README4.md

Índice de subtermas:

- README4.md
  - 6.13 Inyecciones NoSQL
  - 6.14 Inyecciones LDAP
    - 6.14.1 Ejercicio
  - 6.15 Ataques de Deserialización
  - 6.16 Inyecciones LaTex

6.13 Inyecciones NoSQL

Las inyecciones NoSQL son una vulnerabilidad de seguridad en las aplicaciones web que utilizan bases de datos NoSQL, como MongoDB, Cassandra y CouchDB, entre otras. Estas inyecciones se producen cuando una aplicación web permite que un atacante envíe datos maliciosos a través de una consulta a la base de datos, que luego puede ser ejecutada por la aplicación sin la debida validación o sanitización.

La inyección NoSQL funciona de manera similar a la inyección SQL, pero se enfoca en las vulnerabilidades específicas de las bases de datos NoSQL. En una inyección NoSQL, el atacante aprovecha las consultas de la base de datos que se basan en documentos en lugar de tablas relacionales, para enviar datos maliciosos que pueden manipular la consulta de la base de datos y obtener información confidencial o realizar acciones no autorizadas.

A diferencia de las inyecciones SQL, las inyecciones NoSQL explotan la falta de validación de los datos en una consulta a la base de datos NoSQL, en lugar de explotar las debilidades de las consultas SQL en las bases de datos relacionales.

A continuación, se proporciona el enlace al proyecto de Github que nos descargamos para poner en práctica esta vulnerabilidad:

- Vulnerable-Node-App: https://github.com/Charlie-belmer/vulnerable-node-app
- https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/NoSQL%20Injection
- script python nosqli.py

6.14 Inyecciones LDAP

Las inyecciones LDAP (Protocolo de Directorio Ligero) son un tipo de ataque en el que se aprovechan las vulnerabilidades en las aplicaciones web que interactúan con un servidor LDAP. El servidor LDAP es un directorio que se utiliza para almacenar información de usuarios y recursos en una red.

La inyección LDAP funciona mediante la inserción de comandos LDAP maliciosos en los campos de entrada de una aplicación web, que luego son enviados al servidor LDAP para su procesamiento. Si la aplicación web no está diseñada adecuadamente para manejar la entrada del usuario, un atacante puede aprovechar esta debilidad para realizar operaciones no autorizadas en el servidor LDAP.

Al igual que las inyecciones SQL y NoSQL, las inyecciones LDAP pueden ser muy peligrosas. Algunos ejemplos de lo que un atacante podría lograr mediante una inyección LDAP incluyen:

- Acceder a información de usuarios o recursos que no debería tener acceso.
- Realizar cambios no autorizados en la base de datos del servidor LDAP, como agregar o eliminar usuarios o recursos.
- Realizar operaciones maliciosas en la red, como lanzar ataques de phishing o instalar software malicioso en los sistemas de la red.

Para evitar las inyecciones LDAP, las aplicaciones web que interactúan con un servidor LDAP deben validar y limpiar adecuadamente la entrada del usuario antes de enviarla al servidor LDAP. Esto incluye la validación de la sintaxis de los campos de entrada, la eliminación de caracteres especiales y la limitación de los comandos que pueden ser ejecutados en el servidor LDAP.

También es importante que las aplicaciones web se ejecuten con privilegios mínimos en la red y que se monitoreen regularmente las actividades del servidor LDAP para detectar posibles inyecciones.

A continuación, se proporciona el enlace directo al proyecto de Github que nos descargamos para desplegar un laboratorio práctico donde poder ejecutar esta vulnerabilidad:

- LDAP-Injection-Vuln-App: https://github.com/motikan2010/LDAP-Injection-Vuln-App
- LDAP: Qué es y para qué se utiliza este protocolo https://www.profesionalreview.com/2019/01/05/ldap/

6.14.1 Ejercicio

Instalamos openldap con docker:

```
docker run -p 389:389 --name my-openldap-container --detach
osixia/openldap:1.2.2
```

Descargamos el repositorio de LDAP-Injection-Vuln-App, construimos la imagen y la ejecutamos:

```
git clone https://github.com/motikan2010/LDAP-Injection-Vuln-App && cd
LDAP-Injection-Vuln-App
docker run -p 389:389 --name openldap-container --detach
osixia/openldap:1.2.2
docker build -t ldap-client-container .
docker run -dit --link openldap-container -p 8888:80 ldap-client-container
```

Con el siguiente comando ya podríamos extraer información del servidor LDAP:

```
ldapsearch -x -H ldap://localhost -b dc=example,dc=org -D
"cn=admin,dc=example,dc=org" -w admin 'cn=admin'
```

(Si no lo tienes instalado: `sudo apt-get install slapd ldap-utils -y`)

Con el anterior comando, estamos buscando el usuario admin en el servidor LDAP. Si el usuario admin existe, el servidor LDAP devolverá información sobre el usuario. Si el usuario admin no existe, el servidor

LDAP devolverá un mensaje de error.

