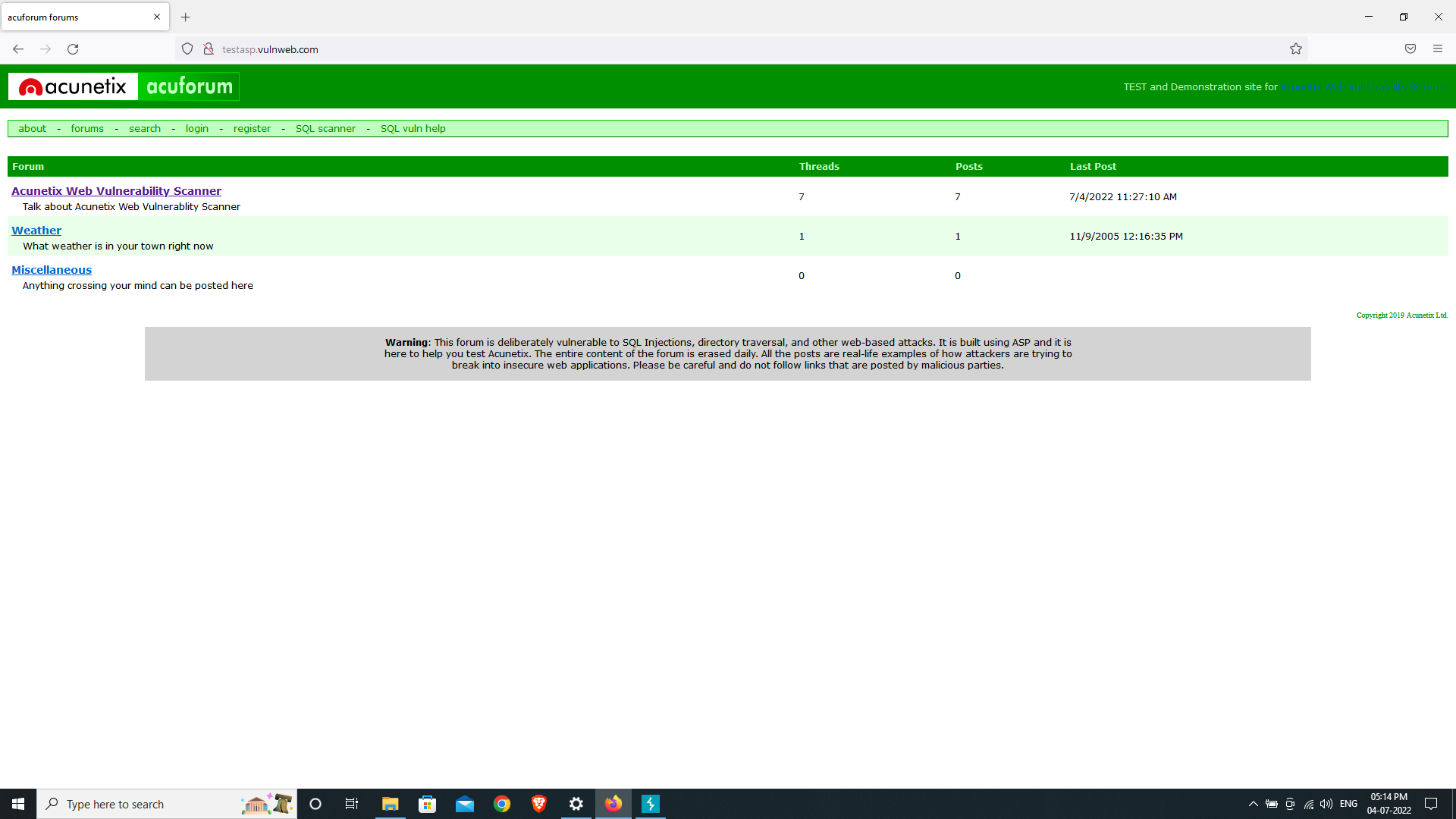
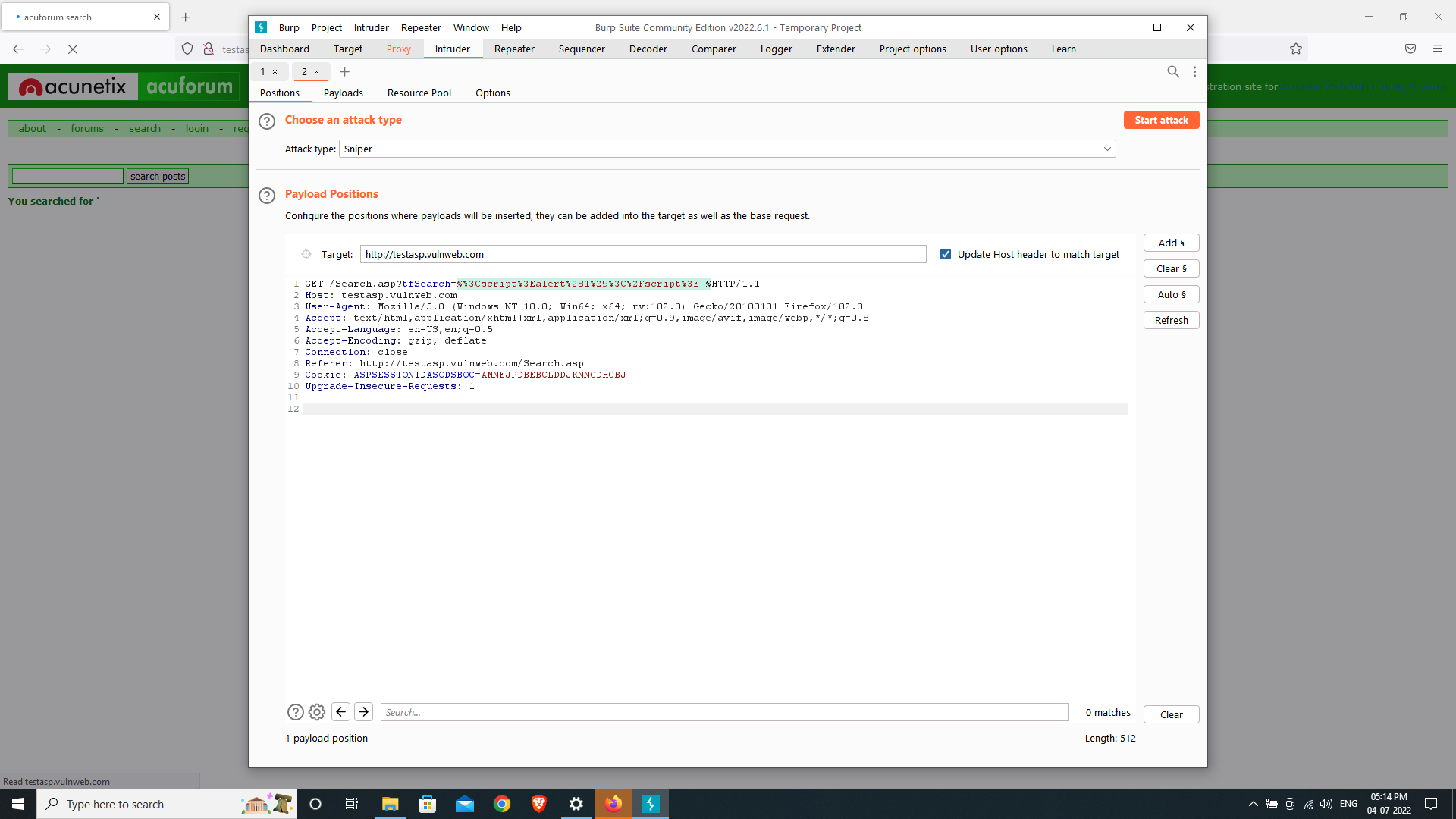
**Task 3 :-**

