Seguridad en Aplicaciones: Primera práctica

Máster Interuniversitario en Ciberseguridad

Curso académico 2023/2024

1. Introducción

La primera práctica de la asignatura consiste en el análisis de una aplicación web para localizar, explotar y solucionar las vulnerabilidades de seguridad que se estudiarán durante el curso.

Para la realización de esta práctica se proporciona una aplicación web tradicional que ha sido desarrollada en Java utilizando la arquitectura Modelo-Vista-Controlador.

El objetivo principal consiste en encontrar y solucionar el máximo número posible de vulnerabilidades y para 3 de esas vulnerabilidades será necesario generar un *exploit* que pueda causar un daño real.

Una vez finalizada la práctica es necesario presentar, por un lado, una nueva versión de la aplicación web con las vulnerabilidades arregladas y, por otro lado, un documento en el que se detallarán todas las vulnerabilidades detectadas, cómo se han solucionado, y para 3 de ellas, los *exploits* que han sido desarrollados.

En el desarrollo del laboratorio se realizarán varios apartados competitivos (*bonus*) con el objetivo de premiar a los grupos que sean capaces de localizar en primer lugar cada una de las vulnerabilidades.

2. Planteamiento

2.1 Visión global

La empresa *Amazoncillo* ha decidido abrir una sucursal de su tienda de comercio electrónico en nuestro país y para ello ha contratado el desarrollo de la aplicación web a la empresa *BonitoYBarato*.

Una vez terminado el desarrollo, justo antes de poner la tienda en línea, desde la empresa matriz solicitan un informe de seguridad que, por política de empresa, es obligatorio para todas las filiales. En la división local de *Amazoncillo* se dan cuenta de que se han olvidado de ese informe de seguridad, por lo que solicitan los servicios a la empresa especializada *Fortifícame*, líder mundial en seguridad.

Terminado el análisis preliminar, *Fortifícame* informa a *Amazoncillo* de que la cantidad de vulnerabilidades es tan alta que el coste de arreglarlas superaría ampliamente el coste del desarrollo inicial.

Como el presupuesto disponible no es suficiente para encargarle a *Fortifícame* que arregle las vulnerabilidades, la división local de *Amazoncillo* decide buscar otra alternativa, en este caso, opta por contratar los servicios de los alumnos del Máster Inter-Universitario en Ciberseguridad que se han ofrecido voluntarios para realizar el trabajo como parte de las prácticas de la asignatura de Seguridad en Aplicaciones.

La aplicación web ofrece las funcionalidades de una tienda de comercio electrónico tradicional:

- Permite a los usuarios darse de alta y modificar su perfil.
- Permite recordar los datos del usuario en el navegador local y la posibilidad de reestablecer la contraseña cuando el usuario se ha olvidado de ella.
- Los usuarios no registrados sólo podrán ver los productos sin posibilidad de realizar la compra.
- Por simplicidad, la cesta de la compra se almacena en la sesión del usuario, por lo tanto, si se reinicia el servidor estos datos se perderán.
- El usuario podrá realizar un pedido a partir de los productos de la cesta de la compra.
- El proceso de pago no está implementado, asumiendo que la transacción bancaria se realiza de forma correcta una vez que se formaliza el pedido.
- A la hora de pagar el pedido, es posible almacenar la tarjeta de crédito para futuras compras.
- No es necesario pagar el pedido en el momento de hacer la compra y se podrán cancelar los pedidos que todavía no han sido pagados.
- Los usuarios podrán realizar valoraciones de los productos que han comprado y podrán ver todas las valoraciones de un producto antes de comprarlo.

2.2 Detalles

La primera práctica de la asignatura consiste en analizar una aplicación web con el objetivo de localizar y solucionar el máximo número posible de vulnerabilidades. Además, para 3 de esas vulnerabilidades, será necesario proporcionar un *exploit* de calidad que cause el mayor daño posible (daño, engaño, robo de información, etc.).

Esta aplicación ha sido implementada en Java siguiendo la arquitectura Modelo-Vista-Controlador y puede contener vulnerabilidades en cualquiera de esos tres subsistemas.

