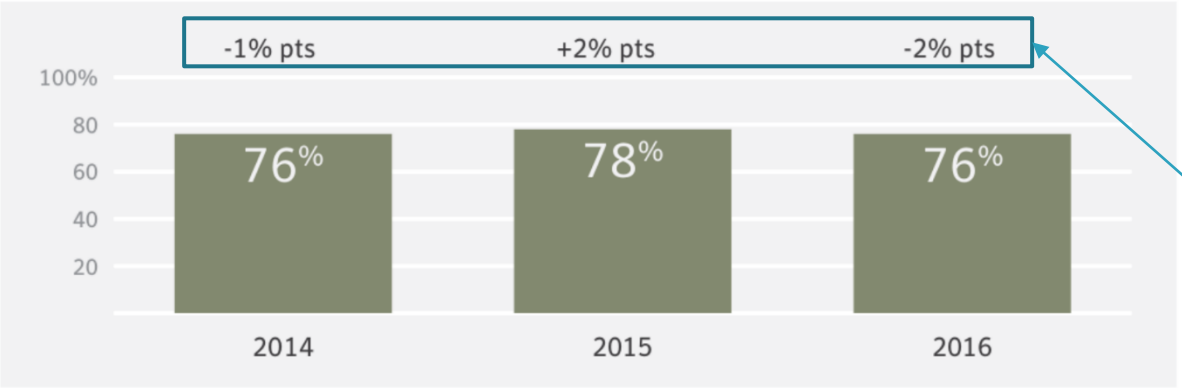
Sicurezza Web e dei protocolli applicativi Internet

Indice

- Cosa è la sicurezza Web
- HTTP in breve
- Problemi di sicurezza client e server side
- TLS

Sicurezza Web: perchè?

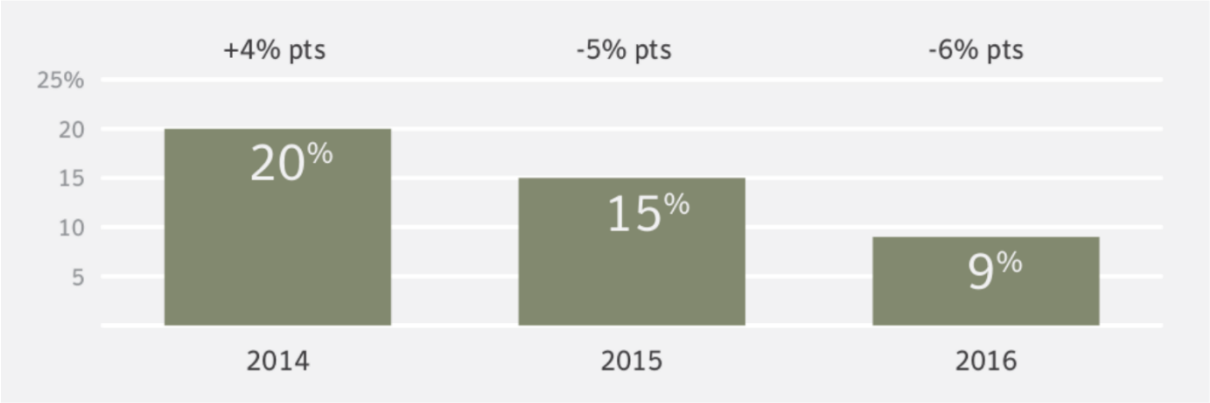
- Negli ultimi anni, circa il 76% dei siti web ha mostrato di contenere vulnerabilità.



Differenze in punti percentuali del numero delle vulnerabilità rispetto all'anno precedente

Sicurezza Web: perchè? /2

- Tra queste, una percentuale importante è «critica» ma una forte coscienza delle problematiche di sicurezza web, unita all'acquisizione di nuove competenze in questo settore, ha permesso di ridurre la percentuale di tali vulnerabilità critiche negli anni.



Cosa è la sicurezza Web?

- La sicurezza web non è un ambito ben definito come ad esempio la crittografia.
- La sicurezza web e di rete dipende da:
  - Dettagli e standard di rete
  - Dettagli implementativi
  - Versioni utilizzate di browser e server
- La sicurezza web studia attacchi legati alla sicurezza, alla privacy e alla qualità del servizio.
- Non esistono in questo campo soluzioni definitive e durature.

