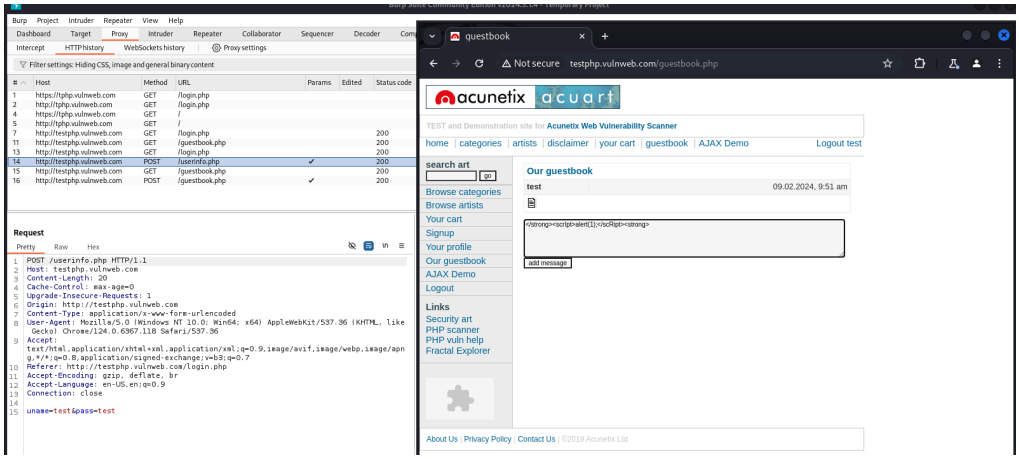
TASK - 2 (Web Application security)

OBJECTIVES:

- The objective is to scan the web application to find the vulnerability for exploitation.
- Used OWASP Zed Attack Proxy tool to find the unusual threats/alerts

STEPS TAKEN:

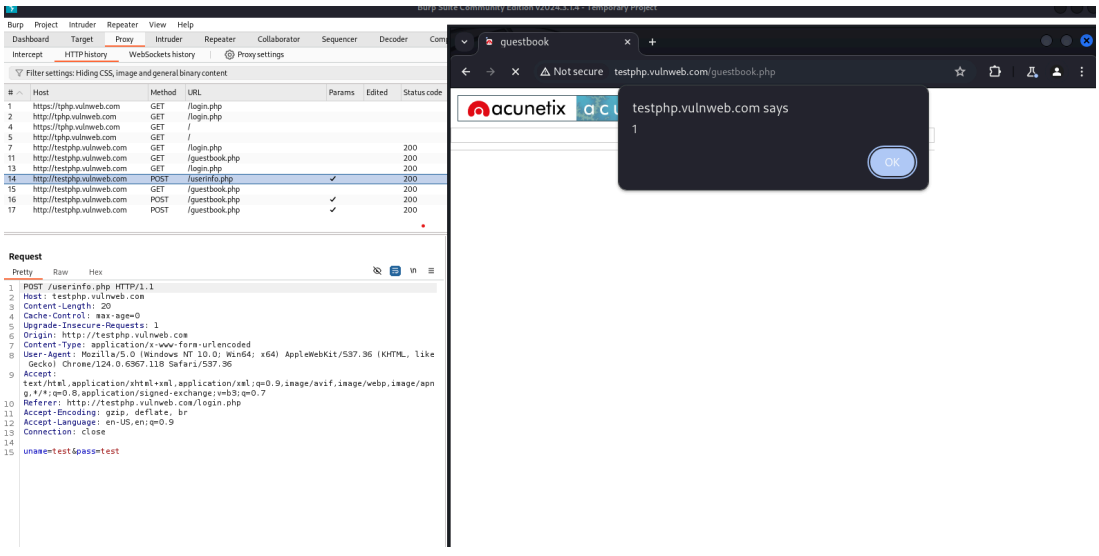
Pic-1&2 – Cross-site script attack



From the above Pic 1–

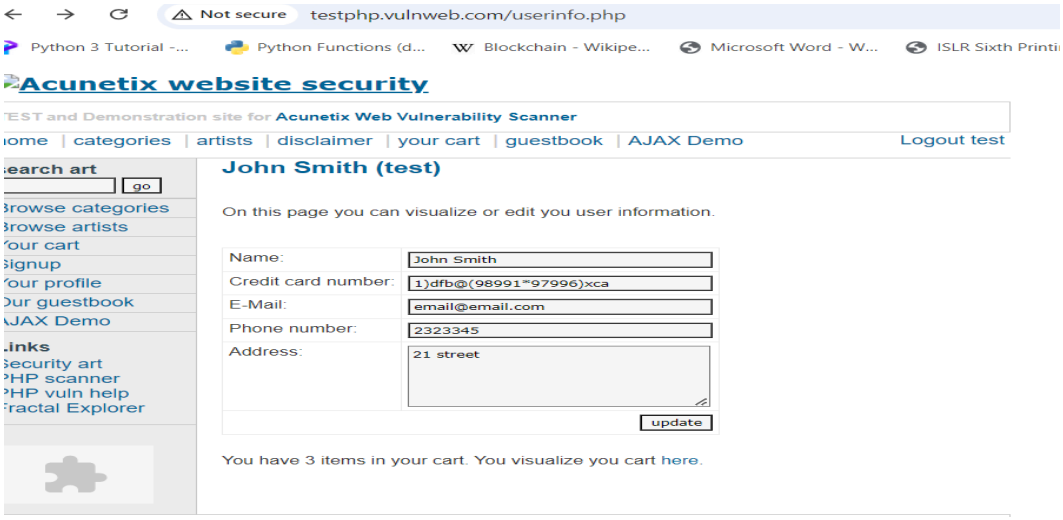
- After logging in to the website, using HTML tags for XSS(cross-site script)attack