- El modelo o lógica de negocio ha sido implementado utilizando Hibernate + JPA. Los datos se almacenan en una base de datos Derby local que ya está preconfigurada.
- El controlador ha sido implementado utilizando la librería Spring MVC.
- La capa vista utiliza plantillas desarrolladas con Thymeleaf además de JQuery y Bootstrap como librerías web.
- La aplicación se distribuye como un proyecto Maven y utiliza Spring Boot para simplificar la configuración y facilitar su ejecución.

Para solucionar una vulnerabilidad se debe buscar siempre la mejor solución posible, aplicando de forma adecuada los conocimientos que han sido adquiridos en las clases de teoría.

La entrega de la práctica consistirá en:

- El código fuente de la aplicación web con las vulnerabilidades corregidas. Se debe incluir el proyecto Maven, pero sin ficheros binarios ni compilados.
- La documentación detallada de las vulnerabilidades y de los *exploits*.

La documentación de las vulnerabilidades debe incluir los siguientes elementos:

- **Descripción y localización**: es necesario indicar de qué tipo de vulnerabilidad se trata, asociándola, si es posible, con alguna de las entradas del CWE. Es necesario indicar los daños concretos que puede provocar y también se debe indicar claramente en qué punto del proyecto se localiza el problema y qué ficheros se ven afectados:
    - Si se trata de una vulnerabilidad en el código fuente, se deben indicar los ficheros (Java, HTML, etc.) y las líneas exactas que se ven afectadas.
    - Si se trata de una vulnerabilidad que afecta a la configuración, se deben indicar los ficheros involucrados.
- **Solución**: es necesario indicar los principios teóricos utilizados para solucionar la incidencia. Si existen varias alternativas, se deberá elegir siempre la más correcta aplicando los conocimientos adquiridos en la clase de teoría.
- ***Exploit***: para 3 de las vulnerabilidades es necesario incluir la secuencia de pasos que se pueden utilizar para causar el mayor daño posible. Estos *exploits* deben ser de calidad y causar un daño real. Está permitido combinar varias vulnerabilidades para generar un *exploit* lo más efectivo posible.

También se podrán utilizar los informes de las herramientas DAST y SAST. En caso de que una vulnerabilidad sea encontrada de forma automática por la herramienta, se puede añadir esa información en la documentación.

Si se incluye alguna vulnerabilidad que no es tal, se penalizará de forma negativa en la nota final. Por ejemplo, los falsos positivos detectados por las herramientas DAST y SAST no deben incluirse en la lista de problemas detectados.

Se debe considerar la aplicación como terminada y lista para su puesta en funcionamiento. Esto implica que la configuración también debería estar preparada para entornos en producción. **Excepciones**:

- Para facilitar el desarrollo de la práctica, el proyecto se recarga automáticamente tras una modificación en cualquier fichero. Esta configuración no se considera una vulnerabilidad.
- Además, algunas categorías de log también se encuentran activadas de forma intencionada para facilitar la visualización de las excepciones. Esta configuración tampoco se considera una vulnerabilidad.
- El pago con tarjeta de crédito no está implementado y se considerará que la transacción siempre finaliza con éxito. Por lo tanto, no se considerarán vulnerabilidades en la gestión de la transacción bancaria.
- La aplicación no implementa mecanismos de control contra ataques por fuerza bruta, por lo tanto, tampoco se considerarán este tipo de vulnerabilidades.

2.3 Parte opcional *Bonus*

Como parte opcional de la práctica, se realizará una prueba competitiva entre todos los grupos. Esta parte opcional se tendrá en cuenta para subir la nota y no supondrá una penalización a los grupos que no participen.

Por un lado, se premiará al primer grupo que localice una vulnerabilidad en el código fuente de la aplicación. Si una vulnerabilidad ocurre en varios sitios de la aplicación, se premiará al primer grupo que encuentre una ocurrencia. Por ejemplo, si se trata de inyección de SQL, para obtener el *bonus* no es necesario encontrar todos puntos en los que se produce la vulnerabilidad. Las vulnerabilidades que afectan a la implementación del control de acceso, se consideran problemas diferentes por lo que se puede obtener un *bonus* por cada una de ellas.

Por otro lado, se premiará al primer grupo que reporte nuevas entradas en el CVE que afecten a alguna de las librerías que se usan en la aplicación web. Estas entradas en el CVE tienen que estar dadas de alta durante el período de realización de la práctica. Por lo tanto, no entran en la bonificación las entradas del CVE cuya fecha de alta sea anterior al inicio de la práctica o posterior a la fecha de entrega.

