GIW 2022-23

**Práctica**: Seguridad web

**Grupo**: 04

**Autores**: PETAR KONSTANTINOV IVANOV, JORGE SAN FRUTOS IGLESIAS, IGNACIO VILLEGAS DE MIQUEL y YUEJIE XU

**PETAR KONSTANTINOV IVANOV, JORGE SAN FRUTOS IGLESIAS, IGNACIO VILLEGAS DE MIQUEL y YUEJIE XU** declaramos que esta solución es fruto exclusivamente de nuestro trabajo personal. No hemos sido ayudados por ninguna otra persona ni hemos 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** |
| Ruta(s) de la aplicación involucrada(s) |
| http://localhost:5000/show\_all\_questions  http://localhost:5000/insert\_question |
| Tipo de vulnerabilidad |
| SQL Injection. |
| Causante de la vulnerabilidad |
| qbody = """INSERT INTO Questions(author, title, tags, body, time)  VALUES ('{0}','{1}','{2}','{3}',CURRENT\_TIMESTAMP)"""  query = qbody.format(author, title, tags, body)  cur.executescript(query)  La función executescript() de la insert\_question permite ejecutar varias sentencias de sql.  La sentencia de qbody no escapa la entrada. |
| Situaciones peligrosas o no deseadas que puede provocar |
| Por ello, además del Insert, el usuario puede ejecutar cualesquiera sentencias tras finalizar Insert, es decir, realizar cualquier acción con la BD. |
| Ejemplo paso a paso de cómo explotar la vulnerabilidad (con capturas de pantalla) |
| **Paso 1**: Introducir datos sin escapar.   * Autor: 1 * Título: 1 * Etiqueta: 1 * Pregunta: 1', 0); delete from questions;--   Imagen que contiene Diagrama  Descripción generada automáticamente  **Paso 2**: Insertar la pregunta en la BD.  Interfaz de usuario gráfica, Texto, Aplicación  Descripción generada automáticamente  **Paso 3**: Pulsar volver para ver todas las preguntas.  Imagen que contiene Interfaz de usuario gráfica  Descripción generada automáticamente  Una vez llegado a este paso, podemos visualizar que no tiene ninguna pregunta en la página de show\_all\_questions. Ya que fueron borrados tras delete from questions. |
| Medidas para mitigar la vulnerabilidad |
| * Escapar todas las entradas.   qbody = """INSERT INTO Replies(author,body,time,question\_id)  VALUES (:author, :body, CURRENT\_TIMESTAMP, :question\_id)"""  params = {'author': author, 'body': body, 'question\_id': question\_id}   * Cambiar el método executescript por execute.   cur.execute(qbody, params)   * Aplicar principios de mínimo privilegio y segregar usuarios, es decir, los usuarios deben tener únicamente los privilegios necesarios para su tarea y ninguno más y utilizar diferentes usuarios para los distintos accesos a la base de datos, no un solo usuario omnipotente. |

|  |
| --- |
| **INFORME DE VULNERABILIDAD** |
| Ruta(s) de la aplicación involucrada(s) |
| http://localhost:5000/show\_all\_questions  http://localhost:5000/search\_question |
| Tipo de vulnerabilidad |
| XSS Reflejado. |
| Causante de la vulnerabilidad |
| Por la falta de desinfectar los datos introducidos por el usuario (tag = request.args['tag']) y por la incorporación de ello en la página devuelta(return render\_template('messages\_search.html', questions=res, tag=tag)). |
| Situaciones peligrosas o no deseadas que puede provocar |
| Robo de cookies, redireccionamiento a sitios maliciosos, cambio de apariencia de la pagina web y robo de credenciales |
| Ejemplo paso a paso de cómo explotar la vulnerabilidad (con capturas de pantalla) |
| **Paso 1**: Introducir <script>alert("XSS Reflejado")</script> en la barra de búsqueda.    **Paso 2**: Pulsar el botón de buscar y aparece “XSS Reflejado”. |
| Medidas para mitigar la vulnerabilidad |
| * Escapar todas las entradas.   tag = html.escape(request.args['tag'])   * Desinfectar todo el texto que va a aparecer en la página HTML generada, tanto el que provenga de la petición como aquel obtenido de la BD.   En **search\_question()**  res = list(list())  for i in cur.fetchall():  aux = list()  for j in i:  aux.append(html.escape(j))  res.append(aux)  conn.close()  return render\_template('messages\_search.html', questions=res, tag=tag) |

|  |
| --- |
| **INFORME DE VULNERABILIDAD** |
| Ruta(s) de la aplicación involucrada(s) |
| http://localhost:5000/show\_all\_questions  http://localhost:5000/show\_question?id=3  http://localhost:5000/insert\_reply |
| Tipo de vulnerabilidad |
| XSS Persistente. |
| Causante de la vulnerabilidad |
| Por la falta de desinfectar los datos introducidos por el usuario (body = request.form['body'] y author = request.form['author']) y la incorporación de ello en la BD.  qbody = """INSERT INTO Replies(author,body,time,question\_id)  VALUES (:author, :body, CURRENT\_TIMESTAMP, :question\_id)"""  params = {'author': author, 'body': body, 'question\_id': question\_id}  cur.execute(qbody, params)  conn.commit()  Provoca que el servidor utilice dicho dato con código malicioso en la construcción de la propia página web (return render\_template("message\_detail.html", q=question, replies=replies, ident=ident)). |
| Situaciones peligrosas o no deseadas que puede provocar |
| Robo de cookies, redireccionamiento a sitios maliciosos, cambio de apariencia de la página web y robo de credenciales. |
| Ejemplo paso a paso de cómo explotar la vulnerabilidad (con capturas de pantalla) |
| **Paso 1**: Pulsar Ver de una pregunta.    **Paso 2**: Introducir <script>alert("XSS Persistente")</script> en el campo de Autor y pulsar el botón de Contestar.    **Paso 3**: Pulsar Volver y acudir a la pregunta respondida por el usuario. A partir de ello, cada acceso a dicha pregunta mostrará “XSS Persistente”. |
| Medidas para mitigar la vulnerabilidad |
| * Escapar todas las entradas.   author = html.escape(request.form['author'])  body = html.escape(request.form['body'])  question\_id = html.escape(request.form['question\_id'])   * Desinfectar todo el texto que va a aparecer en la página HTML generada, tanto el que provenga de la petición como aquel obtenido de la BD.   En **show\_question()**  question = list()  for i in cur.fetchone():  question.append(i)  cur.execute(qbody2, params)  replies = list(list())  for i in cur.fetchall():  aux = list()  for j in i:  aux.append(html.escape(j))  replies.append(aux)  conn.close()  return render\_template("message\_detail.html", q=question, replies=replies, ident=ident) |