioni web eseguite da **browser headless**, cioè browser senza finestra incorporati nella finestra di un'altra applicazione.
- Per questo motivo, si tratta di **software critici per la sicurezza e la privacy degli utenti**.

I Web browser
History

- Il dato più basilare conservato da un browser è la **history** dei siti visitati.
- La history è utilizzata dai browser per agevolare gli utenti nell'accesso di siti precedentemente già visitati.
- La history contiene **tutte le URL richieste**, e spesso anche alcuni **dati ad esse associati** (ad esempio parametri di moduli codificati nelle *query string*).
- E' possibile **leggere e in alcuni casi modificare la history tramite Javascript** e persino con i CSS.
- L'accesso alla history di un browser rappresenta quindi un **veicolo di potenziale violazione della privacy** e persino della sicurezza dell'utente.
- La history può essere cancellata, disabilitata o limitata temporalmente dall'utente, mitigando i pericoli di cui sopra.

I Web browser
Cache

- La **cache** è un elemento del browser spesso ignoto o sottovalutato dagli utenti.
- Viene utilizzata per **velocizzare l'accesso** a molte risorse web (soprattutto immagini, script e fogli di stile), tenendone una **copia locale** e quindi senza doverle riscaricare dalla rete ad ogni richiesta.
- La cache contiene **tutti i file scaricati localmente, associati alle informazioni di ultimo accesso, URL di origine**, ecc.
- Al pari della history, quindi, la cache può essere usata per **analizzare le attività recenti dell'utente**.
- Non è però possibile accedervi via script: è necessario **avere accesso alla macchina dell'utente** per poterla esplorare.
- Sono anche noti dei veri e propri **attacchi che sfruttano la cache per minare la sicurezza della navigazione**: il più noto è la ***cache poisoning***, che permette di immagazzinare nella cache del browser un contenuto malevolo associandolo all'indirizzo di una pagina lecita, in modo che alla successiva richiesta l'utente riceva la pagina malevola in cache invece di scaricare quella lecita dalla rete.
- La cache viene periodicamente epurata dai file più vecchi e meno utilizzati, e può essere cancellata o disabilitata temporaneamente dall'utente.

I Web browser
Downloads

- I **file scaricati** da una pagina web costituiscono una traccia tangibile dell'attività dell'utente in rete.
- Tuttavia, in generale, per esaminare un file scaricato è necessario **accedere alla macchina** dell'utente: i file, una volta scaricati, non sono accessibili da Javascript.
- Gli utenti possono ovviamente cancellare i file scaricati eliminando quindi la traccia da essi costituita (ovviamente a patto che siano veramente eliminati dal disco!).
- Va però sottolineato che i file scaricati sono i soli elementi che **sopravvivono dalla cosiddetta navigazione in incognito** (o privata) dei browser moderni, cosa che spesso trae in inganno gli utenti.

I Web browser
Cookie

- Un cookie HTTP è un piccolo **frammento di dati** che un server invia al browser. **Il browser memorizza il cookie** e **lo rispedisce al server** con tutte le successive richieste, come parte delle intestazioni, senza alcun intervento dell'utente.
- I cookie sono utilizzati principalmente per tre scopi: gestione delle **sessioni**, memorizzazione permanente nel browser di **preferenze** dell'utente specifiche per un sito, registrazione e **analisi del comportamento** dell'utente (*profilazione*).
- Dopo aver ricevuto una richiesta HTTP, il server può inviare nelle intestazioni della risposta dei **Set-Cookie**. Il browser memorizzerà tali cookie li reinvierà al server insieme tutte le successive richieste usando l'intestazione **Cookie**.
  - Da notare che i cookie possono essere impostati da ogni richiesta HTTP, quindi ad esempio non solo dal server che restituisce una pagina HTML, ma anche da quelli su cui risiedono le risorse usate dalla pagina, ad esempio le immagini.
- E' possibile limitare l'invio del cookie alle sole **connessioni HTTPS** tramite l'attributo **Secure**.
- E' possibile (e spesso molto utile!) impedire la lettura e la **manipolazione del cookie da parte di script** tramite l'attribuito **HttpOnly**.