For the task 3 I’m gonna use BurpSuite for XSS payloads :-

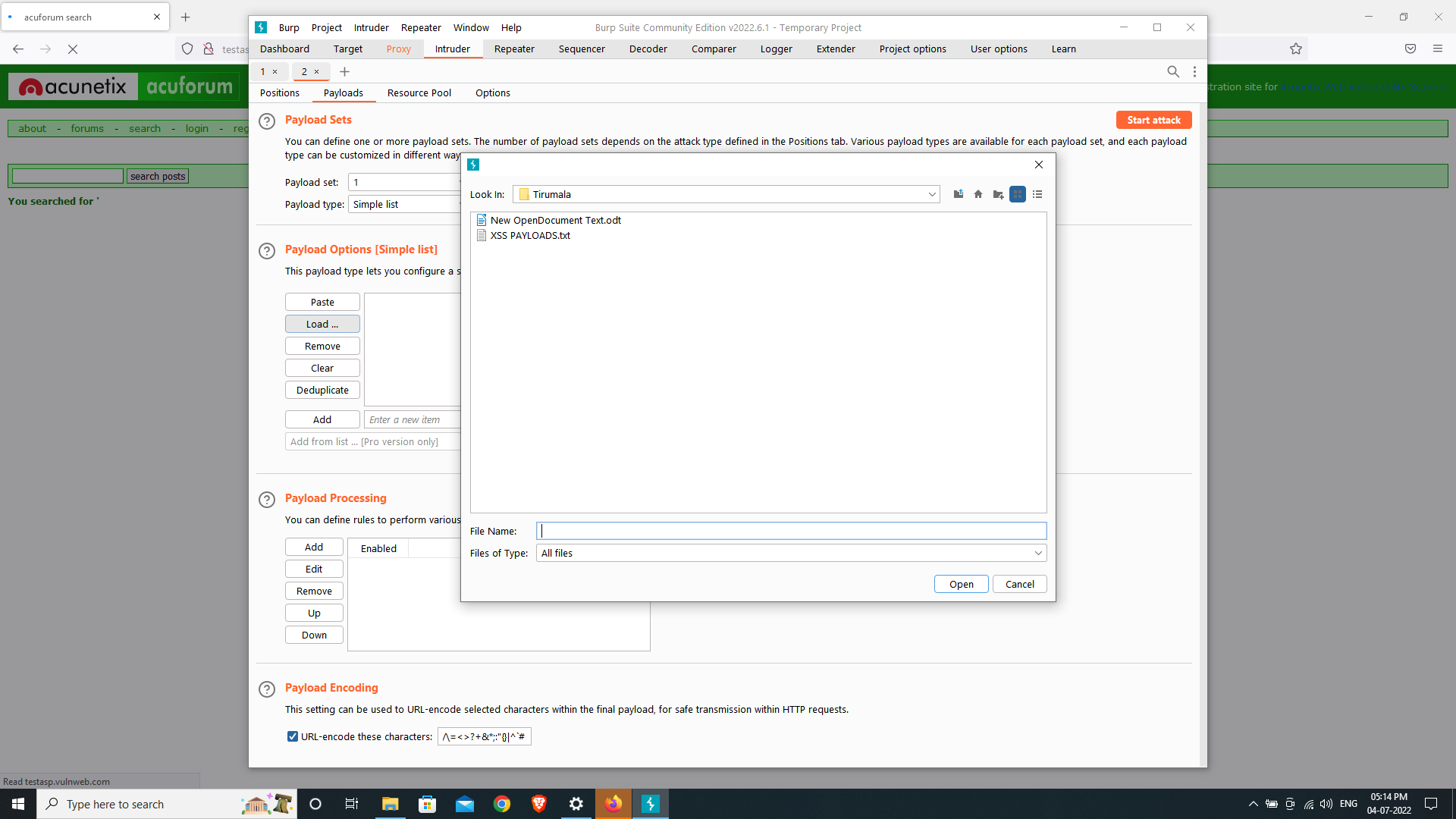


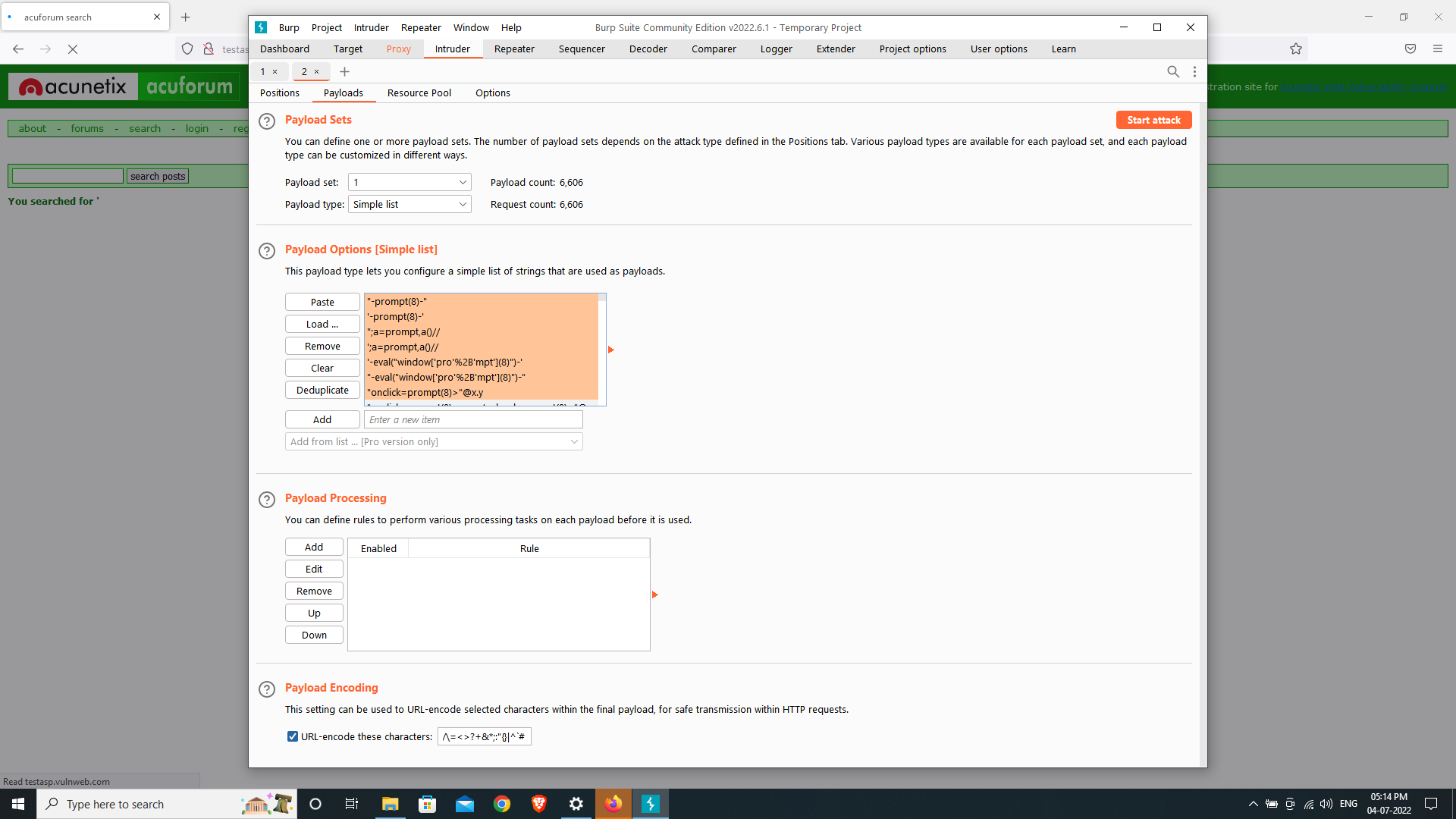


Using Burp suite I launched a sniper attack on the target address.

