Sri Lanka Institute of Information Technology

Bug Bounty - Report 02

Absence of Anti-CSRF Tokens

Zooplus.de

Student Name – Wanasinghe N.K

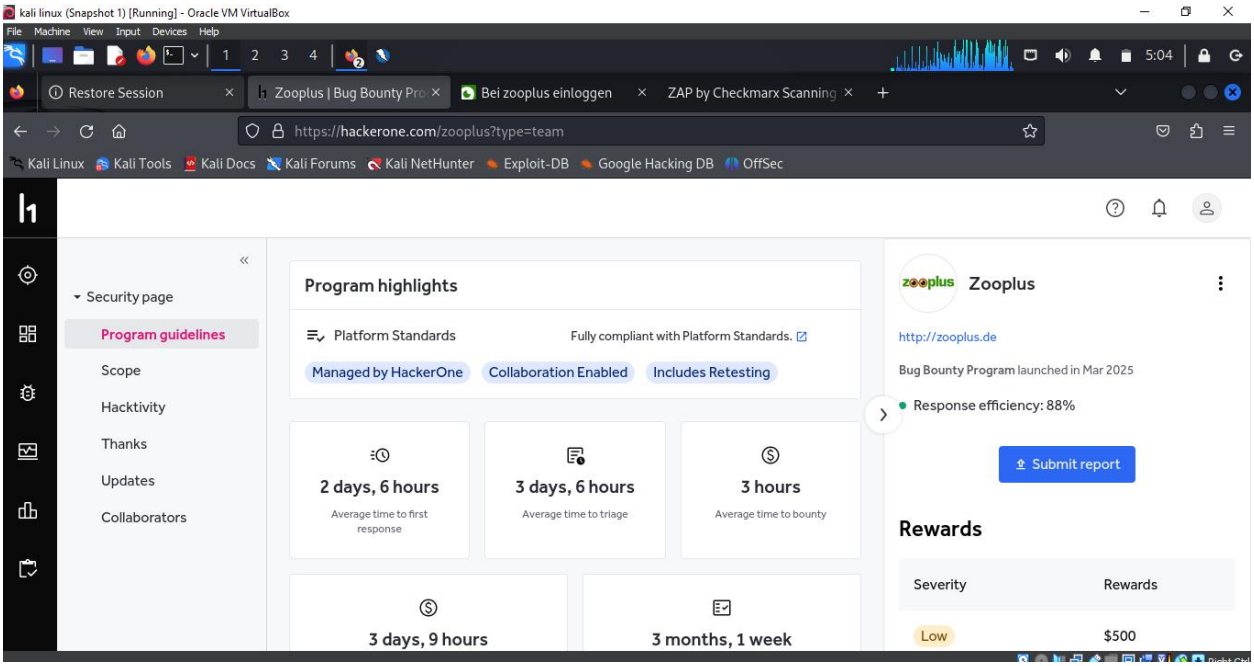
Student ID – IT23221000

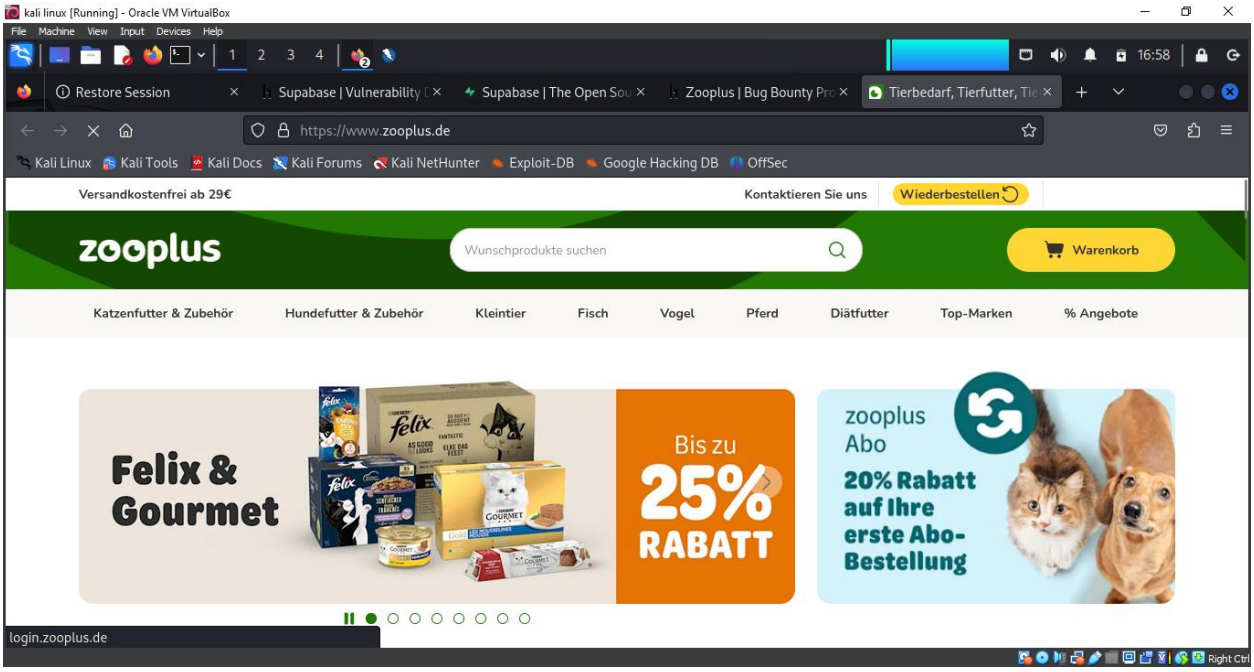
IE2062 - Web Security

B.Sc. (Hons) in information Technology Specializing in Cyber Security

Report 02 – zooplus.de (Hackerone)

Main domain – www.zooplus.de





I used OWASP ZAP tool to scan the website

In scope domains

http://zooplus.de/

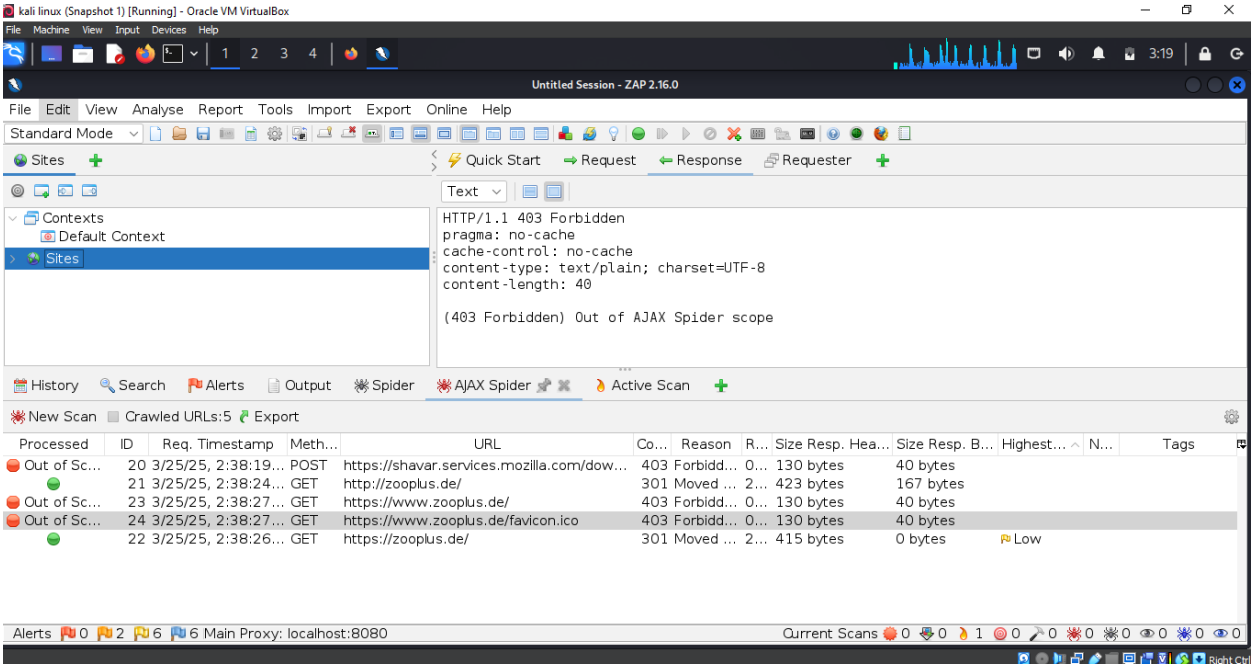
https://zooplus.de/

Out of scope domains

https://shavar.services.mozilla.com/downloads?client=navclient-auto-ffox&appver=115.13&pver=2.2

https://www.zooplus.de/

https://www.zooplus.de/favicon.ico



Reconnaissance: Gather information about the target.

Nmap – Network scanning and enumeration

I found all the open ports and detected the running services on the target server using Nmap.

```
┌──(nelushi㉿kali)-[~]
└─$ nmap -Pn -sV zooplus.de
Starting Nmap 7.95 ( https://nmap.org ) at 2025-05-01 14:40 CDT
Nmap scan report for zooplus.de (3.160.196.113)
Host is up (0.36s latency).
Other addresses for zooplus.de (not scanned): 3.160.196.20 3.160.196.62 3.160.196.110
rDNS record for 3.160.196.113: server-3-160-196-113.mrs52.r.cloudfront.net
Not shown: 998 filtered tcp ports (no-response)
PORT    STATE SERVICE  VERSION
80/tcp  open  http     Amazon CloudFront httpd
443/tcp open  ssl/http Amazon CloudFront httpd

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 79.36 seconds
```