I Web browser
Cookie di sessione e permanenti

- È possibile specificare una **scadenza** per i cookie:
  - I **cookie di sessione** vengono eliminati al termine della sessione di navigazione corrente (solitamente quando il browser viene chiuso).
    - Sono solitamente accettabili e necessari per il funzionamento della applicazioni web.
  - I **cookie permanenti** vengono eliminati a una data specificata dall'attributo **Expires** o dopo un periodo di tempo specificato dall'attributo **Max-Age**.
    - Sono utilizzati molto spesso per la profilazione dell'utente, anche in maniera illegale.

I Web browser
Cookie di terze parti

- E’ possibile indicare il **dominio** e/o il **path** a cui il cookie deve essere rispedito (*ambito* o *scope* del cookie), anche diversi da quelli della risorsa che li ha impostati.
  - L'attributo **Domain** specifica quali **host** devono ricevere il cookie. Se non specificato, l'attributo predefinito è lo stesso host che ha impostato il cookie, esclusi i sottodomini. Se si specifica, i sottodomini sono sempre inclusi.
  - L'attributo **Path** indica un **percorso** che deve essere presente nella URL perché il cookie venga rispedito.
- Se il dominio e lo schema (HTTP, HTTPS) del cookie corrispondono a quelli della pagina corrente, il cookie viene definito **cookie first-party**.
  - I first-party cookies sono spesso cookie di sessione o usati per la memorizzazione di preferenze.
- In caso contrario, il cookie viene indicato come **cookie di terze parti**.
  - I cookie di terze parti sono utilizzati principalmente per la pubblicità e la profilazione.

I Web browser
Local storage

- Il **localStorage** e il **sessionStorage** sono strutture introdotte recentemente tramite delle specifiche Web API di Javascript.
- Entrambi consentono di **salvare dati associati all'origine** (dominio/protocollo/porta) del documento corrente. Tali dati vengono mantenuti se si aggiorna la pagina (per sessionStorage) o per sempre (per localStorage).
- A differenza dei cookie, però, questi dati non sono ricevuti o inviati al server con le richieste/risposte HTTP, ma sono **gestiti totalmente dal client tramite Javascript**.
- Pertanto se si memorizzano dati sensibili, soprattutto nel localStorage, **un attacco basato su Javascript potrebbe estrarli** e renderli noti all'attaccante.
- Come per tutti gli altri dati memorizzati dal browser, l'utente può mitigare questi problemi cancellando o limitando tale spazio di memorizzazione.

Le sessioni nelle applicazioni web

- HTTP è un protocollo *stateless*, come già illustrato: ciò significa che **non esiste un modo per tener traccia delle richieste HTTP correlate**. Tuttavia, senza il concetto di stato non sarebbe possibile, ad esempio, effettuare una login su un sito e poi navigarvi come utente registrato, tutte operazioni che si svolgono su connessioni differenti.
- Il concetto di **sessione** (da non confondersi l'omonimo livello dei protocolli di rete) è utilizzato ampiamente nella programmazione delle applicazioni web. La sessione permette di **associare delle informazioni di stato alle richieste di un utente**, permettendo in pratica di aggirare la caratteristica stateless del protocollo HTTP.
- Tipicamente, per gestire le sessioni si utilizza un identificatore unico (**session identifier**) che viene assegnato all'utente al suo ingresso nell'applicazione web (ad esempio quando esegue la login) e **trasmesso insieme ad ogni successiva richiesta HTTP**, per poi venire invalidato quando si effettua il logout o dopo un certo periodo di inattività.
  - Il server dell'applicazione web avrà a sua disposizione una tabella in cui a ogni session indentifier sono associate delle informazioni relative alla sessione cui è stato assegnato, che possono essere transitorie (*informazioni di sessione*) o permanenti (salvate tra le sessioni, nel caso queste siano associate con un utente tramite login, ad esempio). Questa informazioni costituiscono lo **stato**.