Before starting the attack I went to payloads and selected the XSS PAYLOAD.txt

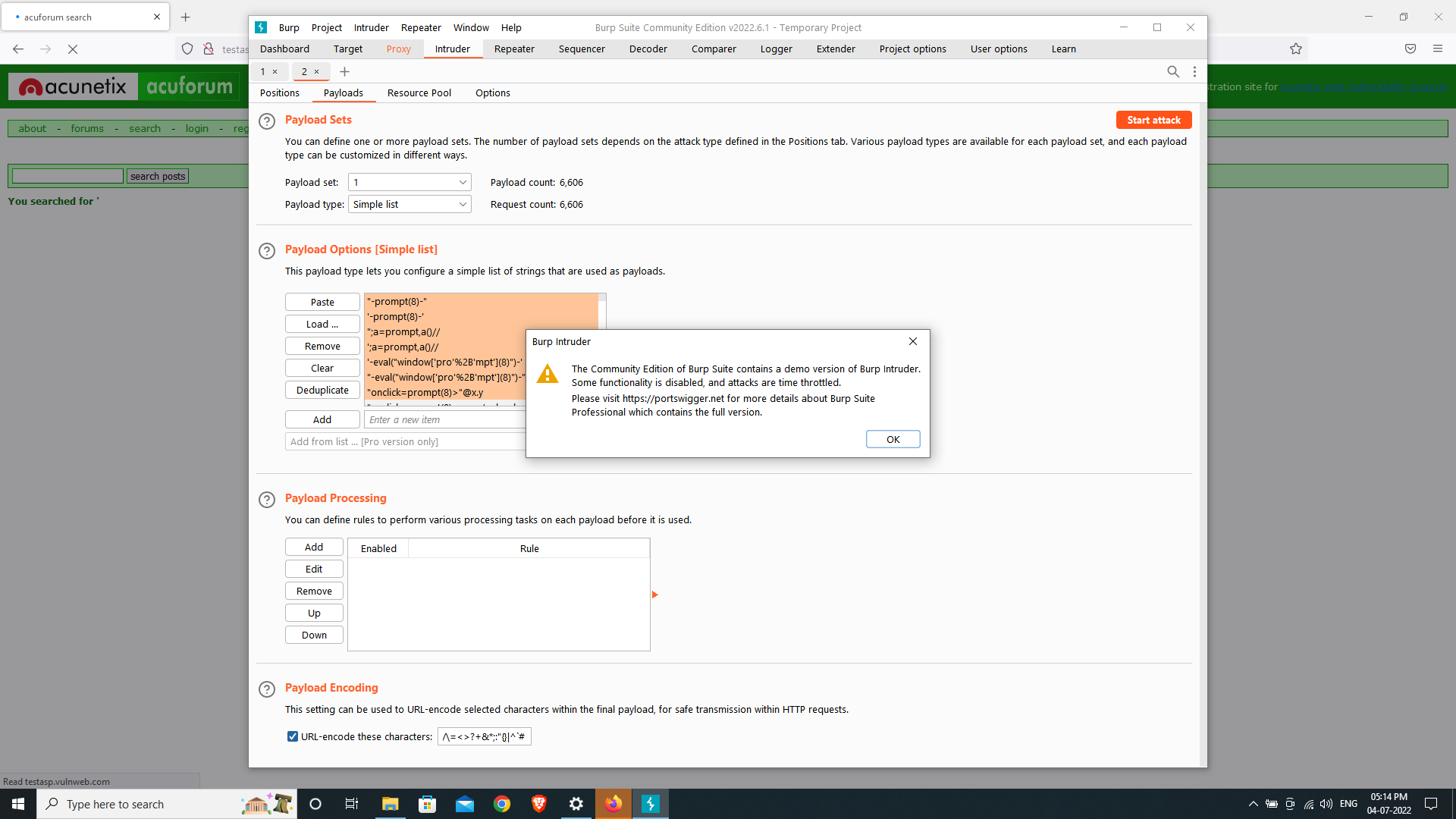
The text file is a list of strings that are to be used as the payload list.

