Practica Auditoria de Seguridad

Asignatura: Gestión de la información en la Web

Practica Auditoria de Seguridad

Grupo: 7

Autores: MIGUEL ÁNGEL ARROYO CLEMENTE, DANIELA-NICOLETA BOLDUREANU, DAVID PRATS ULLOA, IVÁN RUIZ QUINTANA.

MIGUEL ÁNGEL ARROYO CLEMENTE, DANIELA-NICOLETA BOLDUREANU, DAVID PRATS ULLOA, IVÁN RUIZ QUINTANA declaramos que esta solución es fruto exclusivamente de nuestro trabajo personal. No hemos sido ayudados por ninguna otra persona ni hemos obtenido la solución de fuentes externas, y tampoco hemos compartido nuestra solución con nadie. Declaramos además que no hemos realizado de manera deshonesta ninguna otra actividad que pueda mejorar nuestros resultados ni perjudicar los resultados de los demás.

**INFORME DE VULNERABILIDAD**

**Vulnerabilidad 1:**

|  |
| --- |
| **Rutas(s) de la aplicación involucrada(s):** /insert\_question |
| **Tipo de vulnerabilidad:** inyección SQL |
| **Causante de Vulnerabilidad:**    Cuando introducimos una sentencia SQL muy concreta en alguna(s) de la(s) entrada(s) de la pagina web , que en este caso son Autor ,Titulo, Etiquetas, Cuerpo , se ejecuta sin el consentimiento del programador o programa. |
| **Situaciones peligrosas o no deseadas que puede provocar:**  Una situación no deseada o peligrosa es la eliminación de una tabla de la bases de datos o la eliminación de la base de datos. |
| **Ejemplo paso a paso de como explotar la vulnerabilidad (con capturas de pantalla):** |
| **Medidas para mitigar la vulnerabilidad:**  Tenemos que ser realmente consientes porque cualquier entrada puede acabar en una consulta SQL. Para mitigar la vulnerabilidad hay que revisar todas las entradas de usuario comprobando sus tipos, escapar todos los caracteres problemáticos( ejemplo: ‘ , “ ; --) , utilizar sentencias SQL preparadas ,desarrollar una capa abstracta de seguridad reutilizable entre proyectos(Se encargara de todas las entradas de los usuarios ,e incluso de llamadas a la base de datos). |

**Vulnerabilidad 2:**

|  |
| --- |
| **Rutas(s) de la aplicación involucrada(s):** message\_detail.html, action="search\_question", @post(‘/insert\_question’) |
| **Tipo de vulnerabilidad:** XSS persistente |
| **Causante de Vulnerabilidad:**      Permite introducir scripts HTML en, por ejemplo, descripciones de preguntas o cualquier campo de una entrada de un foro sin que este texto sea comprobado antes de guardarlo. |
| **Situaciones peligrosas o no deseadas que puede provocar:**  Puede provocar que se muestren páginas web no deseadas o mensajes en la pantalla independientes del programa o el programador. |
| **Ejemplo paso a paso de como explotar la vulnerabilidad (con capturas de pantalla):**    Una vez guardados los datos nos vamos a la página que muestra las preguntas, al mostrar la pregunta añadida anteriormente se ejecutará el script. |
| **Medidas para mitigar la vulnerabilidad:**  Validar las entradas de usuario, desinfectar todo el texto que va a aparecer en la página HTML generada.  Es importante escapar los caracteres '<', '>', '&', comillas simples (') y dobles (“).  Se puede filtrar la salida permitiendo HTML seguro (basado en “listas blancas”). |

**Vulnerabilidad 3:**

|  |
| --- |
| **Rutas(s) de la aplicación involucrada(s):** message\_detail.html, action="search\_question" |
| **Tipo de vulnerabilidad:** XSS reflejado |
| **Causante de Vulnerabilidad:** Estas dos imágenes del código reflejan la vulnerabilidad en la que se puede introducir un script en la posición de tag. |
| **Situaciones peligrosas o no deseadas que puede provocar:** Se pueden realizar todas las opciones posibles que permita HTML ya que se introduce código y permite usar todo el potencial del lenguaje. |
| **Ejemplo paso a paso de como explotar la vulnerabilidad (con capturas de pantalla):** desde la página de search\_question podemos forzar una alerta de esa manera |
| **Medidas para mitigar la vulnerabilidad:** Validar los parámetros que se introducen en la ruta, no permitiendo símbolos como “<” |