Le sessioni nelle applicazioni web
Sessioni e cookie

- Il session identifier deve essere un numero identificatore unico, e di solito viene generato con algoritmi random basati sulla data/ora corrente. Ma come avviene lo scambio del session identifier? Esistono principalmente due modi:
  - Il session identifier può essere **scritto nella pagina web** stessa, come campo di un modulo o parte delle URL dei suoi collegamenti: in questo modo, verrà inviato ogni volta che l'utente esegue la rispettiva azione di navigazione (ad esempio cliccando sul link contenente quella URL).
    - I session identifier incorporati nelle URL o nei campi di form trasmessi in modalità GET sono visibili nella barra degli indirizzi dei browser e spesso salvati anche nella sua cronologia, quindi particolarmente insicuri e semplici da rubare.
  - Esistono (anche se rari) schemi manuali di conservazione e trasmissione dei session identifier basati sul **sessionStorage** e sul localStorage.
  - Più comunemente, il session identifier è immagazzinato in un **cookie di sessione** dal server, e gli viene quindi automaticamente rispedito dal client ad ogni richiesta, fino alla sua scadenza. I normali **cookie di sessione** scadono non appena il browser viene chiuso.

La Privacy sul Web
Considerazioni di sicurezza

- Come già discusso, tutte le strutture appena elencate possono essere fonte di informazioni utili ad analizzare (anche in maniera illecita) le attività online dell'utente, violando quindi la sua privacy.
- Alcuni attacchi bastati su queste strutture, come il *cache poisoning*, possono essere usati per indurre l'utente a inserire dati sensibili in siti fasulli, compromettendone quindi la sicurezza.
- Tuttavia le sessioni, in tutte le loro implementazioni (cookie, parametri URL, localStorage), costituiscono il bersaglio preferito dagli hacker, perché rubare un identificativo di sessione vuol dire potersi spesso sostituire in tutto e per tutto all'utente reale, agendo per suo conto.
- Per mitigare questi pericoli, le applicazioni web dovrebbero sempre applicare dei sistemi di controllo minimi, come la **verifica degli IP** legati alle richiesta, **invalidare le sessioni inutilizzate** e richiedere l'inserimento delle credenziali, per **confermare l'identità dell'utente**, quando questo cerca di effettuare operazioni pericolose.

La sicurezza sul Web

Vulnerabilità, attacchi e protezione

This work is licensed under CC BY-NC-SA 4.0. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-sa/4.0/

Sicurezza: le vulnerabilità dei siti Web

OWASP Top Ten

- Quando si parla di sicurezza sul web, bisogna necessariamente citare un riferimento importante per gli sviluppatori in questo campo: OWASP.
- Di seguito vedremo, in maniera generale, quali sono le vulnerabilità più comuni delle applicazioni web moderne, e il modo in cui queste possono compromettere l'utente e i suoi dati.
- Le informazioni che seguono sono tratte dalle pagine OWASP, e in particolare dalla sua nota **Top Ten**.

Sicurezza: le vulnerabilità dei siti Web

Broken Access Control

- Si tratta, in generale, di errori di programmazione che permettono di violare il principio del *deny by default*, in base al quale ogni funzionalità dovrebbe essere negata all'utente a meno che questo non possegga determinati ruoli o privilegi.
  - Un sistema con questo tipo di vulnerabilità permette all'utente malevolo di aggirare il controllo degli accessi modificando manualmente le URL, il codice HTML o quello Javascript delle pagine.
- Solitamente questa vulnerabilità permette di **elevare i propri privilegi**, ad esempio agire come un utente senza esserlo o agire come amministratore quando si è un utente base.
  - Questo permette di visualizzare o modificare i dati degli account di altri utenti.