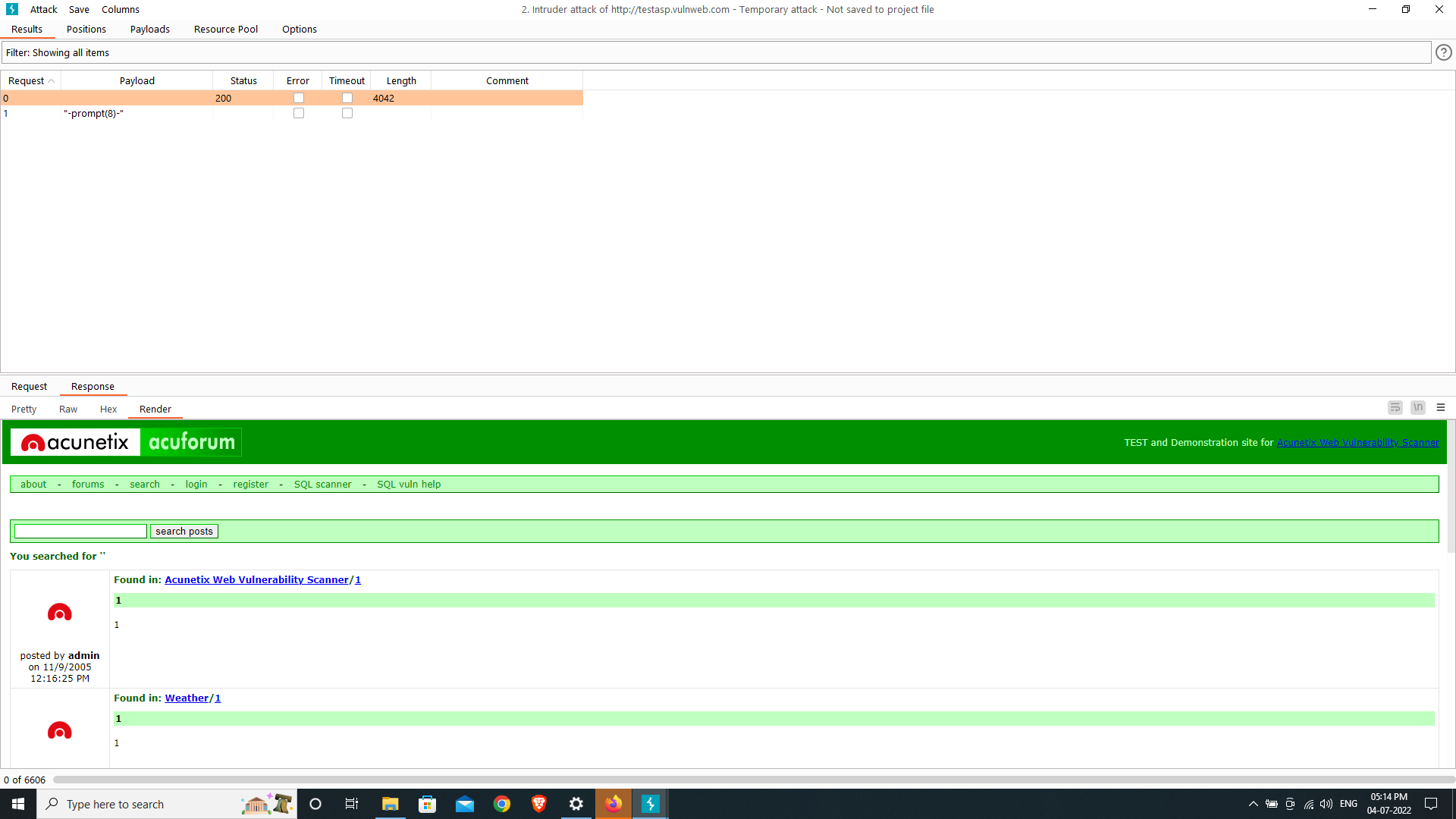




Paste the entire data of text file into the payload list.

Start the attack.





This is the result of the attack.

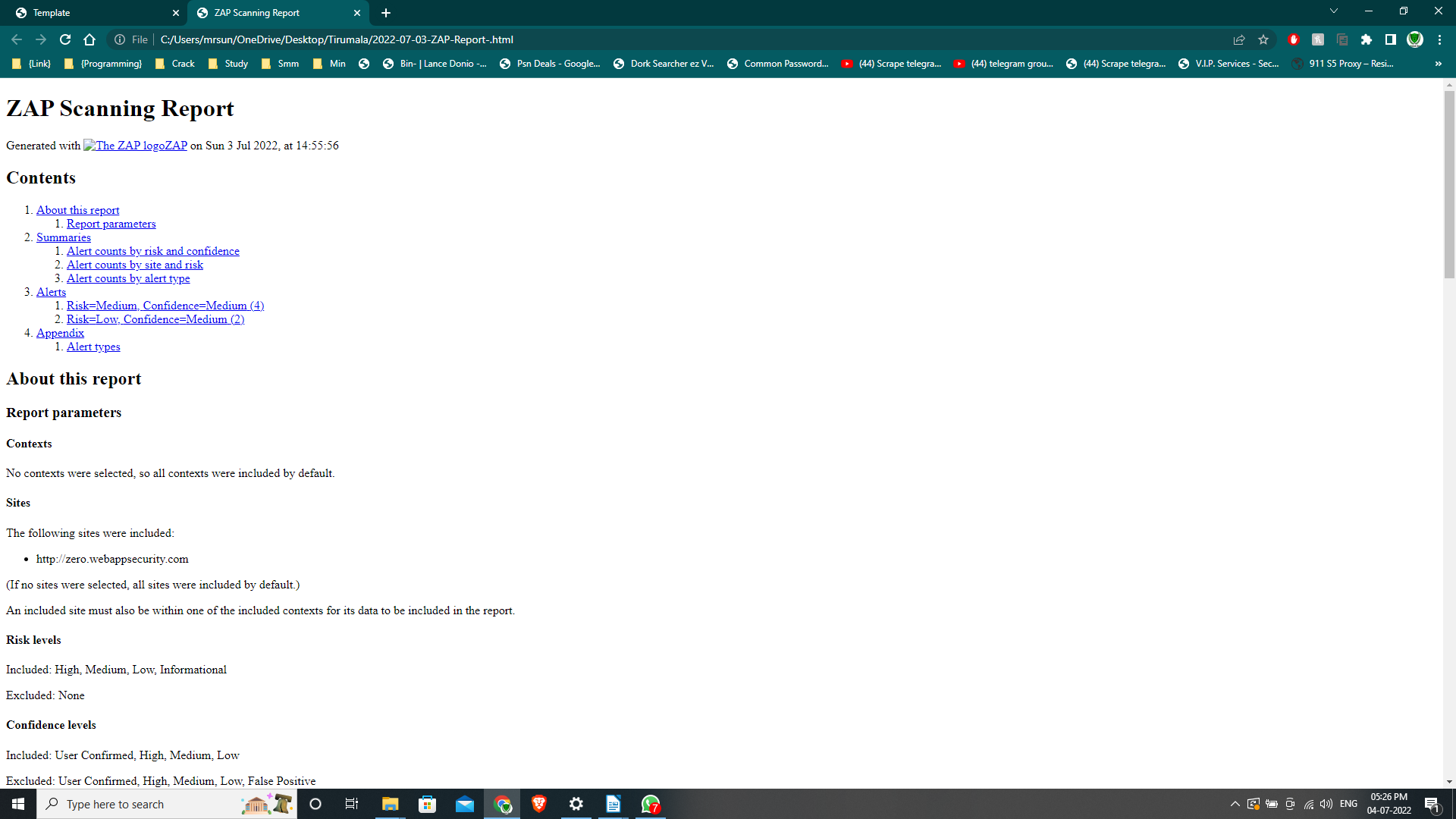
**Mitigation Steps :-**

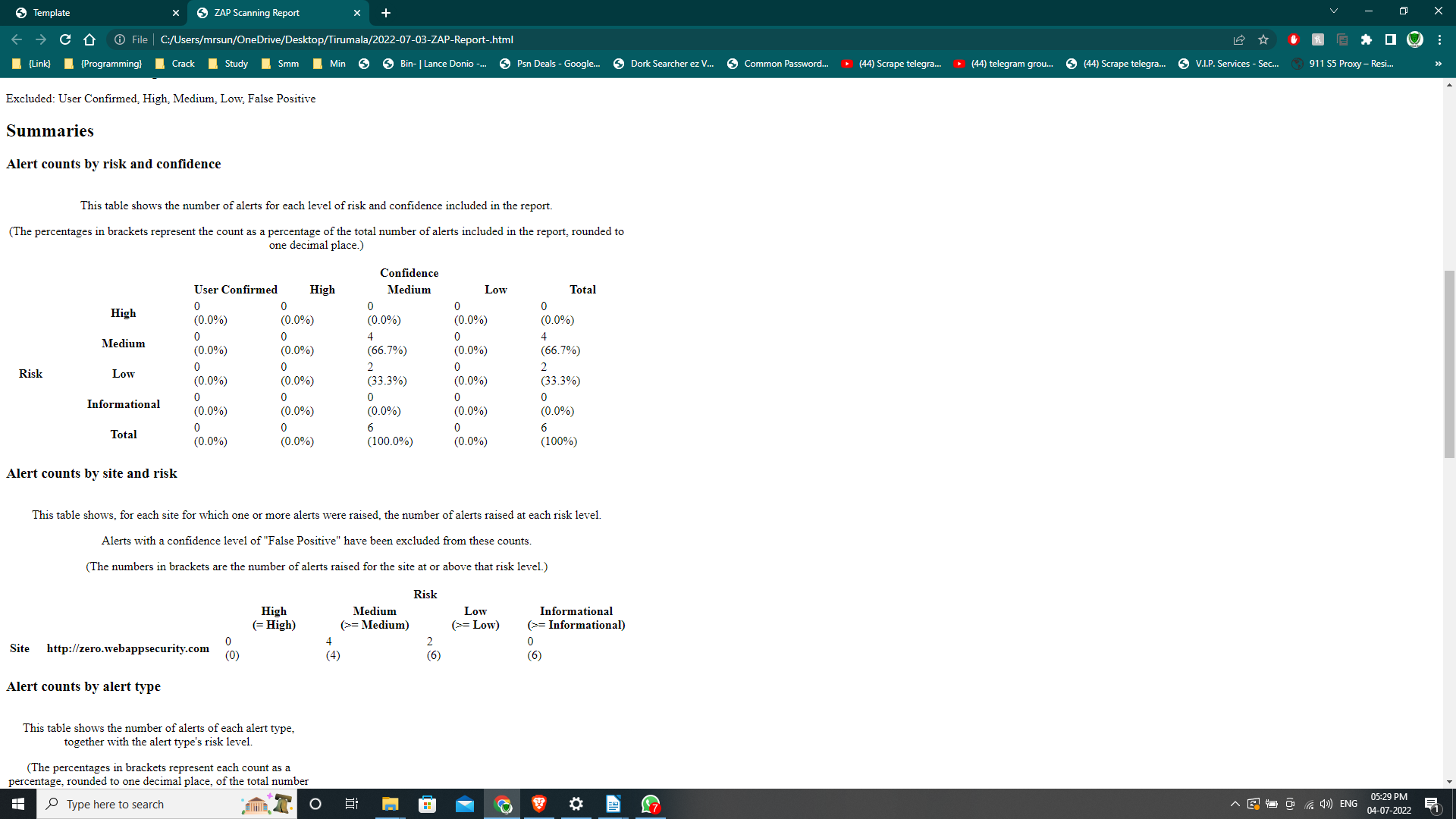
**1.** **Web Developer** **:-** when website would be developing that time only developers should be allowed only the alpha numeric key then only can payloads won’t work.there should not allowed to use any kind of special characters.