Amass – Subdomain and DNS mapping

I found all the subdomains related to the target domain using Amass.

```
┌──(nelushi㉿kali)-[~]
└─$ amass enum -d zooplus.de
zooplus.de (FQDN) --> ns_record --> ns-1251.awsdns-28.org (FQDN)
zooplus.de (FQDN) --> ns_record --> ns-1870.awsdns-41.co.uk (FQDN)
zooplus.de (FQDN) --> ns_record --> ns-282.awsdns-35.com (FQDN)
zooplus.de (FQDN) --> ns_record --> ns-922.awsdns-51.net (FQDN)
zooplus.de (FQDN) --> mx_record --> mail02.mail-gw.de (FQDN)
zooplus.de (FQDN) --> mx_record --> mail03.mail-gw.de (FQDN)
clicks.zooplus.de (FQDN) --> cname_record --> clicks-zooplus-de.catip.ext.aws.zooplus.io (FQDN)
login-dev.zooplus.de (FQDN) --> cname_record --> login-dev-zooplus-de.idpc-dev.ext.aws.zooplus.io (FQDN)
mail05.zooplus.de (FQDN) --> a_record --> 62.209.48.170 (IPAddress)
dii3.zooplus.de (FQDN) --> cname_record --> zooplus.tt.omtrdc.net (FQDN)
tracker.zooplus.de (FQDN) --> cname_record --> master.jiraprod.int.aws.zooplus.io (FQDN)
t.zooplus.de (FQDN) --> cname_record --> shop-wall-prod-zooplus-de.shopwallp.ext.aws.zooplus.io (FQDN)
www.zooplus.de (FQDN) --> cname_record --> shop-wall-prod-zooplus-de.shopwallp.ext.aws.zooplus.io (FQDN)
62.209.32.0/19 (Netblock) --> contains --> 62.209.48.170 (IPAddress)
```

Wafw00f – Firewall Detection

Command used – wafw00f https://www.zooplus.de

```
                    ___
                   /   \
                  (  W00f! )
                   \  ___/
                  ,,    __
             |`-.__   / /
             /"  _/  /_/
            *===*    /
           /     )__//
          /|  /   /___`
          \\/`   \ |
           `\    /_\\_
            `    `  `
                 ____

404 Hack Not Found
                    ___   ___
                   \   \ /   /
                    \   X   /     405 Not Allowed
                   /   / \   \
                  /___/   \___\
                403 Forbidden
                    ___   ___
                   \   \ /   /
                    \   X   /
                   /   / \   \
                  /___/   \___\
502 Bad Gateway                   500 Internal Error

                  ~ WAFW00F : v2.2.0 ~
      The Web Application Firewall Fingerprinting Toolkit

[*] Checking https://www.zooplus.de
[+] The site https://www.zooplus.de is behind Cloudfront (Amazon) WAF.
[~] Number of requests: 2
```

Whatweb – to identify the technologies used by the site.

Commans used – whatweb https://www.zooplus.de

```
┌──(nelushi㉿kali)-[~]
└─$ whatweb https://www.zooplus.de
https://www.zooplus.de [200 OK] Cookies[sid], Country[UNITED STATES][US], Email[service@zooplus.de,zooplus@2x.png], HTML5, HTTPServer[istio-envoy], HttpOnly[sid], IP
[3.160.196.62], JQuery[3.6.0], Meta-Author[zooplus SE], Open-Graph-Protocol, Script[application/json], Title[play], UncommonHeaders[access-control-allow-origin,x-env
oy-upstream-service-time,x-content-type-options,x-lambda-region,x-stream-status,x-amz-cf-pop,alt-svc,x-amz-cf-id], Via-Proxy[1.1 247137278488ab1b89e4a784ee1baf22.clo
udfront.net (CloudFront)], X-Frame-Options[SAMEORIGIN], X-Powered-By[Next.js], X-UA-Compatible[IE=edge], X-XSS-Protection[1; mode=block]
```