Sicurezza: le vulnerabilità dei siti Web

Cryptographic Failures

- Dati quali, password, numeri di carte di credito e informazioni personali richiedono una protezione adeguata, soprattutto se ricadono sotto le leggi sulla privacy.
- Gli errori più comuni in questo senso possono essere
  - Trasmettere i **dati in chiaro**, utilizzando protocolli come HTTP
  - Utilizzare algoritmi o **protocolli crittografici vecchi** o deboli
  - Utilizzare password o **chiavi crittografiche predefinite o deboli**
  - Non **validare il certificato di sicurezza** ricevuto dal server e la *chain of trust*
  - Utilizzare **vettori di inizializzazione** deboli per il funzionamento di alcuni algoritmi di crittografia
  - Utilizzare **password come chiavi crittografiche** senza l'utilizzo di una funzione di derivazione adeguata
  - Utilizzare **generatori casuali non adeguati** per la crittografia o non propriamente inizializzati
  - Utilizzare funzioni hash deprecate come MD5 o SHA1, o in ogni caso usare **funzioni hash non crittografiche per la conservazione delle password**

Sicurezza: le vulnerabilità dei siti Web

Injection

- L'injection è uno degli attacchi più comuni e maggiormente sfruttati nelle applicazioni web, tant'è vero che esistono specifici software per eseguire tentativi standard di injection senza la supervisione umana.
- La vulnerabilità all'injection deriva spesso da pratiche di programmazione non corrette:
  - Utilizzare dati forniti dall'utente senza validarli e/o **sanificarli**.
  - Costruire **interrogazioni dinamiche per i database** passandole direttamente all'interprete del linguaggio SQL, senza applicare escaping o tecniche similari.
- Tuttavia, le tecniche di injection **non si limitano all'interazione con i database relazionali**, come spesso si crede: oltre alla ben nota SQL injection, esistono la NoSQL, OS command, Object Relational Mapping (ORM), LDAP, e Expression Language (EL) injection!

Sicurezza: le vulnerabilità dei siti Web

Security Misconfiguration

- L'insicurezza spesso può derivare non solo dall'applicazione web vera e propria, ma sul mancato *hardening* di sicurezza di qualsiasi elemento dello stack che la supporta: sistema operativo (Linux, Windows, ecc.), server (Apache HTTPD, Tomcat, ecc.), linguaggio di programmazione server side (PHP, Java, Javascript, ecc.), framework di supporto e sviluppo (Node, Laravel, Spring, ecc.), DBMS (MySQL, Oracle, SQLServer, ecc.), librerie.
- Gli errori più comuni in questo senso possono essere
  - Sono abilitate o installate **funzioni non necessarie** (ad es. porte, servizi, pagine, account o privilegi non necessari)
  - Ci sono **account di default** sono ancora abilitati con password di default
  - A seguito di errori, vengono presentati all'utente stack traces o altri **messaggi di errore contenenti troppi dettagli applicativi**
  - Le impostazioni di sicurezza non sono corrette (magari ancora su **impostazioni di sviluppo** e non su quelle di produzione)
  - Il software **non è aggiornato** o è vulnerabile

Sicurezza: le vulnerabilità dei siti Web

Vulnerable and Outdated Components

- Con riferimento alla vulnerabilità precedente, tutto il software «di supporto» alle applicazioni web viene spesso aggiornato per sanare le falle di sicurezza via via scoperte.
- Tali aggiornamenti, però, raramente si riflettono in maniera automatica sulle applicazioni web, e richiedono agli sviluppatori operazioni manuali di riscrittura, ricompilazione, reinstallazione.
- Gli errori più comuni in questo senso possono essere
  - Non conoscere le **versioni di tutti i componenti** utilizzati (sia lato client che lato server) nell'applicazione
  - Non effettuare **scansioni delle vulnerabilità** regolarmente e/o non consultare i bollettini di sicurezza relativi ai componenti utilizzati
  - Non correggere o non **aggiornare tempestivamente** i componenti
  - Non testare la **compatibilità** delle librerie aggiornate

