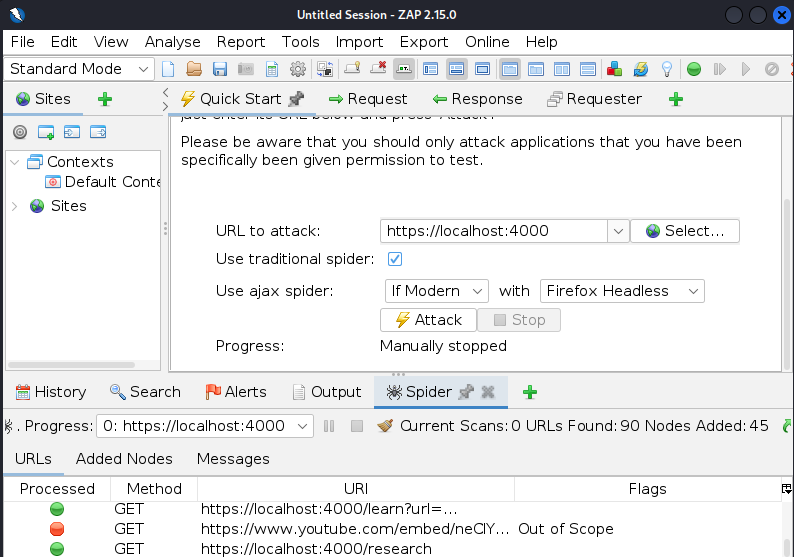
**DEVELOPERSHUB INTERNSHIP**

**Week 6: Advanced Security Audits & Final Deployment**

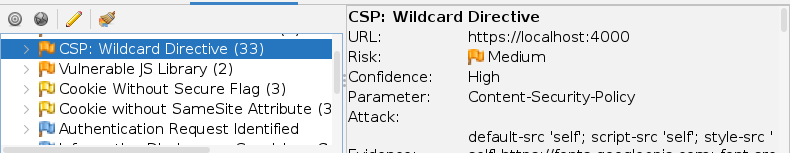
**Security**

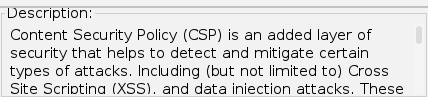
***Security Audits & Compliance***

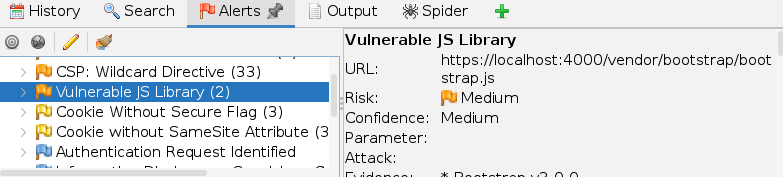
***OWASP ZAP:***



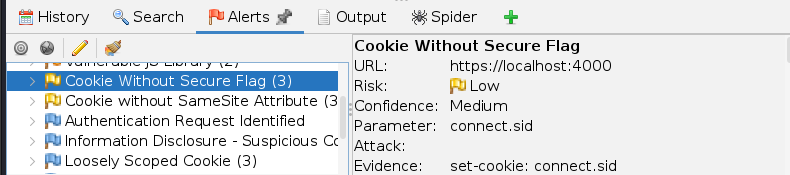
**ALERTS WITH DESCRIPTION:**

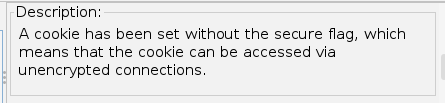


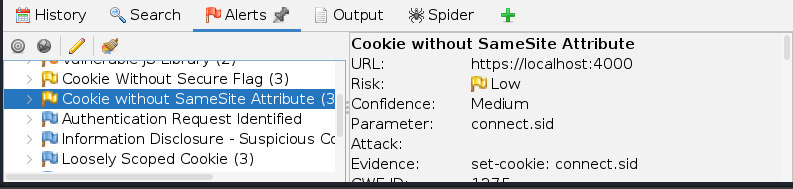


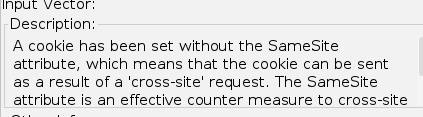