Ripasso: HTTP in breve

- HyperText Transfer Protocol (HTTP) è definito dall'RFC 2068.
- HTTP è un protocollo applicativo
- HTTP trasferisce richieste di «ipertesto» dal client (il browser) al server, e ritorna delle pagine web al browser che le interpreta e le «rappresenta» all'utente.

HTTP: lato client

- Il client inizia tutte le comunicazioni verso il server. Ogni richiesta inizia con un metodo
  - GET: utilizzato per richiedere dati da una determinata risorsa, è uno dei metodi HTTP più comuni.
  - POST: utilizzato per inviare dati ad un server per creare/aggiornare una risorsa.
- Per ogni richiesta, il client manda al server un HTTP header (non cifrato) che contiene informazioni di cui alcune sensibili, come
  - l'indirizzo IP
  - il linguaggio richiesto
  - l'encoding dei caratteri
  - la versione del browser e del sistema operativo.
- L'analisi degli header HTTP può portare a profilare gli utenti.

HTTP: lato client /2

- I metodi per la richiesta (i.e., HTTPRequest) sono:

Method	Description
GET	request a web page
HEAD	request header of a web page
PUT	store a web page
POST	request with payload

- Esempi di HTTP header per tali richieste:

GET Request

```
GET /search.jsp?name=blah&type=1 HTTP/1.0
User-Agent: Mozilla/4.0
Host: www.mywebsite.com
Cookie: SESSIONID=2KDSU72H9GSA289
<CRLF>
```

POST Request

```
POST /search.jsp HTTP/1.0
User-Agent: Mozilla/4.0
Host: www.mywebsite.com
Content-Length: 16
Cookie: SESSIONID=2KDSU72H9GSA289
<CRLF>
name=blah&type=1
```

HTTP: lato client /3

- Il metodo GET espone informazioni sensibili per l'autenticazione dell'URL
  - Nei Web Server e nei log dei Proxy server
  - Nel campo HTTP referer
- Il POST mette le informazioni nel body della richiesta e non nell'URL
- Occorre usare il POST con HTTPS (che viene realizzato tramite protocollo Trasport Layer Security - TLS) per inviare dati sensibili al server!

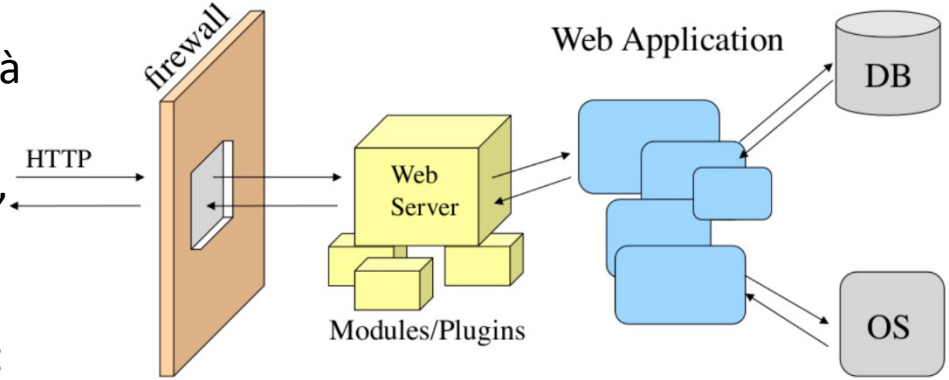
HTTP: lato server

Il server fornisce «dati» in risposta alla richiesta del client:

- Dati arbitrari possono essere trasferiti (sarà compito del client gestirli).
- I dati possono essere statici (pagine HTML, immagini, …) o dinamici (i.e. calcolati su richiesta da parte di una web application)

Esecuzione di codice (scripting) può avvenire:

- Server-Side (e.g. perl, asp, jsp)
- Client-Side (javascript, flash, applets)



Cookie

- HTTP è stateless, ovvero non utilizza la nozione di di sessione: se il client e il server eseguono due volte il protocollo HTTP, queste due esecuzioni sono logicamente scorrelate.
- Tuttavia, occorre tenere traccia di questo collegamento. Per farlo, si usano i cookie. L'idea è semplice:
  - Il client fa una richiesta ed il server, in risposta, include un file detto cookie
  - Ad ogni successiva richiesta HTTP, il client allega il cookie in modo che il server possa collegare la sessione corrente a quelle precedenti.
  - Il cookie ha un tempo di vita fissato e può contenere ogni tipo di informazione.