Vulnerability 01

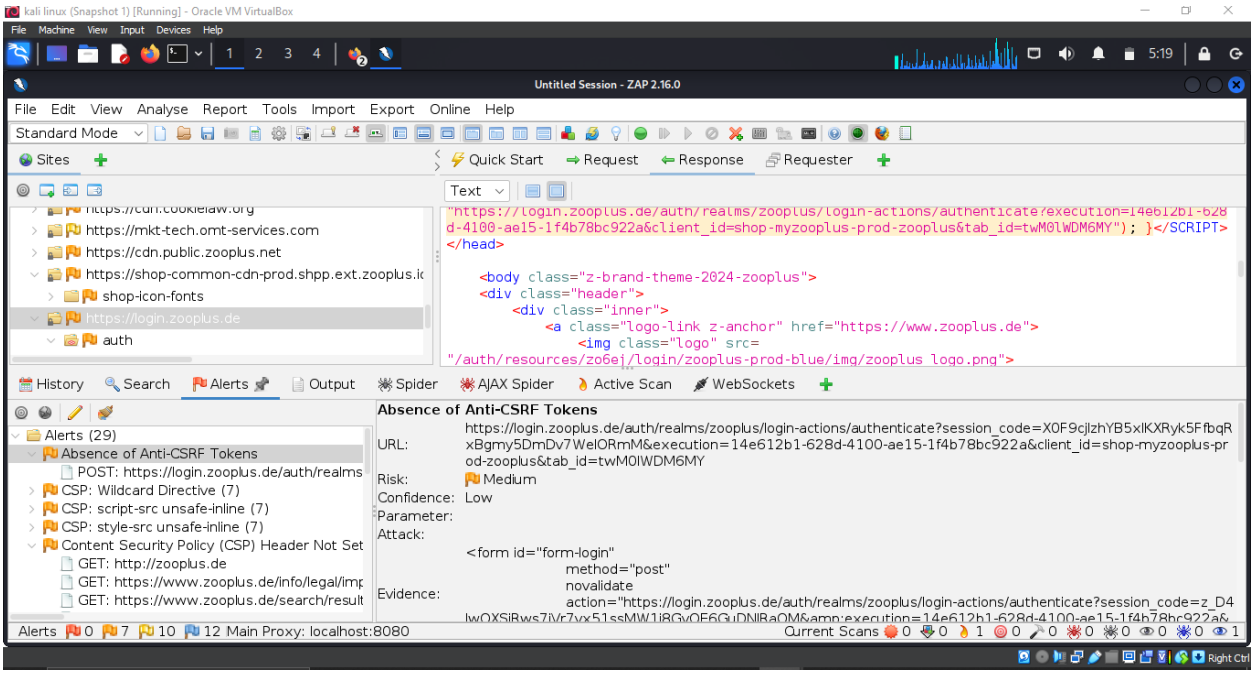
Domain

Login page of zooplus.de

https://login.zooplus.de/auth/realms/zooplus/protocol/openid-connect/auth?response_type=code&client_id=shop-myzooplus-prod-zooplus&redirect_uri=https%3A%2F%2Fwww.zooplus.de%2Fweb%2Fsso-myzooplus%2Flogin&state=983828ef-4a17-461a-9dd9-d2b70b7d61a0&login=true&ui_locales=de-DE&scope=openid

Vulnerability title

Absence of Anti-CSRF Tokens



Vulnerability description

No Anti-CSRF tokens were found in a HTML submission form.

Cross-Site Request Forgery (CSRF) is a type of attack where the victim is misled into sending an HTTP request to a target website without the victim's intent. In here, the malicious request forces the victim's browser to perform an action on a website where the victim is already authenticated.

The vulnerability is that an application allows predictable and repeatable URL or form actions to be executed that can lead to unauthorized action being taken to exploit what a website trusts the user to do.

CSRF attacks are notable because they demonstrate a weakness in the trust a site has for the user rather than how Cross-Site Scripting (XSS) exploits trust a user has for a site.

When Are CSRF Attacks Effective?

- CSRF attacks are effective when the victim has an active session on the target website.
- The victim is authenticated via HTTP auth on the target site.
- The victim is on the same local network as the target site.

An Attacker can exploit the active session of the victim to perform unauthorized actions, such as transferring money, changing account settings, without the victim being aware**.**

Affected components

The login form has no CSRF token (like <input type="hidden" name="csrf_token" ...>).

Sensitive actions can be submitted without origin validation.

Impact assessment

Severity – Medium

- On the Victim's Behalf, an attacker may,

  Alter account settings, such as email or password

  Make unauthorized purchases

  Transfer money into online banking

  Delete or change sensitive information

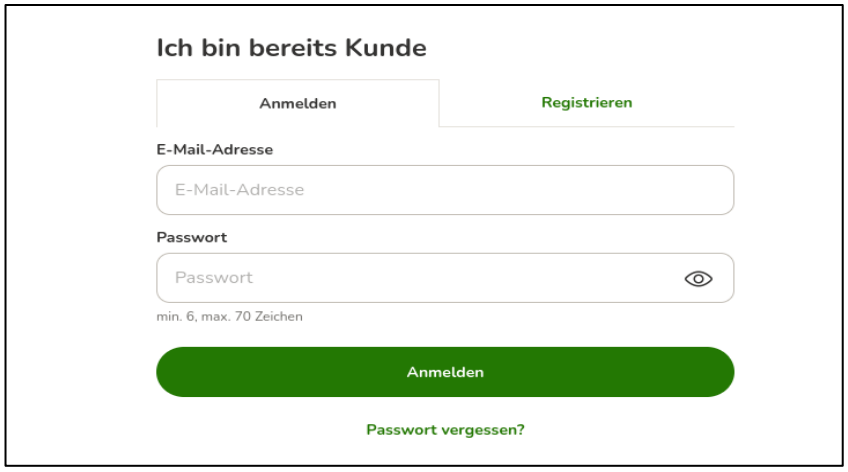
  Send emails/messages from their account.