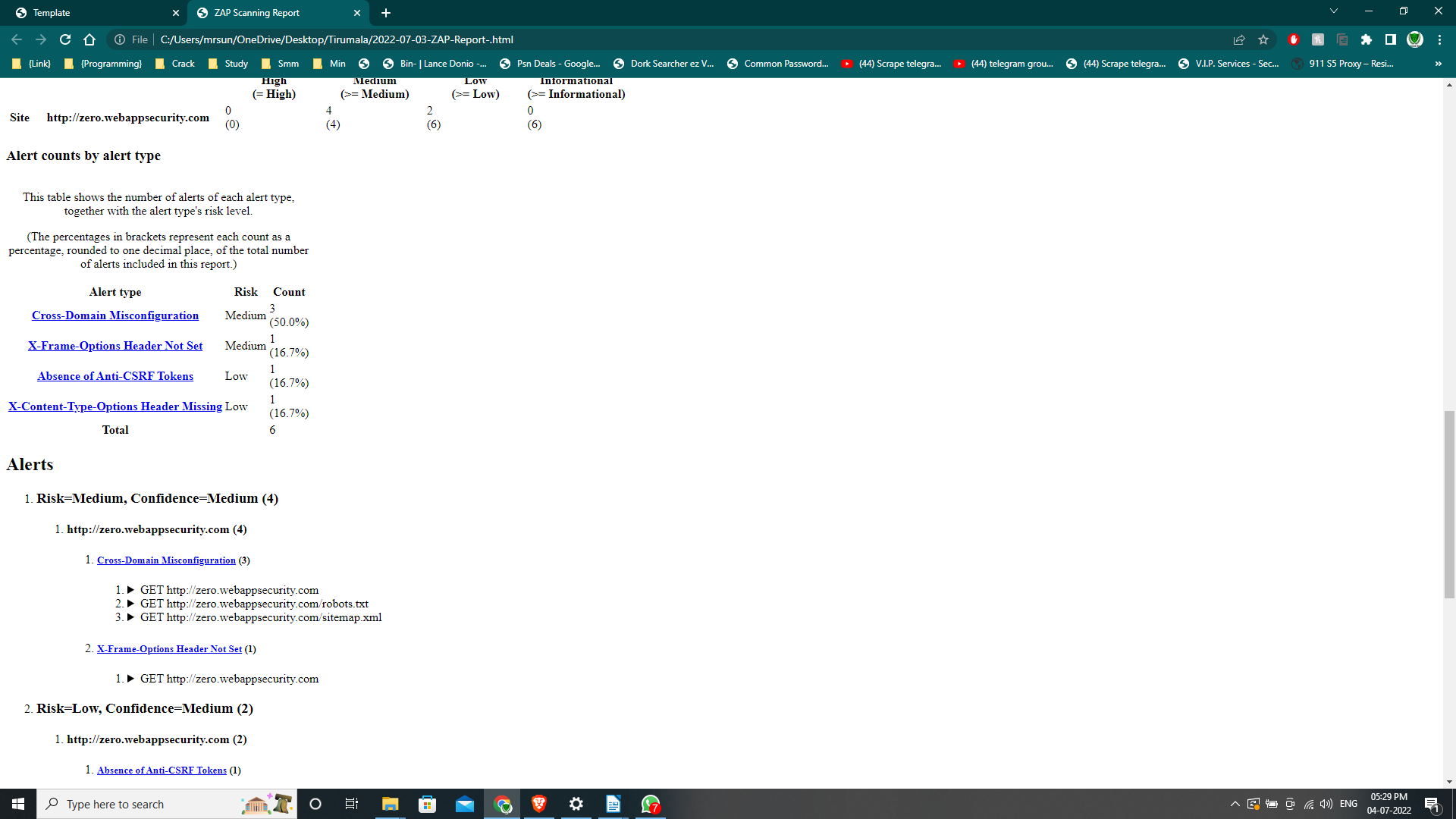
**2. Database Admin :-** Data should be encrypted in the database other wise payloads will work easily. If data will be encrypted then payloads won’t work.