Con nmap y utilizando scripts podríamos obtener información del servidor LDAP:

```
nmap -sV --script ldap\* -p 389 localhost
```

Ahora podemos jugar a concatenar filtros para obtener información del servidor LDAP:

```
ldapsearch -x -H ldap://localhost -b dc=example,dc=org -D
"cn=admin,dc=example,dc=org" -w admin '(&(cn=admin)(description=LDAP*))'
```

Entramos en el contenedor openldap:

```
docker exec -it openldap-container bash
```

Y buscamos el fichero `new-user.ldif` en la ruta `/container/service/slapd/assets/test`.

Lo copiamos en local y cambiamos los datos a nuestro antojo:

```
dn: uid=invent,dc=example,dc=org
uid: invent
cn: invent
sn: 3
objectClass: top
objectClass: posixAccount
objectClass: inetOrgPerson
loginShell: /bin/bash
homeDirectory: /home/invent
uidNumber: 14583102
gidNumber: 14564100
userPassword: invent123
mail: invent@example.org
description: Uno que pasaba por aquí
telephoneNumber: 657849302
```

Creamos desde anfitrión el nuevo usuario referenciando el nuevo fichero:

```
ldapadd -x -H ldap://localhost -D "cn=admin,dc=example,dc=org" -w admin -f
newuser.ldiff
```

Ahora podremos ver el nuevo usuario creado con el siguiente comando:

```
ldapsearch -x -H ldap://localhost -b dc=example,dc=org -D
"cn=admin,dc=example,dc=org" -w admin
```

Bien, pues creamos un par más y le damos caña al LDAP Injection con el script de Python ldapi.py.

6.15 Ataques de Deserialización

Los ataques de deserialización son un tipo de ataque que aprovecha las vulnerabilidades en los procesos de serialización y deserialización de objetos en aplicaciones que utilizan la programación orientada a objetos (POO).

La serialización es el proceso de convertir un objeto en una secuencia de bytes que puede ser almacenada o transmitida a través de una red. La deserialización es el proceso inverso, en el que una secuencia de bytes es convertida de nuevo a un objeto. Los ataques de deserialización ocurren cuando un atacante puede manipular los datos que se están deserializando, lo que puede llevar a la ejecución de código malicioso en el servidor.

Los ataques de deserialización pueden ocurrir en diferentes tipos de aplicaciones, incluyendo aplicaciones web, aplicaciones móviles y aplicaciones de escritorio. Estos ataques pueden ser explotados de varias maneras, como:

- Modificar el objeto serializado antes de que sea enviado a la aplicación, lo que puede causar errores en la deserialización y permitir que un atacante ejecute código malicioso.
- Enviar un objeto serializado malicioso que aproveche una vulnerabilidad en la aplicación para ejecutar código malicioso.
- Realizar un ataque de “man-in-the-middle” para interceptar y modificar el objeto serializado antes de que llegue a la aplicación.

Los ataques de deserialización pueden ser muy peligrosos, ya que pueden permitir que un atacante tome el control completo del servidor o la aplicación que está siendo atacada.

Para evitar estos ataques, es importante que las aplicaciones validen y autentiquen adecuadamente todos los datos que reciben antes de deserializarlos. También es importante utilizar bibliotecas de serialización y deserialización seguras y actualizar regularmente todas las bibliotecas y componentes de la aplicación para corregir posibles vulnerabilidades.

A continuación se os proporciona el enlace directo a la máquina de Vulnhub donde explotamos un ‘PHP Deserialization Attack‘:

- Máquina Cereal 1: https://www.vulnhub.com/entry/cereal-1,703/

6.16 Inyecciones LaTex

Las inyecciones LaTeX son un tipo de ataque que se aprovecha de las vulnerabilidades en las aplicaciones web que permiten a los usuarios ingresar texto formateado en LaTeX. LaTeX es un sistema de composición de textos que se utiliza comúnmente en la escritura académica y científica.

Los ataques de inyección LaTeX ocurren cuando un atacante ingresa código LaTeX malicioso en un campo de entrada de texto que luego se procesa en una aplicación web. El código LaTeX puede ser diseñado para

aprovechar vulnerabilidades en la aplicación y ejecutar código malicioso en el servidor.

Un ejemplo de una inyección LaTeX podría ser un ataque que aprovecha la capacidad de LaTeX para incluir gráficos y archivos en una aplicación web. Un atacante podría enviar un código LaTeX que incluya un enlace a un archivo malicioso, como un virus o un troyano, que podría infectar el servidor o los sistemas de la red.

Para evitar las inyecciones LaTeX, las aplicaciones web deben validar y limpiar adecuadamente los datos que se reciben antes de procesarlos en LaTeX. Esto incluye la eliminación de caracteres especiales y la limitación de los comandos que pueden ser ejecutados en LaTeX.

También es importante que las aplicaciones web se ejecuten con privilegios mínimos en la red y que se monitoreen regularmente las actividades de la aplicación para detectar posibles inyecciones. Además, se debe fomentar la educación sobre la seguridad en el uso de LaTeX y cómo evitar la introducción de código malicioso.

A continuación, se os proporciona el enlace correspondiente al proyecto de Github que nos descargamos para desplegar un laboratorio vulnerable donde poder practicar esta vulnerabilidad:

- Laboratorio LaTeX Injection: https://github.com/internetwache/Internetwache-CTF-2016/tree/master/tasks/web90/code

```
apt install -y zathura latexmk rubber
```

README.md principal

README3.md <--> README5.md