Sicurezza: le vulnerabilità dei siti Web

Identification and Authentication Failures

- Il sistema di autenticazione di un'applicazione web può essere attaccabile se
  - Permette **attacchi *brute force*** o altri attacchi automatizzati
  - Permette l'uso di **password di default** o deboli.
  - Utilizza un **recupero delle credenziali** debole o inefficace.
  - Memorizza le **password in chiaro**, o cifrate con funzioni di hash deboli.
  - Non ha un sistema di **autenticazione a più fattori** efficace.
  - Espone l'**identificatore di sessione del URL**.
  - **Riutilizza l'identificatore di sessione** dopo un login avvenuto con successo.
  - Non **invalida correttamente l'identificatore di sessione**: la sessione dell'utente o i token di autenticazione devono essere sempre invalidati durante il logout o dopo un periodo di inattività

Sicurezza: le vulnerabilità dei siti Web

Security Logging and Monitoring Failures

- Un'applicazione può essere vulnerabile se non dispone di log su cui effettuare analisi di sicurezza:
  - I login, i login falliti e le transazioni più importanti non vengono registrati
  - Gli avvisi e gli errori generati dall'applicazione non generano messaggi di log adeguati
  - I ***penetration test*** e le scansioni effettuati sull'applicazione non vengono rilevati e registrati
- ..oppure tali log esistono ma non sono monitorati!

Sicurezza: le vulnerabilità dei siti Web

Server-Side Request Forgery (SSRF)

- La Server-Side Request Forgery (SSRF) è una vulnerabilità che consente a un utente malintenzionato di indurre un'applicazione web ad **effettuare richieste da lui definite verso una destinazione arbitraria**.
  - E' possibile indurre l'applicazione web vulnerabile ad effettuare una richiesta verso un server interno alla stessa organizzazione, che non dovrebbe essere accessibile dall'esterno. Grazie alla SSRF, invece, l'attaccante può interrogare dall'esterno tale server, **usando il server pubblico dell'applicazione web come tramite**.
  - E' possibile indurre l'applicazione web vulnerabile ad effettuare una richiesta verso un server esterno, allo scopo di attaccarlo, mascherando però l'origine dell'attacco, che sembrerà **provenire dall'organizzazione che ospita l'applicazione vulnerabile**.
- Le vulnerabilità SSRF si verificano solitamente quando l'applicazione web accede a risorse remote tramite **URL costruite in base all'input dell'utente**, senza validarlo o sanificarlo.

Sicurezza: le vulnerabilità dei siti Web

Cross Site Scripting (XSS)

- Negli attacchi XSS (Cross-Site Scripting) script (Javascript) dannosi vengono iniettati in siti web affidabili.
  - Questo è in generale reso possibile da applicazioni web che utilizzano l'input di un utente all'interno dell'output generato senza validarlo o **sanitizzarlo**.
- Il browser dell'utente non ha modo di sapere che lo script non deve essere considerato attendibile e lo esegue. Poiché lo script sembra essere parte di un sito attendibile, questo **potrà in generale accedere a cookie, identificatori di sessione o altre informazioni sensibili conservate dal browser** e utilizzate con quel sito.
- Questi script possono anche riscrivere il contenuto della pagina HTML, dando all'utente l'illusione di trovarsi su una pagina diversa, ad esempio per **indurlo ad inserire le proprie credenziali**.