Cookie e privacy

- I cookie hanno ricevuto molte critiche per la privacy, tanto è vero che ora, quando ci si collega ad un sito web viene richiesto all'utente di «accettare» i cookie.
- I cookie possono essere usati per tracciare gli utenti in diversi modi:
  - Analizzando i log dei server
  - Ascoltando il traffico (gli header HTTP non cifrati sono informativi)
  - Applicando firewall e proxy che monitorino il passaggio dei cookie tra client e server.
  - Leggendo la «cronologia» del browser
- I cookie devono essere considerati come informazioni *confidenziali*.

Autenticazione HTTP

- HTTP supporta due metodi di autenticazione verso il server:
  - **Basic authentication:**
    - Basata su login e password.
    - Le credenziali sono però inviate in chiaro!
    - Le credenziali sono inviate in risposta ad ogni richiesta proveniente dal server
    - È il metodo più adottato perché supportato da praticamente tutti i client e i server.
  - **Digest authentication:**
    - Il server invia un nonce
    - Il client crea un hash crittografico del nonce, basata su login e password.
    - Il client non manda login e password ma solo un hash
    - È un metodo molto più sicuro ma viene usato raramente.

Considerazioni Generali

- Stare attenti quando si usano browser da computer pubblici
- I siti vistati sono salvati:
  - Nella cronologia del browser
  - Nella cache del browser
  - E possono essere rivelati tramite le funzioni di autocompletamento
- Usare i password manager con attenzione
- Molti attacchi sono causati da componenti maliziosi attivi ed eseguiti sul browser (JavaScript, ActiveX, . . .).

Rendere sicuro HTTP: TLS

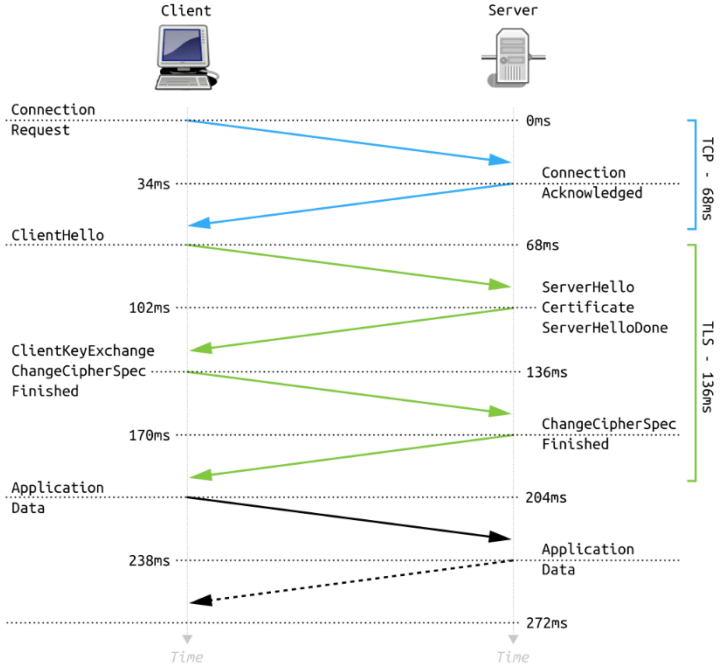
- Per ovviare alla mancanza di confidenzialità, integrità e autenticazione di HTTP, sono stati sviluppati opportuni protocolli crittografici che lavorano «alla base» di HTTP: TLS e il suo precessore SSL.
- HTTP + SSL/TLS = HTTPS, ovvero la versione «sicura» di HTTP.
- TLS sta per «Transport Layer Security» ed è uno standard IETF.
- Quando un client ed un server comunicano, TLS garantisce che nessuna terza parte maliziosa sia in grado di leggere o modificare i messaggi senza che le parti se ne rendano conto.