- Exploit User Trust - Take advantage of the site's trust in the user's browser to perform malicious actions.
- Take Control of the User's Account - Change account settings to lock the legitimate user out.
- Perform Malicious Actions in the Background - Submit forms or change configuration without the victim's awareness.

Steps to reproduce with Proof of Concept (poc)

1. First, I Inspected the Target Website

- I Opened the target website in my browser. (the login page of zooPlus.de)
- I Focused on the login form and I logged in and now my session is active.

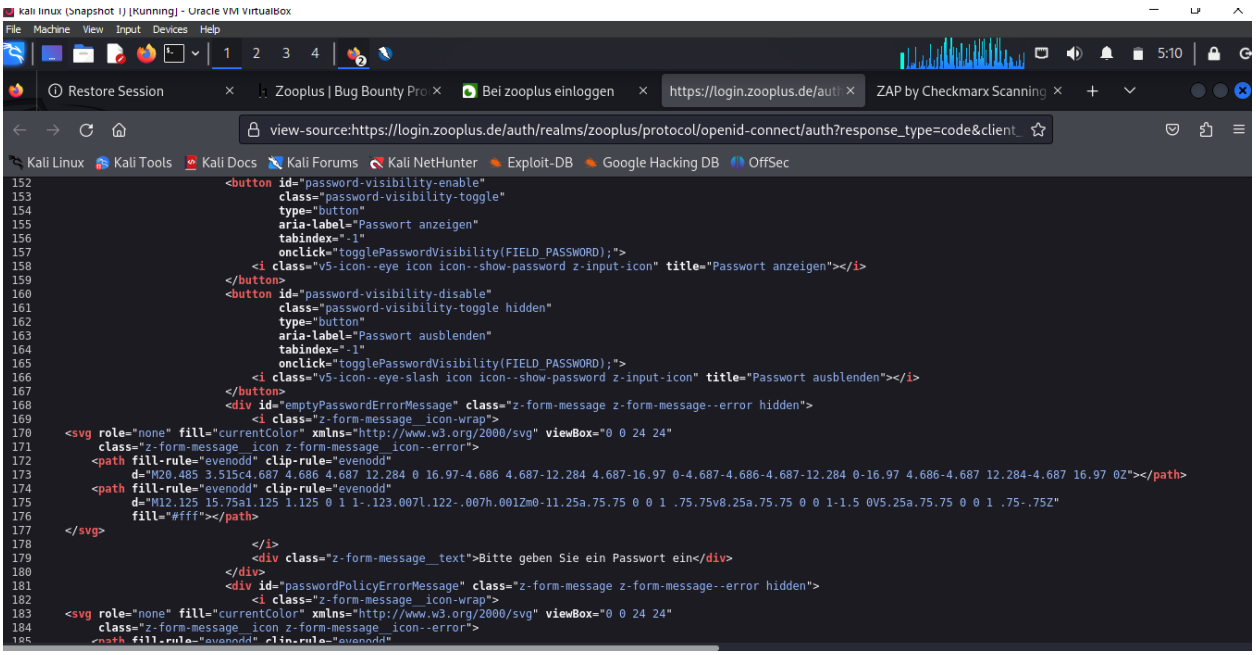


2. Next, I Checked for CSRF Token in Forms

- In the Developer Tools, I went to the Elements tab.
- Looked for hidden form fields or cookies that contain a CSRF token.

Typically, CSRF tokens are stored in a hidden input field in a form:

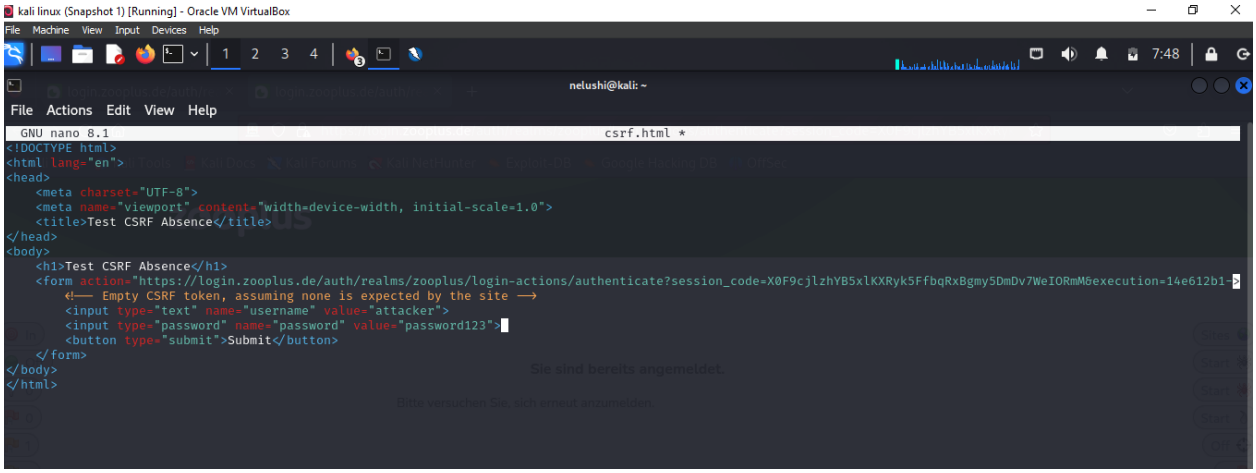
<input type="hidden" name="csrf_token" value="RANDOMCSRFVALUE">