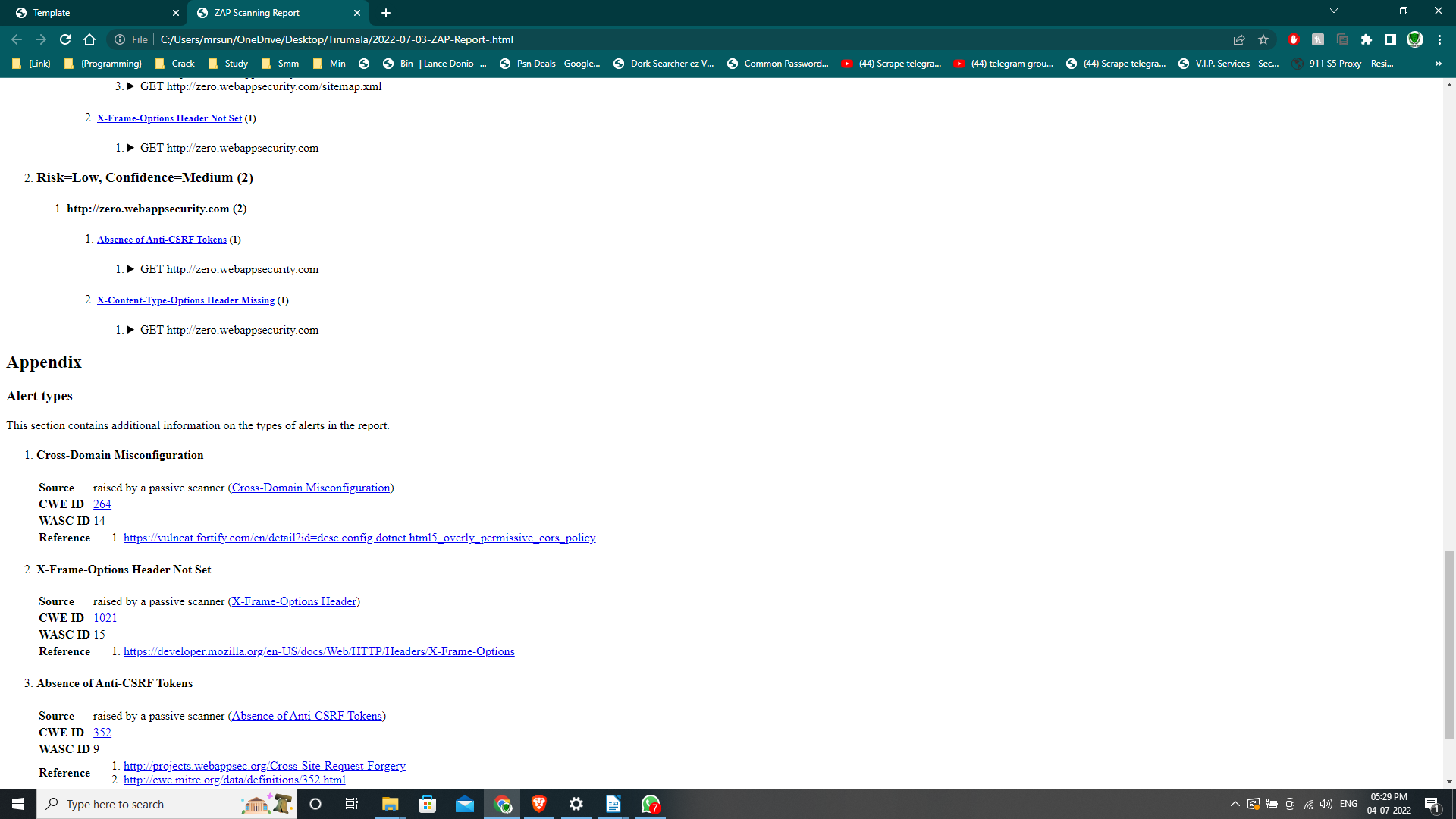
**Task 2 :-**

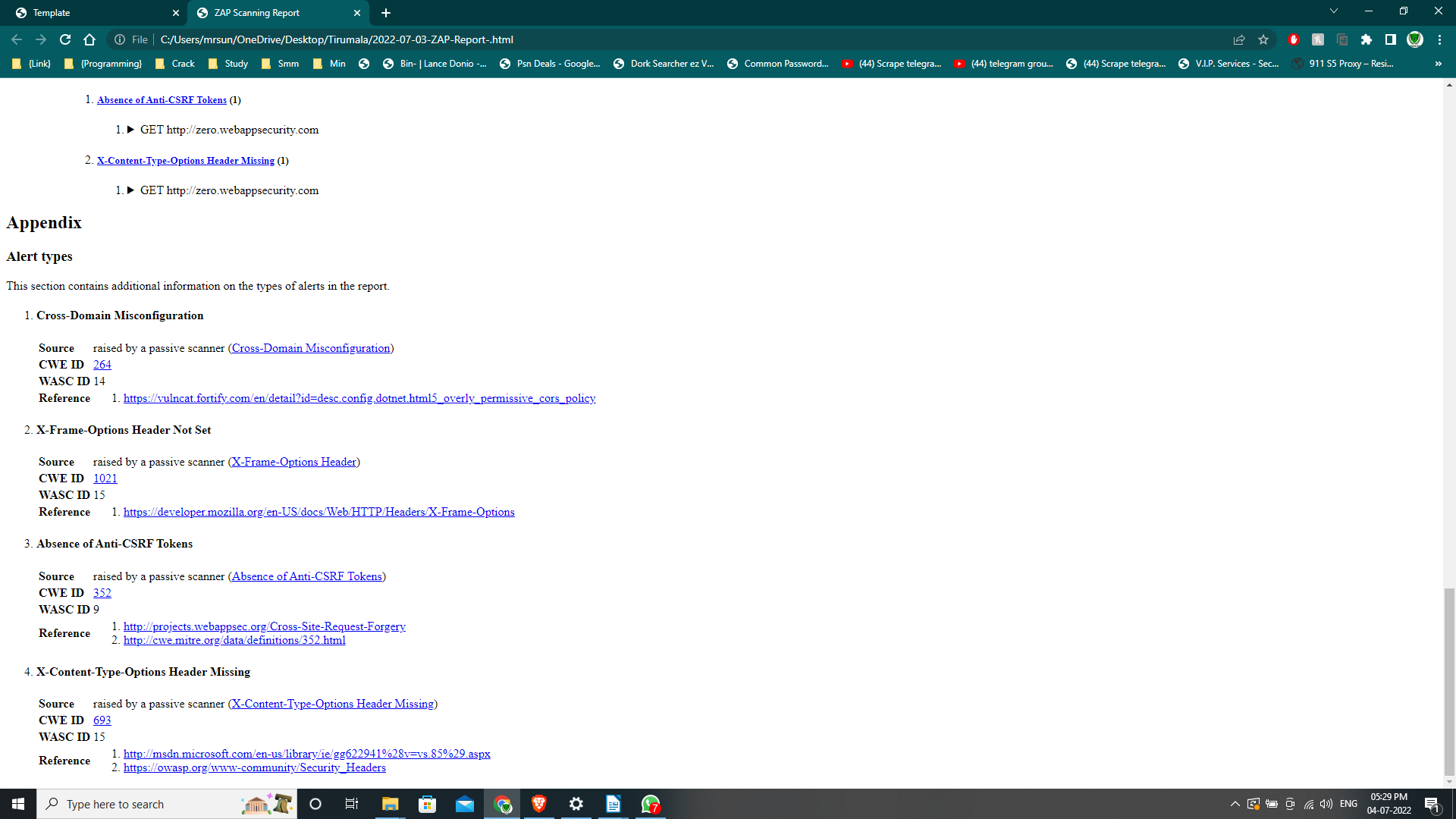
For finding the loopholes on a website that given already I’m using OWASP ZAP tool that is automated tool for finding the loopholes in a website.