TLS: fasi

- Il TLS è composto da due fasi (o protocolli):
  - TLS Handshake Protocol
  - TLS Record Protocol
- Il TLS Handshake Protocol permette al client ed al server di autenticarsi (più o meno mutuamente) e di negoziare sia l'algoritmo di cifratura che le chiavi crittografiche che saranno poi usate dal TLS Record Protocol per cifrare le comunicazioni.
- Il TLS Record Protocol garantisce una comunicazione sicura cifrando il traffico con un algoritmo a chiave simmetrica (come il DES), ma può essere anche usato senza cifratura in caso non sia necessaria.

TLS Handshake Protocol: passi

- Il client manda un messaggio «ClientHello» al server con i cifrari che supporta ed un valore random;
- Il server risponde con un suo valore random ed il suo certificato a chiave pubblica. In caso di mutua autenticazione, richiede anche un certificato del client;
- Il client crea una pre-master key, la cifra con la chiave pubblica del server e gliela invia (tramite HTTP).
- Il server riceve la pre-master key, e client e server usano la pre-master key per generare le successive chiavi.
- Il client manda un messaggio «ChangeCipherSpec» al server per indicare che inizierà a mandare messaggi cifrati con quel cifrario, usando le chiavi di sessione create tramite la pre-master key. Inoltre, manda un messaggi finale «ClientFinished».
- Il server risponde con ServerFinished ed il protocollo si conclude.



TLS Record Protocol

- Il TLS Record Protocol è responsabile sia della valutazione dell'integrità dei dati sia dell'autenticazione delle parti, nello specifico, TLS:
  - Divide i messaggi in parti e riassembla i messaggi ricevuti.
  - Comprime i blocchi in uscita e decomprime quelli in ingresso (opzionale)
  - Applica il Message Authentication Code (MAC) ai messaggi in uscita e verifica il MAC su quelli in entrata
  - Cifra i messaggi in uscita e decifra quelli in entrata.
- I pacchetti elaborati dal TLS sono poi inviati al livello TCP, che li gestisce ed li invia come pacchetti di rete dello standard TCP/IP.

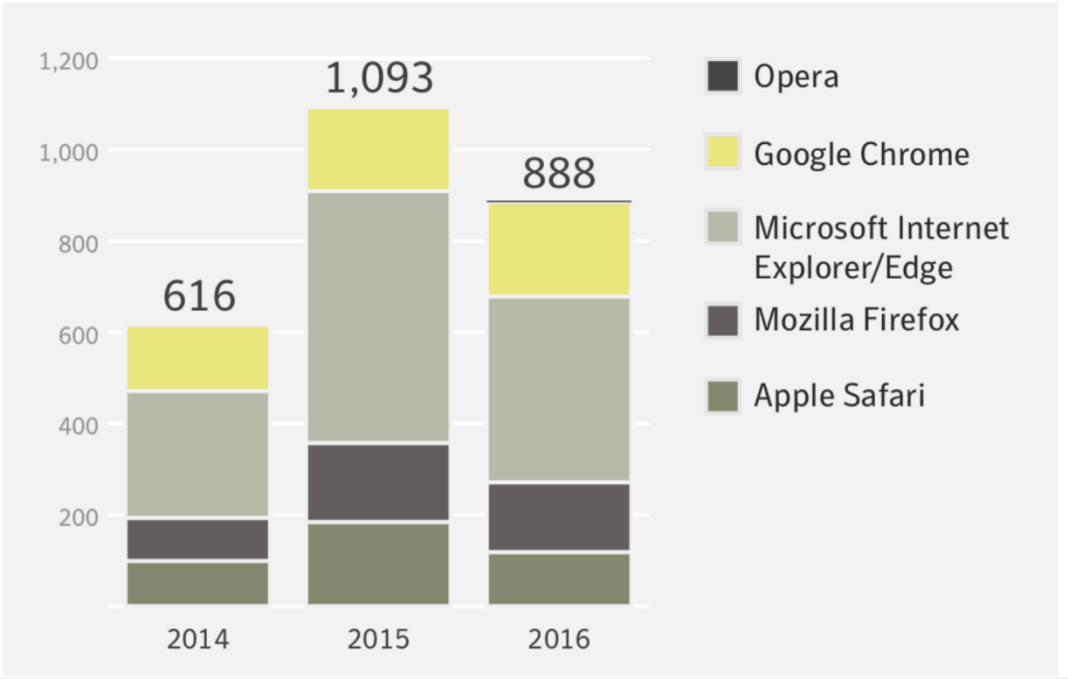
Approfondimento 1: Classificazione dei siti web maggiormente attaccati

Classificazione dei siti web maggiormente vittima di attacchi negli anni 2015 e 2016, divisi per categorie tematiche.