Strategie di Difesa della Privacy e della Sicurezza /1

- Disabilitare i cookie
  - Tutti i browser, direttamente o tramite plugin, permettono di **disabilitare i cookie**, escludendo magari quelli di sessione (cookie tecnici) o approvando selettivamente solo alcune origini note.
- Disabilitare (selettivamente) gli script
  - Esistono plugin quali *NoScript* che permettono di **disabilitare l'esecuzione di Javascript** proveniente da sorgenti non affidabili o non convalidate dall'utente. In questo modo, vengono meno le condizioni per l'attuazione di molti attacchi, ad esempio quelli basati su XSS.
- Navigare in incognito
  - La navigazione in incognito dei browser minimizza la quantità di **informazioni salvate localmente** (cookie, cache, history). Tuttavia, essa non ha alcuna influenza sui dati che possono essere **raccolti dal provider internet o dal sito a cui ci si connette**.
- Usare motori di ricerca che rispettano la privacy
  - Com'è noto, Google o Bing utilizzano i dati delle ricerche (compresi i click sui risultati) per migliorare il proprio algoritmo di classificazione, ma soprattutto per **profilare l'utente**, ad esempio per offrire pubblicità mirata. Esistono altri motori di ricerca, come ad esempio Duckduckgo, che sono molto più privacy-friendly.

Strategie di Difesa della Privacy e della Sicurezza /2

- Usare una rete privata virtuale (VPN)
  - Le VPN **mascherano l'indirizzo IP della macchina sorgente**, impedendo così il suo tracciamento, in quanto i dati da essa inviati vengono istradati sulla rete attraverso dei gateway proprietari della VPN (spesso ubicati anche in luoghi geografici diversi da quello in cui si trova l'utente). Il collegamento tra la macchina dell'utente e la VPN viene invece **crittografato**, rendendo impossibile leggerne i dati, anche nel caso in cui la rete a cui l'utente è effettivamente connesso (ad es. un WIFI pubblico) sia inerentemente insicura. Tuttavia, una volta «usciti» dalla VPN sulla rete, i dati saranno in chiaro a meno che non sia prevista ulteriore crittografia da parte dell'applicazione che li ha generati.
- Usare DNS sicuri
  - Poiché i DNS forniscono gli indirizzi IP finali di collegamento per la maggior parte del traffico, un DNS violato può **ridirigere su siti fake** anche se inseriamo l'indirizzo corretto del sito che vogliamo raggiungere, o comunque accumulare informazioni utili riguardanti i **siti che visitiamo**. Esistono siti come *DNSLeakTest.com* che permettono di verificare le vulnerabilità del proprio DNS.
- Usare macchine virtuali
  - Una macchina virtuale è una macchina dentro la macchina, che usa il suo hardware in maniera virtualizzata ed è totalmente isolata dalle sue reali risorse (file system, sistema operativo, applicazioni, ecc.). **Quel che succede all'interno della macchina virtuale non mette a rischio la sicurezza del sistema reale**. L'uso di macchine virtuali può essere utile, ad esempio, per aprire file sospetti o visitare siti potenzialmente malevoli.
- Usare sistemi operativi portatili
  - Un **sistema operativo eseguibile direttamente da un media portabile come un drive USB** (ad esempio *Tails*), permette di usare qualsiasi macchina (anche condivisa) per il proprio lavoro, senza lasciare su di essa alcuna traccia.

Il Web e la Legge: La protezione dei dati personali

GDPR e privacy policy

This work is licensed under CC BY-NC-SA 4.0. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-sa/4.0/

Il trattamento dei dati personali sul web

- Finora abbiamo parlato di privacy quasi del tutto in relazione ai dati che sono derivabili dall'attività online dell'utente agendo (in maniera lecita o illecita) sulla sua macchina, sul suo browser, o sulla sua connessione in rete.
- Tuttavia, una grande quantità di dati è accumulata e può essere estratta dalle **applicazioni web** con cui gli utenti interagiscono, o dagli **intermediari** che gestiscono la sua connessione in rete (si pensi agli Internet Service Providers).
- Le modalità con cui questi dati vengono conservati, e quelle con cui possono essere resi noti a terzi, sono (nel caso di soggetti non malevoli!) **regolamentate** da numerose leggi e regolamenti: in particolare, nella UE, sono oggetto del **GDPR**.