- The form does not contain any hidden CSRF token field like the one above, the site might not be using CSRF protection.

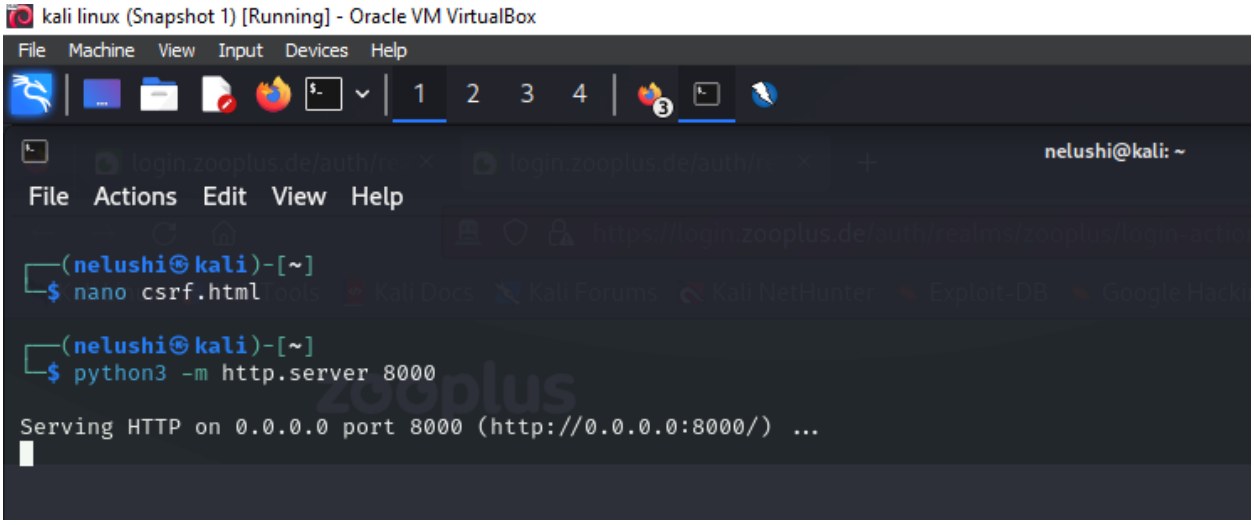
3. Create a Test HTML Page (Simulate Cross-Site Request)

- Next, I Created a simple HTML page that will submit data to the target website from another origin. This tests whether the website is vulnerable to CSRF attacks.



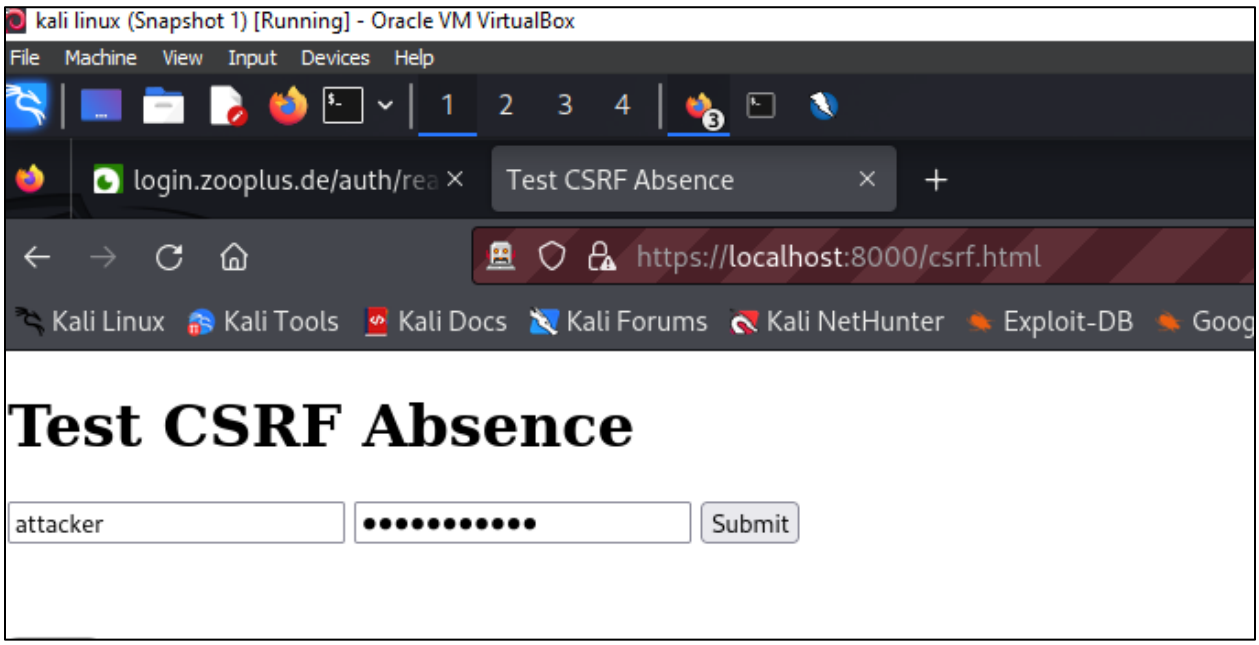
4. After that, I started a local web server to run the html code.