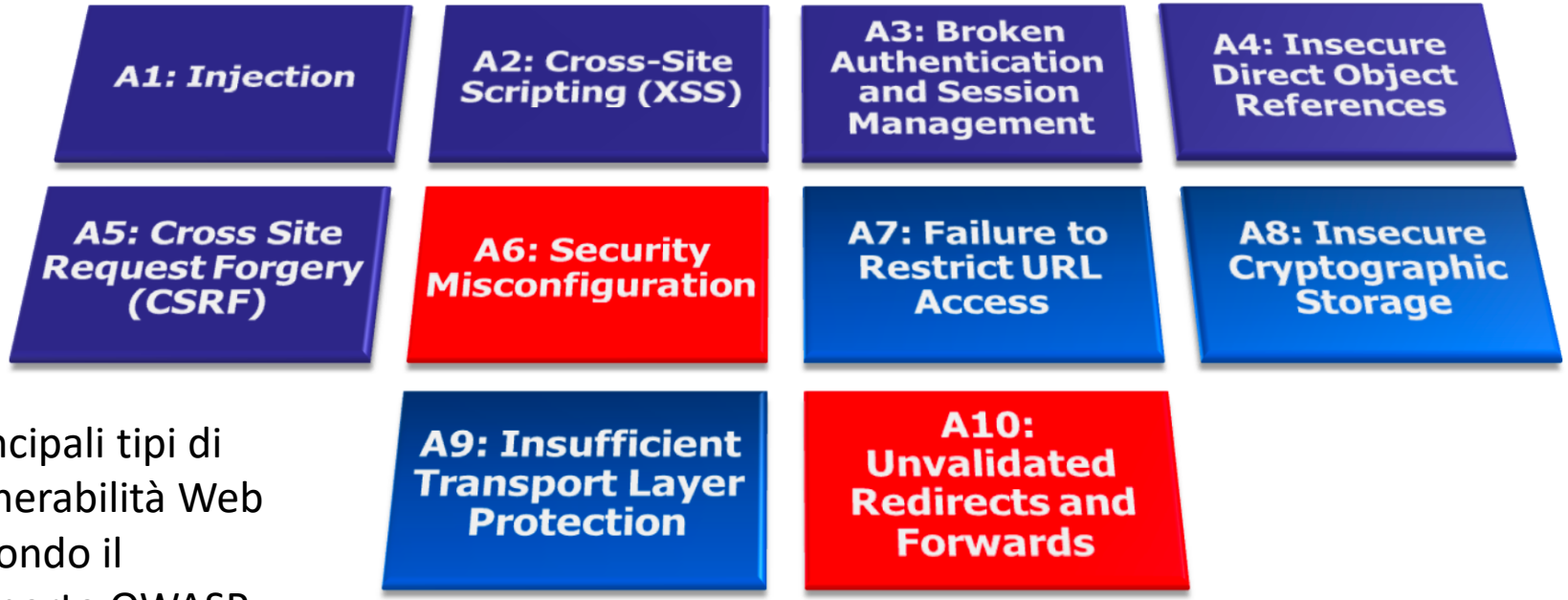
Rank	Domain Categories	2015 (%)	2016 (%)	Percentage Point Difference
1	Technology	23.2	20.7	-2.5
2	Business	8.1	11.3	3.2
3	Blogging	7.0	8.6	1.6
4	Hosting	0.6	7.2	6.6
5	Health	1.9	5.7	3.8
6	Shopping	2.4	4.2	1.8
7	Educational	4.0	4.1	< 0.1
8	Entertainment	2.6	4.0	1.4
9	Travel	1.5	3.6	2.1
10	Gambling	0.6	2.8	2.2

Approfondimento 2: Vulnerabilità dei browser

Numero di vulnerabilità presenti dei principali browser web, dal 2014 al 2016.



Approfondimento 3: OWASP Top 10 Application Security Risks



Principali tipi di vulnerabilità Web secondo il rapporto OWASP.