

Máster Ciberseguridad y Privacidad

Seguridad y pentesting de servidores de datos

Práctica de la asignatura

- Para dudas y aclaraciones sobre el enunciado, debéis dirigiros al consultor responsable de vuestra aula.
- La actividad es completamente individual y se podrá calificar con suspenso todas aquellas entregas que sean susceptibles de haber realizado una copia.
- No está permitido modificar nada del servidor, ya que puede perjudicar al resto de compañeros del aula que han de realizar la actividad.
- Hay que entregar la solución en un fichero .zip que incluya la memoria de la práctica en formato PDF y el código fuente de los ejercicios realizados.

Propiedad intelectual

Con frecuencia es inevitable hacer uso de recursos creados por terceras personas. Es por tanto comprensible hacerlo en el marco de una práctica de los estudios del Master en Seguridad Informática, siempre y cuando esto se documente claramente y no suponga plagio en la práctica.

Por tanto, al presentar una práctica que haga uso de recursos ajenos, deberán citarse todas las fuentes utilizadas.

Enunciado

1- SQL Injection (1punto)

La inyección de comandos es una técnica de ataque a aplicaciones Web cuyo objetivo principal es aprovechar conexiones a bases de datos desde aplicaciones web no aseguradas para permitir a un atacante la ejecución de comandos directamente en la base de datos.

De entre todas las posibles acciones que un atacante podría realizar mediante este ataque, el objetivo de esta actividad es la modificación de las

consultas para acceder a registros y/o objetos de la base de datos a los que inicialmente no se debería tener acceso.

Para esta actividad se proporciona la URL:

`http://(IP_proporcionada_en_el_tablón_del_campus)/UOC/Alumnos/sqlinjection/`

donde se deberá identificar una vulnerabilidad de *SQL Injection* y explotarla, de modo que se extraiga de la base de datos la máxima información posible.



Figura 1: Aspecto del portal web vulnerable

Se espera que este ejercicio se resuelva de manera manual, sin utilizar herramientas para automatizar la extracción de los datos.

2- *Blind SQL Injection* (2 puntos)

Una de las técnicas utilizadas en *SQL Injection* consiste en los ataques a ciegas, es decir, conseguir que los comandos se ejecuten sin la posibilidad de ver ninguno de los resultados. A pesar de que un atacante no puede ver los resultados extraídos directamente de la base de datos sí que puede detectar cambios en la página de respuesta según los parámetros enviados al servidor.

El objetivo de esta actividad consiste en detectar el parámetro vulnerable a *Blind SQL Injection* presente en la URL

`http://(IP_proporcionada_en_el_tablón_del_campus)/UOC/Alumnos/blindsqlinjection/`

y explotarla, extrayendo nuevamente la máxima información posible, incluyendo contraseña y servidores de los diferentes usuarios dados de alta en la base de datos en el entorno simulado.



Figura2: Entorno simulado para la explotación de la vulnerabilidad Blind SQL Injection





Una vez realizada la explotación y extracción de la información se deberá adjuntar un fichero con toda la información obtenida, así como una explicación de las acciones realizadas, y las herramientas o metodología utilizadas, incluyendo cualquier script o herramienta que hayáis desarrollado para extraer los datos.





Aunque se utilicen herramientas automáticas para realizar la extracción completa es imprescindible para aprobar este ejercicio que quede demostrado que el alumno entiende y conoce el proceso que se debería seguir en este caso concreto para extraer los datos de manera manual. En este sentido, al menos una parte de los datos deben ser extraídos de manera manual.