python3 -m http.server 8000

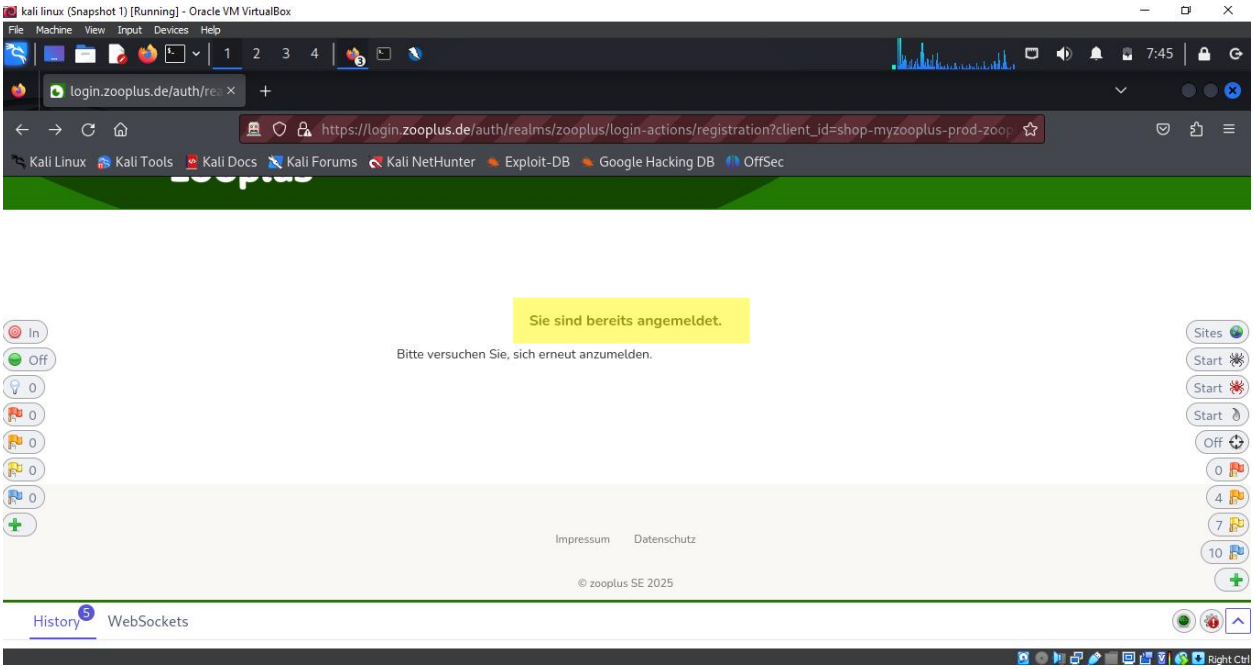


- Then I opened firefox and searched this to run the html code.

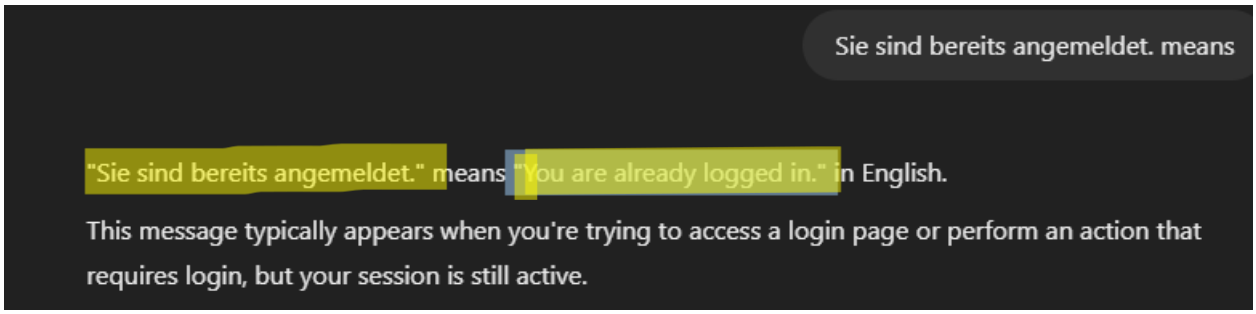
http://localhost:8000/csrf.html



After submitting this attacker credentials, it directed me to the logged page as I was the victim.