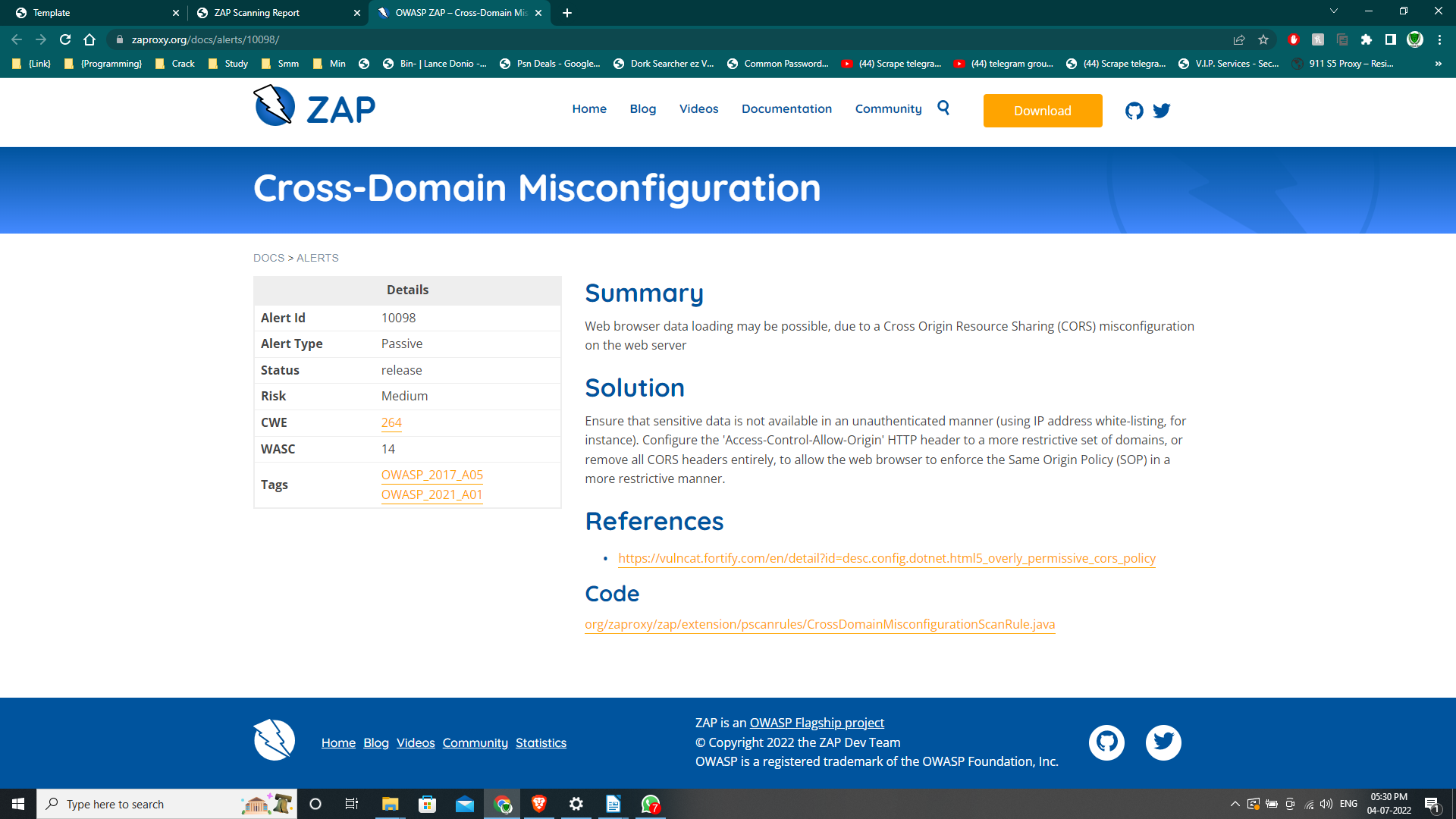






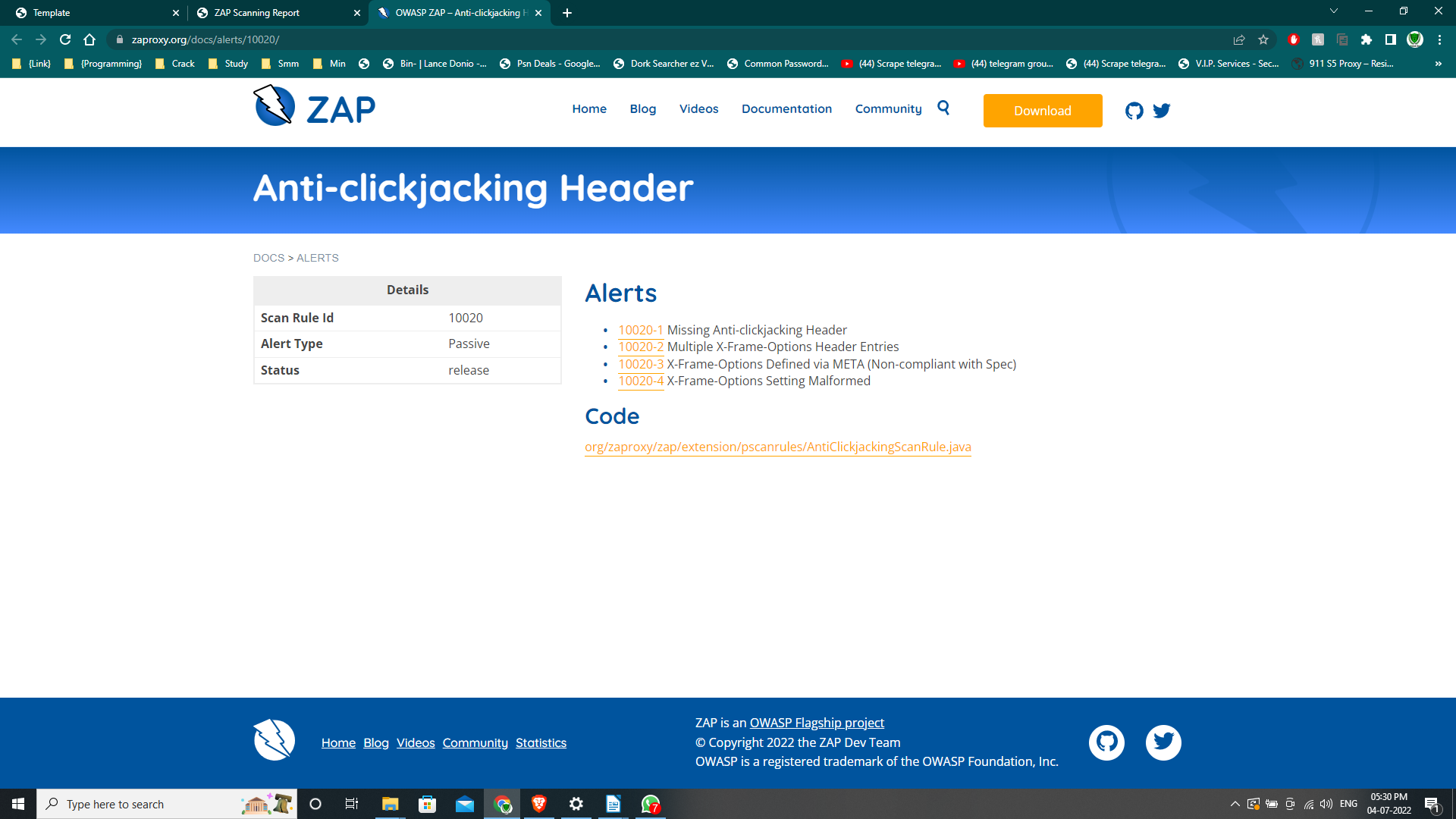


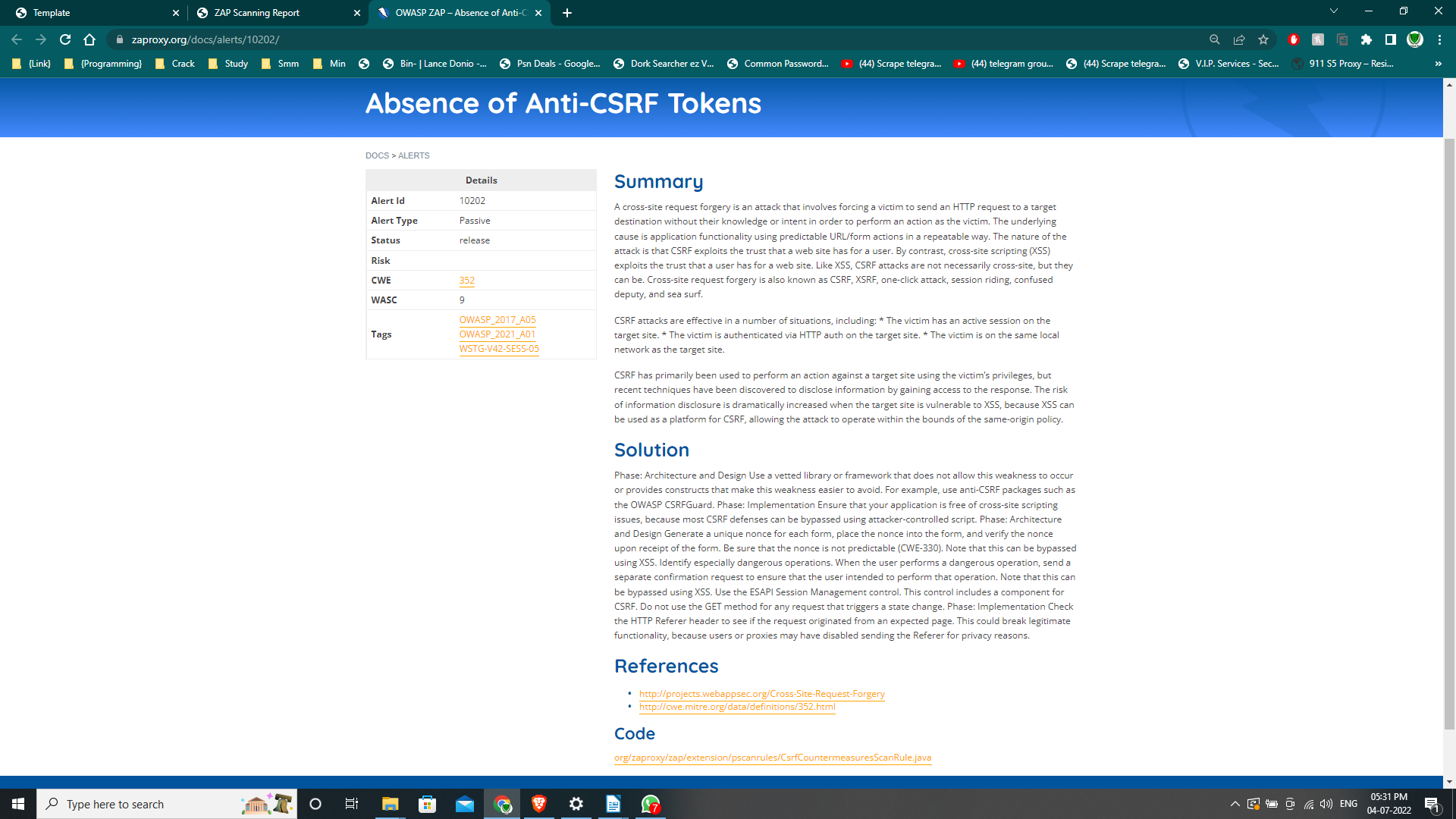




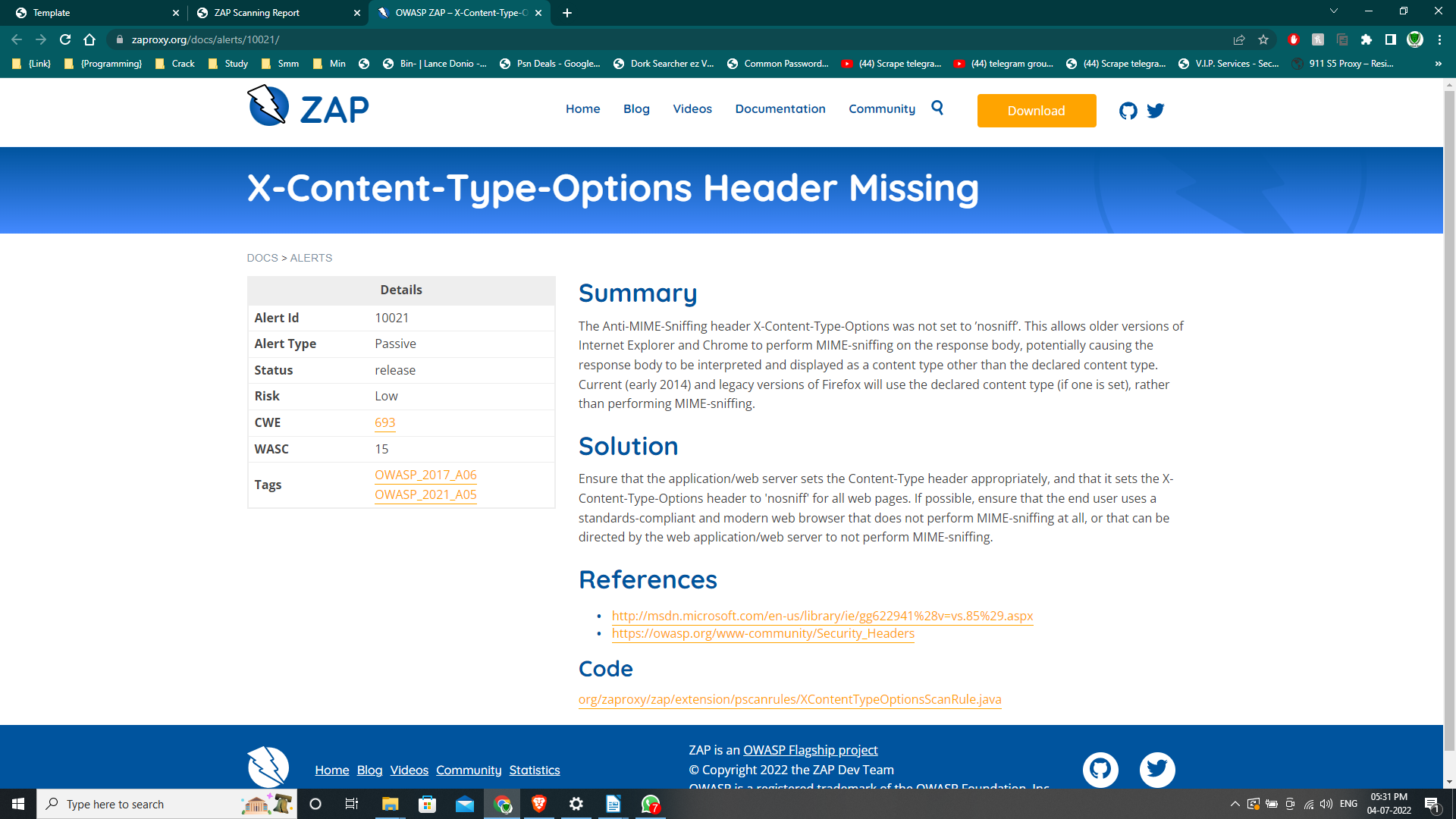
Data leak may be possible due to CORS(cross origin resource sharing) misconfiguration.

To avoid this make sure there is no sensitive data is available through unauthenticated manner. Remove all CORS headers entirely.





Cross site request forgery (CSRF) and cross site scripting(XSS) is possible.

To avoid this you can use anti-CSRF packages like the OWASP CSRF Guard. Implementation prevents from XSS attacks.

MIME-sniffing can be done on the response body using older versions of chrome and internet explorer.

To avoid this make sure that the X-Content-Type-Options header is set ‘nosniff’. If possible make sure end users use a updated version of the web browser that does not perform MIME\_sniffing.