*«Il GDPR stabilisce norme relative alla protezione delle **persone fisiche** con riguardo al **trattamento dei dati personali**, nonché norme relative alla libera circolazione di tali dati»*

- I dati personali sono rappresentati da **qualsiasi informazione riguardante una persona fisica identificata o identificabile**

GDPR
Ambito di applicazione

- In linea di massima, il GDPR **vieta completamente il trattamento** (e quindi anche la conservazione e la trasmissione) **dei dati personali** a meno che questi non siano trattati per **l'esercizio esclusivo di attività a carattere personale, senza una connessione con un'attività commerciale o professionale.**
- In caso contrario, ogni trattamento richiede che l'interessato (detentore dei suddetti dati) abbia espresso il proprio consenso esplicito al trattamento per una o più finalità specifiche.
- Inoltre, il GDPR non si applica, e quindi è consentita la conservazione e l'accesso ai dati personali, nei seguenti casi:
  - Il trattamento è necessario per assolvere gli obblighi ed **esercitare i diritti specifici** del titolare del trattamento o dell'interessato in materia di **diritto del lavoro e della sicurezza sociale e protezione sociale**, ma in presenza di adeguate garanzie per i diritti fondamentali e gli interessi dell'interessato
  - Il trattamento è necessario per **tutelare un interesse vitale dell'interessato** (o di un'altra persona qualora l'interessato si trovi nell'incapacità fisica o giuridica di esprimere il proprio consenso)
  - Il trattamento è effettuato, nell'ambito delle sue legittime attività e con adeguate garanzie, da una fondazione, associazione o altro organismo senza scopo di lucro che persegua finalità politiche, filosofiche, religiose o sindacali (a condizione che il trattamento riguardi unicamente i membri, gli ex membri o le persone che hanno regolari contatti con la fondazione, l'associazione o l'organismo a motivo delle sue finalità e che i dati non siano comunicati a terzi senza il consenso dell'interessato)
  - Il trattamento riguarda **dati personali resi manifestamente pubblici** dall'interessato
  - Il trattamento è necessario per **accertare, esercitare o difendere un diritto in sede giudiziaria**
  - Il trattamento è necessario per **motivi di interesse pubblico** rilevante sulla base del diritto dell'Unione o degli Stati membri, che deve essere *proporzionato alla finalità perseguita*
  - Il trattamento è necessario per finalità di **prevenzione medica** o di medicina del lavoro, valutazione della capacità lavorativa del dipendente, **diagnosi, assistenza o terapia sanitaria** o sociale ovvero gestione dei sistemi e servizi sanitari o sociali, ma solo se tali dati sono trattati da o sotto la responsabilità di un professionista soggetto al segreto professionale
  - Il trattamento è necessario per **motivi di interesse pubblico nel settore della sanità pubblica**, quali la protezione da gravi minacce per la salute a carattere transfrontaliero o la garanzia di parametri elevati di qualità e sicurezza dell'assistenza sanitaria e dei medicinali e dei dispositivi medici
  - Il trattamento è necessario per **finalità di archiviazione nel pubblico interesse o per finalità di ricerca scientifica e storica o per finalità statistiche**