Para determinar el orden en los envíos, una vez localizado el problema, el grupo debe ponerlo en conocimiento del profesor de prácticas a través de un correo electrónico, aunque en ese momento se encuentren en la clase de prácticas y se lo pueda comunicar de forma presencial. No se tendrá en cuenta cualquier otro tipo de aviso.

3. Normativa y evaluación

3.1 Composición de los grupos

La práctica se realizará de forma preferente en grupos de 3 personas.

3.2 Formato de entrega

La entrega de la práctica debe incluir el código fuente de la aplicación web con las vulnerabilidades arregladas y un documento que detalle de forma clara cada uno de los problemas encontrados: el tipo de vulnerabilidad, las consecuencias que tiene, la localización, una descripción de la solución y, para las 3 vulnerabilidades seleccionadas, se indicará también el *exploit* que se puede utilizar para causar algún tipo de daño.

A continuación, se muestra un ejemplo de plantilla que se puede utilizar para la documentación.

Vulnerabilidad	**Inyección de código SQL**
CWE	CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
Consecuencias	Esta vulnerabilidad permite que cualquier usuario se autentique sin necesidad de conocer la contraseña. Para explotar la vulnerabilidad tan sólo es necesario conocer el nombre del usuario.
Localización	Login.java:326, UserDAO.java:27 localizados en el paquete Java "es.mucs.store.business.user"
Exploit(s)	1. Se accede al formulario de autenticación en la página /Login.html. 2. Se establece el usuario: admin' OR 1 = 1 -- 3. Se establece cualquier valor para la contraseña. 4. Se pulsa en el botón de autenticación y se accede con éxito a la cuenta del usuario administrador.
Solución	Como no es viable utilizar técnicas de "lista blanca", se ha utilizado la técnica de escapado de las entradas de usuario. Concretamente, se han escapado los siguientes caracteres especiales en SQL: el punto y coma, las comillas dobles y las comillas simples.
Otra información	La vulnerabilidad ha sido encontrada de forma automática por la herramienta OWASP ZAP.

3.3 Fecha de entrega

La fecha límite para presentar la práctica será el miércoles, **15 de noviembre**. Unos días antes de esa fecha, se publicarán instrucciones detalladas sobre la entrega.

La defensa se realizará en el horario de clase de prácticas los días 16 de noviembre (UDC) y el día 17 de noviembre (UVIGO).

3.4 Evaluación

Para la evaluación de la práctica se tendrán en cuenta las vulnerabilidades encontradas, la calidad de los 3 *exploits* desarrollados y las posibles bonificaciones.

3.4.1 Evaluación de las vulnerabilidades

Cada vulnerabilidad tendrá una puntuación de 0.5 sobre 10. No hay límite en el número de vulnerabilidades que se pueden presentar hasta un máximo de 7 puntos sobre 10.

A su vez, a cada vulnerabilidad se le asignará una puntuación individual que se obtendrá en base a:

- Localización y descripción (0.25 sobre 10): se valorará que la descripción sea lo más precisa posible y que se hayan encontrado todas las ocurrencias.
- Calidad de la solución (0.25 sobre 10): en el caso de existir varias soluciones, se tendrá en cuenta la calidad de la opción elegida y si se ajusta a los conocimientos adquiridos en las clases de teoría. Si la vulnerabilidad se soluciona de forma parcial, se valorará de manera positiva, pero en todo caso, la puntuación será inferior a 0.25. Es necesario documentar los fundamentos teóricos utilizados para solucionar el problema.

Importante: para obtener la puntuación máxima de una vulnerabilidad, es necesario encontrar y arreglar todas las ocurrencias de ésta. Por ejemplo, si se detecta que la aplicación se ve afectada por inyección de SQL en dos puntos diferentes, es necesario arreglar las dos ocurrencias para obtener la máxima puntuación. Esto mismo aplicaría al resto de vulnerabilidades como XSS, CSRF, inyección, etc.