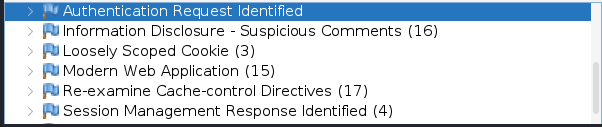






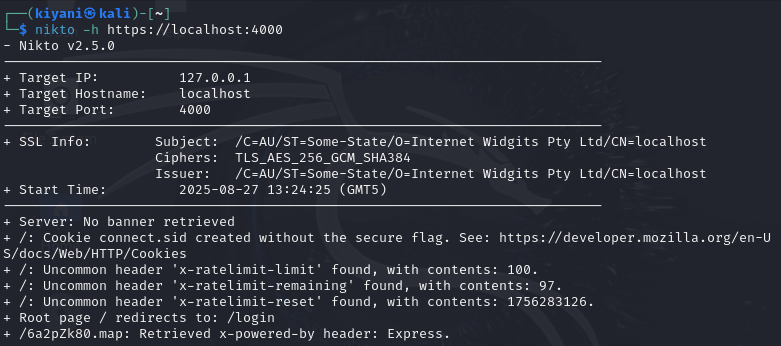


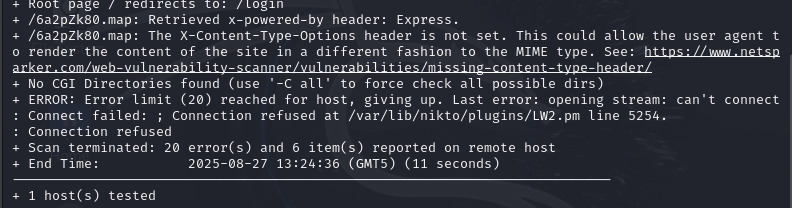




***NIKTO:***

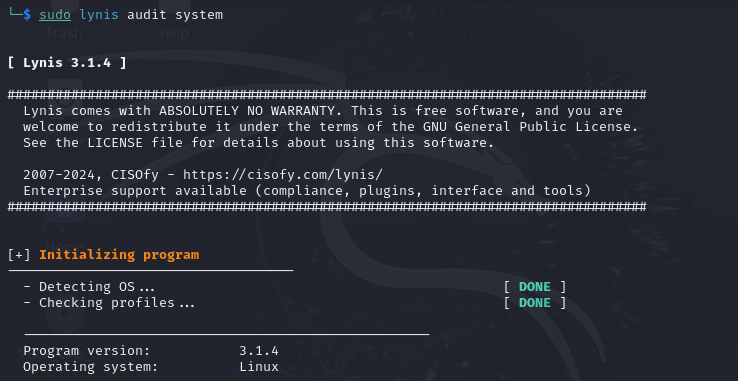
***Scans the web application for any misconfiguration, vulnerabilities.***

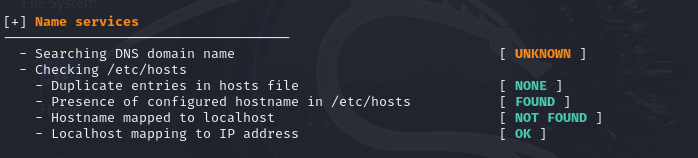


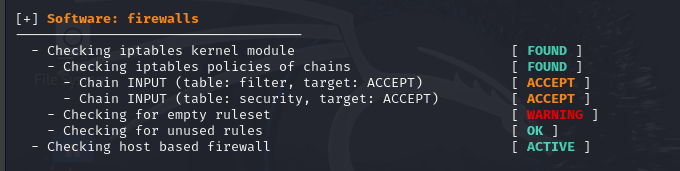


***LYNIS:***

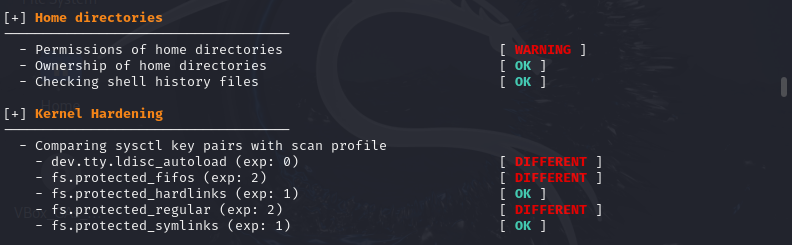
***Audits the system’s configuration and gives recommendations to improve security.***