GDPR
La privacy policy

- Poiché **la maggioranza delle applicazioni web manipolano e conservano dati personali**, il GDPR richiede che queste pubblichino un'adeguata *privacy policy*, che l'utente possa leggere al fine di **accettare** (se necessario) il trattamento che verrà eseguito sui suoi dati, **sapere** quali sono questi dati, poter **esercitare** i suoi diritti in relazione al trattamento, ecc.
- La **privacy policy** di dovrebbe quindi contenere tutte le **informazioni relative al trattamento dei dati personali degli utenti**:
  - Il **titolare** del trattamento dei dati: l'utente ha il diritto di conoscere l'identità del soggetto che tratta i suoi dati ed i riferimenti per poterlo contattare. Se il titolare ha designato un responsabile della protezione dei dati (DPO), deve essere anch'esso riportato.
  - Gli **scopi** per cui i dati verranno trattati, nonché le basi giuridiche che rendono lecito quel trattamento.
  - I soggetti cui verranno eventualmente **comunicati** i dati personali.
  - I **paesi terzi** in cui i dati verranno eventualmente trasferiti (molte piattaforme, ad esempio, hanno i loro server localizzati nel territorio extra-UE).
  - Il periodo di **conservazione** dei dati
  - L'eventuale **obbligatorietà** (legale o come requisito per l'erogazione del servizio) del conferimento dei dati
  - L'eventuale esistenza di un **processo decisionale automatizzato** basato sui dati, compresa la *profilazione*, nonché l'importanza e le conseguenze previste di tale trattamento.
  - I **diritti** dell'interessato in ambito privacy con l'indicazione delle modalità di esercizio degli stessi.
    - Qualora il trattamento sia basato sul consenso, l'esistenza del diritto di revocarlo in qualsiasi momento.
    - Il diritto di proporre reclamo a un'autorità di controllo.

GDPR
La cookie policy

- La normativa prevede che l'utente debba essere **adeguatamente informato sull'uso dei cookie** ed esprimere il proprio valido consenso.
- Si tratta a tutti gli effetti di **informativa privacy specifica per i trattamenti che sfruttano i cookie** che dovrà rispettare i requisiti previsti dal GDPR.
- I cookie tecnici sono utilizzati per garantire le funzionalità di un sito web, ad esempio la permanenza dei dati di sessione, e non possono essere evitati se si vuole usufruire del servizio stesso.
- Altri tipi di cookie sono utilizzati per raccogliere informazioni **in forma aggregata,** ad esempio per elaborare statistiche sul servizio.
- Tuttavia i cookie possono essere utilizzati anche per **monitorare e profilare il singolo utente**, ad esempio studiarne le abitudini in base ai siti visitati allo scopo di inviargli pubblicità mirata. E' chiaramente a questo tipo di cookie che si rivolge maggiormente la normativa.

Riferimenti ed Approfondimenti

This work is licensed under CC BY-NC-SA 4.0. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-sa/4.0/

Riferimenti per le informazioni trattate in queste slide /1

Nozioni di Base

- Uniform Resource Identifier (URI): Generic Syntax, RFC 3986: https://www.rfc-editor.org/info/rfc3986
- Unicode: https://home.unicode.org
- World Wide Web Consortium (W3C): https://www.w3.org

I Protocolli del Web

- Hypertext Transfer Protocol: https://httpwg.org
- TLS spiegato su MDN: https://developer.mozilla.org/en-US/docs/Glossary/TLS
- TLS e sicurezza su OWASP: https://cheatsheetseries.owasp.org/cheatsheets/Transport_Layer_Protection_Cheat_Sheet.html
- Come vengono visualizzati i certificati TLS nei browser (e cosa possiamo verificare), dal blog di *GlobalSign*: https://www.globalsign.com/en/blog/how-to-view-ssl-certificate-details

Il Web Lato Client

- Web Hypertext Application Technology Working Group (WHATWG): https://whatwg.org
- Cascading Style Sheets (CSS): https://www.w3.org/Style/CSS
- ECMAScript 2023 Language Specification: https://tc39.es/ecma262
- MDN Web Docs: https://developer.mozilla.org

Riferimenti per le informazioni trattate in queste slide /2

La Privacy sul Web

› Cookies: cosa sono, come si usano e considerazioni sulla sicurezza su MDN: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies

› Sessioni ed applicazioni web (con considerazioni sulla sicurezza) su OWASP: https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html

La Protezione dei dati Personali sul Web

› General Data Protection Regulation: https://eur-lex.europa.eu/eli/reg/2016/679/oj

La Sicurezza sul Web

› CVE: https://www.cve.org

› The Open Web Application Security Project (OWASP): https://owasp.org

TOR

› TOR Project: https://www.torproject.org

› AmIUnique (Browser fingerprint): https://www.amiunique.org/fp