Another open-source project similar to Ericom Shield, which focuses on secure web browsing and remote browser isolation, is \*\*Secure Remote Browsing (SRB)\*\*. SRB is an open-source implementation that allows users to browse the internet securely by isolating web content in a remote environment. This ensures that any potential malware or threats are contained and do not affect the user's local machine.

Here's how you can set up SRB using Docker on AWS:

### Prerequisites

1. \*\*AWS Account\*\*: Ensure you have an AWS account.

2. \*\*Docker Installed\*\*: Ensure Docker is installed on your local machine for building and testing images.

3. \*\*AWS CLI Installed\*\*: Ensure AWS CLI is installed and configured on your local machine.

### Step-by-Step Guide

#### 1. Set Up an AWS EC2 Instance

- Launch an EC2 instance on AWS.

- Choose an instance type (e.g., t2.medium or larger as SRB might require more resources).

- Configure security group to allow inbound traffic on necessary ports (e.g., 80 for HTTP, 443 for HTTPS, and 22 for SSH).

- Connect to your EC2 instance via SSH.

#### 2. Install Docker on EC2 Instance

SSH into your EC2 instance and run the following commands to install Docker:

For Ubuntu:

```sh

sudo apt-get update -y

sudo apt-get install -y docker.io

sudo systemctl start docker

sudo systemctl enable docker

sudo usermod -aG docker $USER

```

Log out and log back in to apply the usermod changes.

#### 3. Set Up Secure Remote Browsing

There are various implementations of secure remote browsing. One example is using the open-source \*\*Cuckoo Sandbox\*\* for analyzing potentially malicious web pages. However, it's more of an analysis tool than a browsing tool. For pure browsing isolation, consider using \*\*Open Source Browser Isolation (OSBI)\*\* solutions. Below is a simplified setup using a Docker container that isolates the browser:

```sh

# Pull the necessary Docker image (an example image for browser isolation)

sudo docker pull jlesage/firefox

# Run the container

sudo docker run -d --name secure-firefox -p 5800:5800 -v /docker/appdata/firefox:/config:rw jlesage/firefox

```

This command pulls a Docker image with Firefox isolated and accessible via a VNC interface on port 5800.

#### 4. Access the Secure Browser

- Open your browser and navigate to `http://<your-ec2-instance-public-ip>:5800`.

- You should see a VNC interface with the isolated Firefox browser.

#### 5. Configure Security Settings

- Ensure that only authorized users can access the VNC interface by setting up authentication.

- Configure firewall rules to restrict access to the VNC port.

#### 6. Test and Verify

- Test browsing various websites to ensure that the isolation works as expected.

- Verify that no local resources are accessed by the remote browser.

### Conclusion

Setting up a Secure Remote Browsing (SRB) solution using Docker on AWS provides a robust environment for safe web browsing by isolating potential threats. While SRB is not a direct one-to-one replacement for Ericom Shield, it provides similar functionalities for browser isolation. Implementing SRB helps in securing your browsing experience and protecting your local system from web-based threats.

#########The file was splitted to upload to Github - now to cancat locally and load ##########

Yes, you can split a large file into smaller parts and upload them to GitHub. Once you have the parts on GitHub, you can download them and reassemble the original file. Here's how you can do it:

### Step-by-Step Guide

#### Splitting the File

1. \*\*Split the Docker Image Tar File\*\*

- You can use the `split` command to split the file into smaller parts. For example, to split the file into 100MB parts:

```sh

split -b 100M firefox\_image.tar firefox\_image\_part\_

```

This command will create files named `firefox\_image\_part\_aa`, `firefox\_image\_part\_ab`, etc.

#### Uploading the Parts to GitHub

1. \*\*Navigate to Your Repository Directory\*\*

- Ensure you are in the directory of your cloned GitHub repository:

```sh

cd ~/docker-firefox-image/

```

2. \*\*Move the Split Parts to the Repository Directory\*\*

- Move the split parts to the repository directory if they are not already there. For example:

```sh

mv /path/to/firefox\_image\_part\_\* .

```

3. \*\*Add the Split Parts to the Repository\*\*

- Add the split parts to the Git repository:

```sh

git add firefox\_image\_part\_\*

```

4. \*\*Commit the Changes\*\*

- Commit the split parts to the repository:

```sh

git commit -m "Add split parts of Docker image tar file"

```

5. \*\*Push the Changes to GitHub\*\*

- Push the split parts to GitHub:

```sh

git push origin master

```

#### Downloading and Reassembling the File

Once you have the parts on GitHub, you can download and reassemble them:

1. \*\*Clone the Repository or Download the Parts\*\*

- Clone your repository to the destination machine:

```sh

git clone https://github.com/yourusername/docker-firefox-image.git

cd docker-firefox-image

```

2. \*\*Reassemble the Parts\*\*

- Use the `cat` command to concatenate the parts back into a single file:

```sh

cat firefox\_image\_part\_\* > firefox\_image.tar

```

3. \*\*Load the Docker Image\*\*

- Load the Docker image from the reassembled tar file:

```sh

sudo docker load -i firefox\_image.tar

```

Let's say you want to open permissions for a folder named myfolder.

1. **Using Numerical Mode**:

bash

Копировать код

chmod 777 myfolder

This command will give full permissions to everyone (owner, group, others).

### Full Command Sequence

Here’s a full sequence for splitting, uploading, downloading, and reassembling:

#### Splitting and Uploading

1. \*\*Split the File\*\*

```sh

split -b 100M firefox\_image.tar firefox\_image\_part\_

```

2. \*\*Move to the Repository Directory\*\*

```sh

cd ~/docker-firefox-image/

mv /path/to/firefox\_image\_part\_\* .

```

3. \*\*Add and Commit the Parts\*\*

```sh

git add firefox\_image\_part\_\*

git commit -m "Add split parts of Docker image tar file"

git push origin master

```

#### Downloading and Reassembling

1. \*\*Clone the Repository\*\*

```sh

git clone https://github.com/yourusername/docker-firefox-image.git

cd docker-firefox-image

```

2. \*\*Reassemble the Parts\*\*

```sh

cat firefox\_image\_part\_\* > firefox\_image.tar

```

3. \*\*Load the Docker Image\*\*

```sh

sudo docker load -i firefox\_image.tar

```

By following these steps, you can handle large files by splitting them into smaller parts, uploading them to GitHub, and then reassembling them when needed.