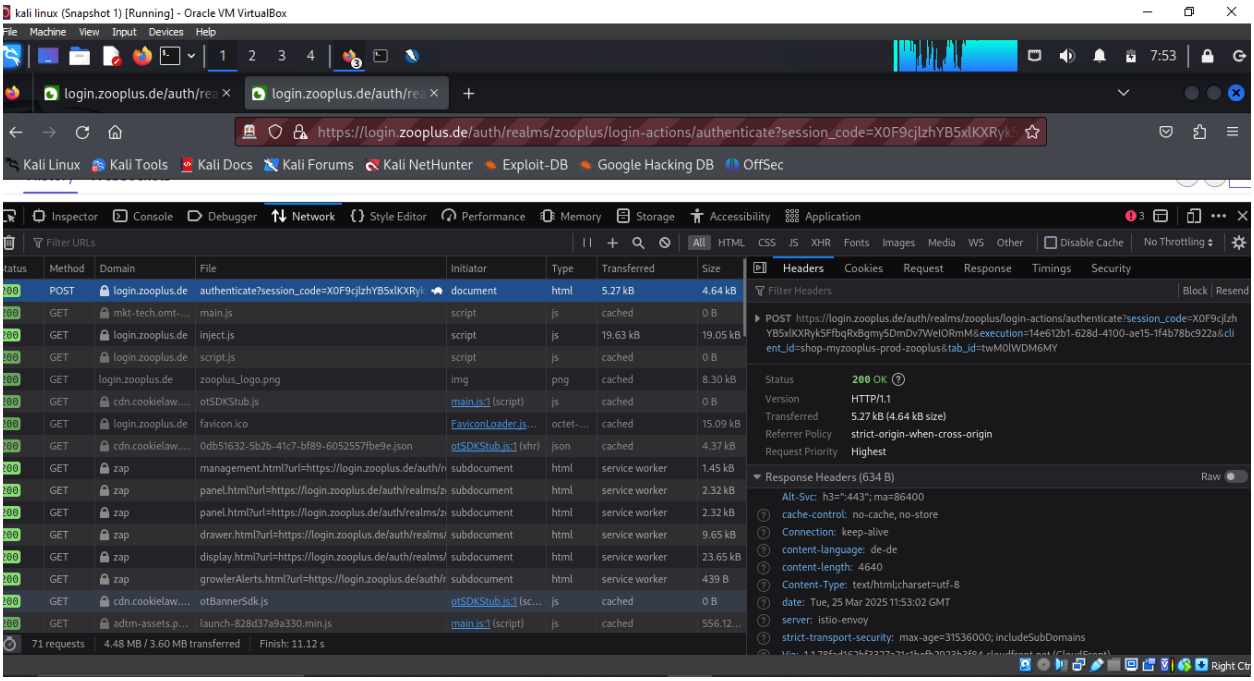


This is a german website. Therefore,



The website's server procceded the request as it came from the victim. I Looked for the response of the POST request made.

The request is successfully processed (200 OK) meaning that the website **does not have CSRF protection**.



An Example of Attack in Action

The victim is logged in to zooplus.de login page.

The attacker sends a link to the victim directing them to the HTML page I created.

The victim will click the link, and the form submission occurs and the target website processes the form submission as if the victim submitted it, and at this point, the victim's account has been compromised without them knowing.

Keys and values of the alert

OWASP_2021_A01	https://owasp.org/Top10/A01_2021-Broken_Access_Control/
WSTG-v42-SESS-05	https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/06-Session_Management_Testing/05-Testing_for_Cross_Site_Request_Forgery

Lack of Anti-CSRF tokens is under **OWASP 2021 A01: Broken Access Control** and **WSTG-v42-SESS-05**, because it allows attackers to trick authenticated users to conduct unauthorized activities. Users may be manipulated into sending unintended requests by lack of CSRF protection, leading to account compromise or privilege escalation**.**

Proposed mitigation or fix

Leverage CSRF Tokens - Implement a unique CSRF token in every form and validate it server-side for all requests that modify the state of the server.

Leverage Same-Site Cookies - When cookies are set, set the SameSite attribute (Strict or Lax) to block requests that are cross-origin.

Use Header Checks - Verify the HTTP referer and origin header to make sure they fall to the domain you are serving.

Use CSRF Tokens in AJAX - Pass the CSRF token as an ajax header for AJAX requests.

Rotate Tokens - Changing tokens periodically helps limit reuse.

Regularly Test - Regularly test for CSRF vulnerabilities with something like OWASP ZAP