3- Identidades digitales (3'5 puntos)

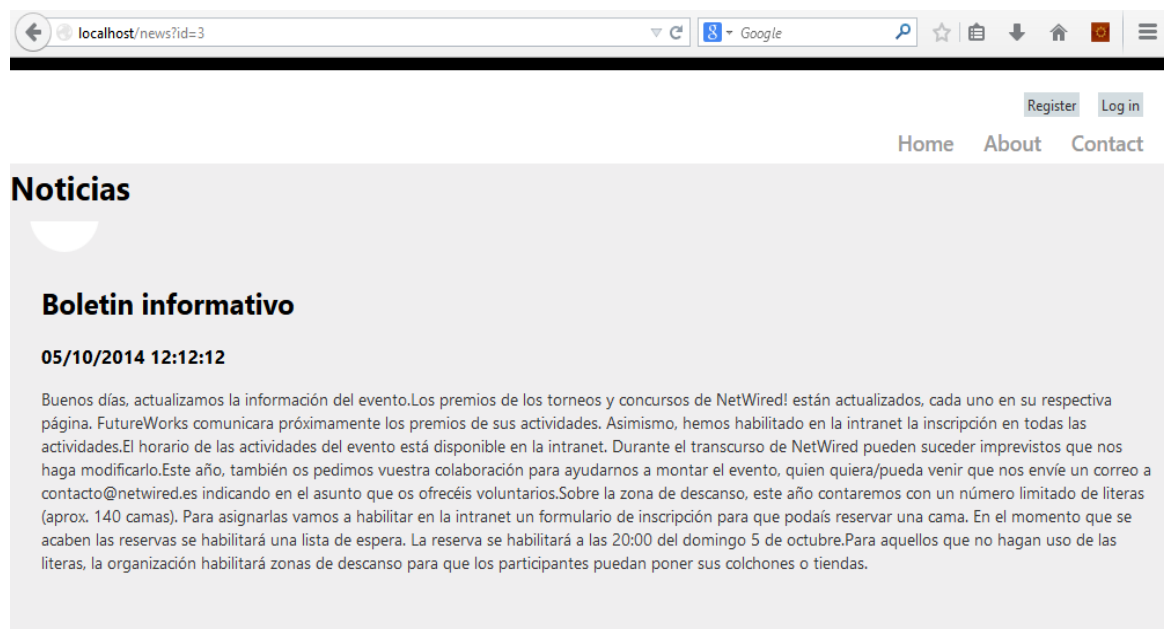
En este apartado se van a poner en práctica los siguientes conceptos: *Blind SQL Injection*, identidades digitales y la protección de éstas. **Todo el proceso llevado a cabo por el estudiante debe ser documentado en una memoria de práctica en la que se incluyan tanto capturas de pantalla de la aplicación como los extractos comentados más relevantes del código desarrollado.**

- a) Se ha de implementar un entorno web utilizando la tecnología que se prefiera (PHP, Java, .Net...) en la que se incluya una vista de *login* dónde un usuario pueda autenticarse. Esta aplicación web deberá disponer de una base de datos SQL con el siguiente esquema:

Users	
	Email
	Name
	Password
	Accountid

News	
	Id
	Title
	Body
	Datetime

- b) Crear una vista o página para mostrar una noticia en concreto a través de un parámetro denominado *id*. Este parámetro será consultado a través de un método HTTP GET. Como ejemplo se indica el siguiente ***http://localhost/news?id=3***, dónde *id* es el atributo de la tabla *News* y el valor 3 corresponde con un registro de la tabla. Ejemplo:



Register Log in

Home About Contact

Noticias

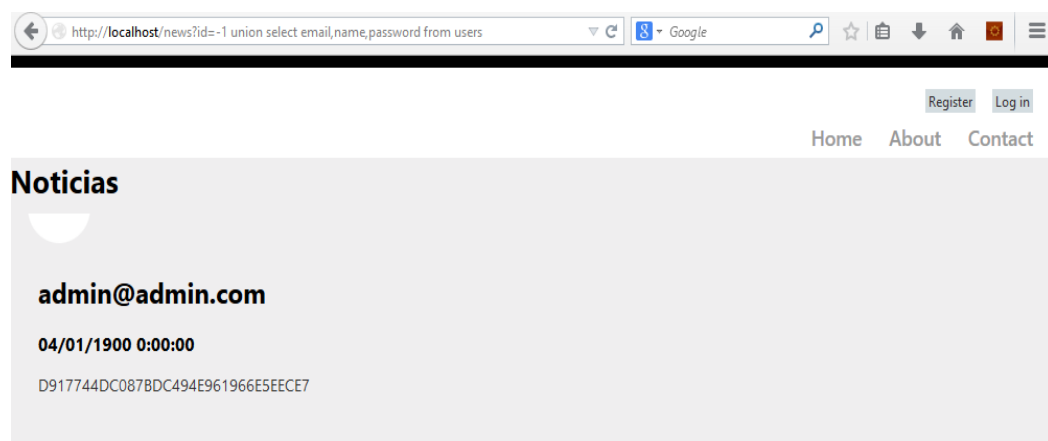
Boletín informativo

05/10/2014 12:12:12

Buenos días, actualizamos la información del evento. Los premios de los torneos y concursos de NetWired! están actualizados, cada uno en su respectiva página. FutureWorks comunicará próximamente los premios de sus actividades. Asimismo, hemos habilitado en la intranet la inscripción en todas las actividades. El horario de las actividades del evento está disponible en la intranet. Durante el transcurso de NetWired pueden suceder imprevistos que nos haga modificarlo. Este año, también os pedimos vuestra colaboración para ayudarnos a montar el evento, quien quiera/pueda venir que nos envíe un correo a contacto@netwired.es indicando en el asunto que os ofrecéis voluntarios. Sobre la zona de descanso, este año contaremos con un número limitado de literas (aprox. 140 camas). Para asignarlas vamos a habilitar en la intranet un formulario de inscripción para que podáis reservar una cama. En el momento que se acaben las reservas se habilitará una lista de espera. La reserva se habilitará a las 20:00 del domingo 5 de octubre. Para aquellos que no hagan uso de las literas, la organización habilitará zonas de descanso para que los participantes puedan poner sus colchones o tiendas.

El parámetro *id* debe ser vulnerable a una **inyección SQL**.

A continuación se muestra un ejemplo de una captura en la que se realiza una inyección SQL:



Register Log in

Home About Contact

Noticias

admin@admin.com

04/01/1900 0:00:00

D917744DC087BDC49A961966E5EECE7

En la imagen se puede visualizar como la contraseña del usuario admin@admin.com queda expuesta y se produce un robo de identidad digital. Se han de implementar la base de datos y el método para extraer la base de datos desde la inyección SQL.

c) Opciones para evitar la vulnerabilidad

En este apartado se deben describir las diversas opciones que existen para evitar la vulnerabilidad de SQL en la aplicación web que habéis desarrollado con indicaciones claras de cómo se implementarían en vuestra aplicación.

d) Solución de la vulnerabilidad mediante el uso de *Stored Procedures*

Utiliza un *Store Procedure* para solucionar la vulnerabilidad y documenta el proceso seguido y la verificación de su resolución.

4- Inyección en base de datos NoSQL (3'5 puntos)

En este ejercicio vamos a explorar la posibilidad de realizar inyecciones en una base de datos NoSQL como MonboDB. **Todo el proceso llevado a cabo por el estudiante debe ser documentado en una memoria de la práctica que incluya el detalle del código fuente implementado.**

- Se pretende implementar un entorno web utilizando la tecnología que se prefiera (PHP, angularJS, Java, .Net...) en la que se incluya una vista de *login* dónde un usuario pueda autenticarse. Esta aplicación web deberá utilizar una colección de la base de datos MongoDB en la que se guardarán los nombres de usuarios y contraseñas. Si el usuario y contraseña insertados en la página de login existe en la colección de MongoDB consideraremos al usuario autenticado y le daremos un mensaje de bienvenida. En caso de que el nombre de usuario o contraseña no sean correctos enviaremos un mensaje de error y pediremos al usuario que vuelva a intentar autenticarse.
- El parámetro de la contraseña deberá ser vulnerable a una inyección NoSQL permitiendo a un usuario malicioso obtener el mensaje de bienvenida sin necesidad de conocer la correspondiente contraseña.
- Explica qué medidas podrían adoptarse para evitar la vulnerabilidad. Utiliza una de estas medidas en tu código y verifica que, efectivamente, la vulnerabilidad ha sido corregida.

Referencia:

https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/07-Input_Validation_Testing/05.6-Testing_for_NoSQL_Injection