Excepción: esta regla no se aplica a los errores que puedan existir en la implementación del control de acceso. El motivo de establecer este criterio es que, para la mayor parte de vulnerabilidades, una vez encontrada la primera ocurrencia, es muy sencillo encontrar las siguientes, pero las vulnerabilidades que afectan al control de acceso pueden tener una naturaleza completamente diferente de tal forma que el hecho de localizar una de ellas, no implica que sea sencillo localizar las demás. Además, estas vulnerabilidades requieren de un análisis exhaustivo de la implementación por lo que su dificultad es mayor. Ejemplo (ficticio): si la implementación del control de acceso permite a un usuario normal acceder a la zona de administración y también permite a un usuario modificar datos de otro, saltándose en ambos casos el control de acceso, se considerarán como dos vulnerabilidades completamente diferentes.

Si la documentación incluye una vulnerabilidad incorrecta (un falso positivo) supondrá una penalización de -0.5 puntos.

3.4.2 Evaluación de los *exploits*

Es necesario presentar 3 *exploits* y cada uno de ellos tendrá una puntuación máxima de 1 punto sobre 10. No pueden presentarse más de 3 y se valorará su calidad. Por ejemplo, cuanta más información sea capaz de robar y más críticos sean los datos extraídos, la puntuación obtenida será más alta.

3.4.3 Evaluación de los *bonus*

La puntuación de las bonificaciones será la siguiente:

- 0.1 sobre 10 para el primer grupo que reporte una vulnerabilidad en el código fuente de la aplicación.
- 0.25 sobre 10 para el primer grupo que reporte una entrada en el CVE relacionada con una de las librerías que se usan en la aplicación y cuya fecha de publicación se encuentre comprendida en el período de realización de la práctica.

El único límite que hay para presentar bonificaciones es que la nota máxima de la práctica es un 10.

La notificación de una vulnerabilidad que no es tal o de una entrada del CVE que no sigue los criterios establecidos, supondrá:

- Una penalización de 0.1 sobre 10 si se trata de una vulnerabilidad en el código fuente.
- Una penalización de 0.25 sobre 10 si se trata de una entrada del CVE.

Si la nota total de las bonificaciones fuese negativa, teniendo en cuenta los envíos correctos y los incorrectos, no se tendrían en cuenta para la nota final de la práctica.

Las vulnerabilidades que aparezcan en el CVE también se pueden utilizar como una de las vulnerabilidades de la aplicación, siempre y cuando se trate de un tipo de vulnerabilidad diferente a otras ya encontradas. Por ejemplo, si ya se ha encontrado en el código fuente una vía para la inyección de SQL, aunque aparezca esa vulnerabilidad reportada en el CVE a través de una librería de terceros, no se podrá incluir como una vulnerabilidad nueva.

3.4.4 Defensa

Una vez realizada la entrega de la práctica, se realizará una defensa de ésta. En la defensa se tendrá en cuenta la documentación presentada, junto con la aplicación web que contiene las vulnerabilidades corregidas.

No se tendrá en cuenta ninguna vulnerabilidad ni ningún *exploit* que no se encuentre en la documentación.

Durante la defensa se comprobará si las vulnerabilidades están correctamente identificadas y arregladas. También se probarán los *exploits* sobre la versión original de la aplicación.

Si durante la defensa de la práctica no es posible ejecutar la aplicación, no se tendrán en cuenta las vulnerabilidades arregladas.

La defensa será conjunta con todos los miembros del grupo, pero se realizarán preguntas de forma individualizada a cada alumno. Por lo tanto, en base a las respuestas obtenidas, la nota de cada uno de los miembros del grupo podría ser diferente. Si durante la defensa, un alumno no es capaz de responder de forma correcta a las preguntas, la práctica se considerará suspensa para ese alumno.

Si se detecta que alguna práctica ha sido copiada, ésta se considerará suspensa para todos los grupos involucrados.

4. Referencias

- OWASP: https://www.owasp.org/index.php/Main_Page
- CWE: https://cwe.mitre.org/
- Java: http://www.oracle.com/technetwork/java/javase/overview/index.html
- Apache Commons: https://commons.apache.org/
- Spring Boot: https://spring.io/projects/spring-boot
- Spring MVC: https://docs.spring.io/spring/docs/current/spring-framework-reference/web.html
- Hibernate: http://hibernate.org/
- Apache Derby: https://db.apache.org/derby/
- Thymeleaf: https://www.thymeleaf.org/
- JQuery: https://jquery.com/
- Bootstrap: https://getbootstrap.com/
- Sonar Qube: https://www.sonarqube.org/
- Find Security Bugs: https://find-sec-bugs.github.io/
- OWASP ZAP: https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project