Pic-2—



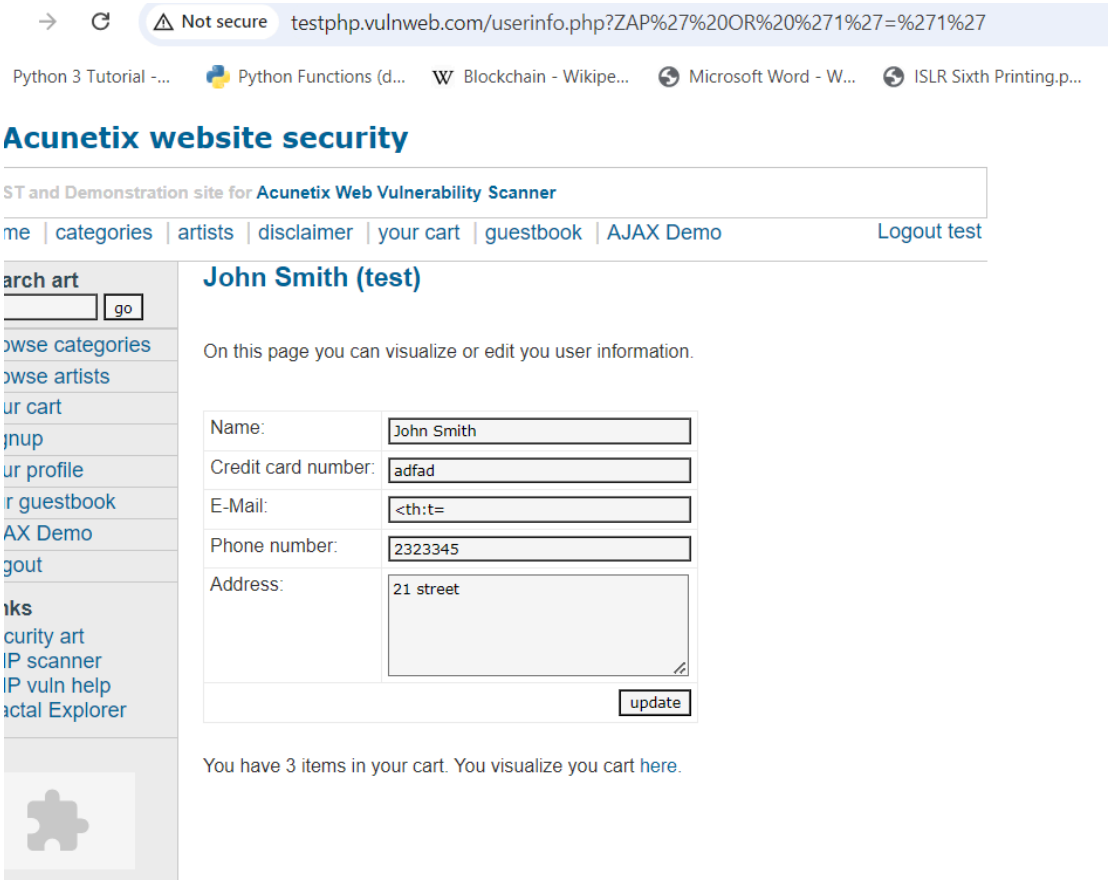
- From the Pic-2– we can notice the response after using that tag. It returns the value as 1. It means that while we modify values it would bypass the client-side checks so that attackers can easily attack that website.

Pic-3&4 – SQL Injection attack



- From the above pic-3– we can identify the standard user details after logging in(username and password as “test”)

Pic-4–



- From the above pic-4– I can see after passing the SQL query [ZAP’ OR ‘1’=’1’], it returned the same logging-in page with modified user details. So, it responds to the SQL

command instead of showing the error as “404 error”. So, an SQL injection attack might be possible to occur on this website.

Explanations:
$ In the above exploitation testing process, I have used the vulnerable website that can be seen.
$ Mostly after logging into the site, manually testing the exploits using OWASP ZAP scan report.
$ In that vulnerability scan, the site is only unusual to XSS and SQLi attacks.
$ I have produced screenshots with basic explanations for understanding purposes.

Solutions for mitigating these attacks:--

1. To help mitigate XSS attacks against the user's session cookie, set the session cookie to HTTP-only. In browsers that support the HTTP-only feature (such as more recent versions of Internet Explorer and Firefox), this attribute can prevent the user's session cookie from being accessible to malicious client-side scripts that use document cookies.
2. When performing input validation, consider all potentially relevant properties, including length, type of input, the full range of acceptable values, missing or extra inputs, syntax, consistency across related fields, and conformance to business rules.
3. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation.
4. Escape all data received from the client.
5. Apply an 'allow list' of allowed characters, or a 'deny list' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible.
6. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection but minimizes its impact. Grant the minimum database access that is necessary for the application

CHALLENGES FACED:

- No challenges faced