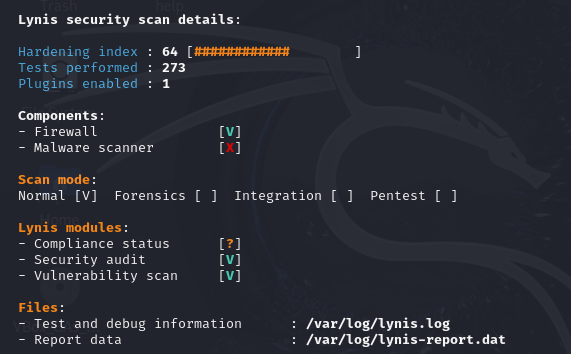




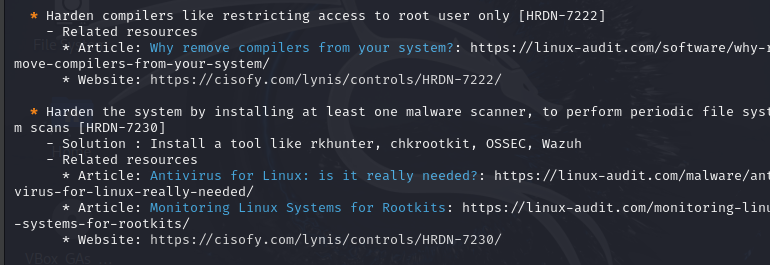


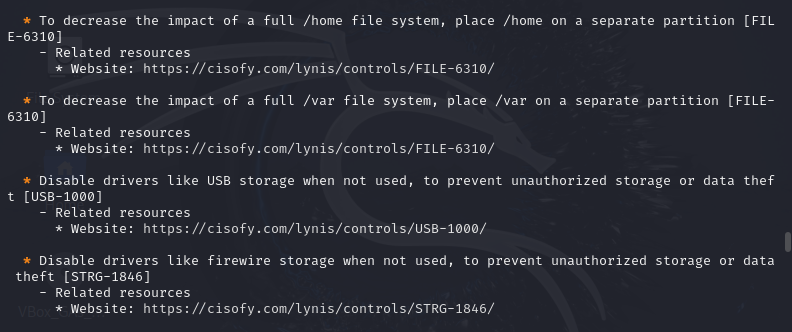






**Few of the recommendations to improve security:**





**COMPLIANCE WITH OWASP Top 10:**

**1. Vulnerable JavaScript library (ZAP)**

* **OWASP Mapping:** *A06: Vulnerable and Outdated Components*
* Explanation: Using old JS libraries means attackers can exploit **known CVEs**. This is one of the most common ways websites get hacked.

**2. Cookie without Secure Flag (ZAP & Nikto)**

* **OWASP Mapping:** *A05: Security Misconfiguration* OR *A07: Identification & Authentication Failures* (depending on context).
* Explanation: Without the Secure flag, cookies can be sent over HTTP > **session hijacking risk**.

**3. Cookie without SameSite attribute (ZAP)**

* **OWASP Mapping:** *A05: Security Misconfiguration*
* Explanation: No SameSite > vulnerable to **CSRF attacks** because the browser sends cookies with cross-site requests.

**4. Authentication request identified (ZAP)**

* **OWASP Mapping:** *A07: Identification & Authentication Failures*
* Explanation: ZAP flags endpoints handling login/auth. They need **extra hardening** (rate limiting, MFA, lockout, CSRF protection).

**5. Information Disclosure – suspicious comments (ZAP)**

* **OWASP Mapping:** *A05: Security Misconfiguration*
* Explanation: Devs sometimes leave hints (//TODO, API keys, debug info). Attackers use this to craft attacks.

**6. Loosely scoped cookie (ZAP)**

* **OWASP Mapping:** *A05: Security Misconfiguration*
* Explanation: Cookie valid for a **broad domain/path** > increases risk of XSS/session fixation.

**7. Re-examine cache-control directives (ZAP)**

* **OWASP Mapping:** *A05: Security Misconfiguration*
* Explanation: If cache is misconfigured, sensitive pages (like /profile or /invoice) might get cached > attackers could retrieve them.

**8. X-Content-Type-Options header missing (Nikto)**

* **OWASP Mapping:** *A05: Security Misconfiguration*
* Explanation: Without X-Content-Type-Options: nosniff, browsers may try to “guess” file types > **MIME sniffing attacks**.

**9. Uncommon headers (Nikto)**

* **OWASP Mapping:** *A05: Security Misconfiguration*
* Explanation: Rare headers could indicate misconfigured middleware or leaking system info.

**10. Root page redirects to login (Nikto)**

* **OWASP Mapping:** Not directly in Top 10 (more an *info finding*). But still related to *A05: Security Misconfiguration* if redirect logic leaks details.

**11. MongoDB authorization disabled (Lynis)**

* **OWASP Mapping:** *A01: Broken Access Control* AND *A05: Security Misconfiguration*
* Explanation: If MongoDB has **no auth**, anyone can connect > total data theft. This is critical.

**12. Firewall empty ruleset (Lynis)**

* **OWASP Mapping:** *A05: Security Misconfiguration*
* Explanation: Firewall not configured properly > network exposed.

**13. Permissions of home directories warning (Lynis)**

* **OWASP Mapping:** *A05: Security Misconfiguration*
* Explanation: Loose file permissions > local privilege escalation risk.

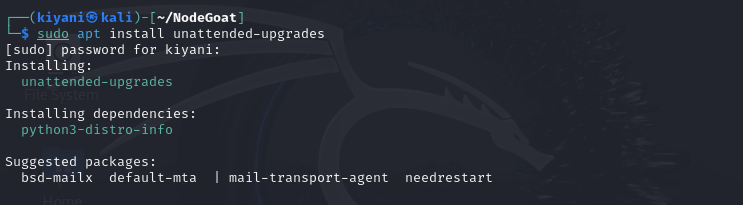
**14. Installed malware scanner not found (Lynis)**

* **OWASP Mapping:** *A09: Security Logging & Monitoring Failures* (indirect)
* Explanation: Lack of antivirus/IDS means malicious activity might go undetected.

***Secure Deployment Practices***

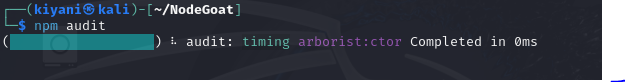
**Enabling automatic security updates and automatic scanning is extremely crucial. As the attackers first target to get into your system is to exploit the vulnerabilities, hence by this the surface area of the risk shrinks.**

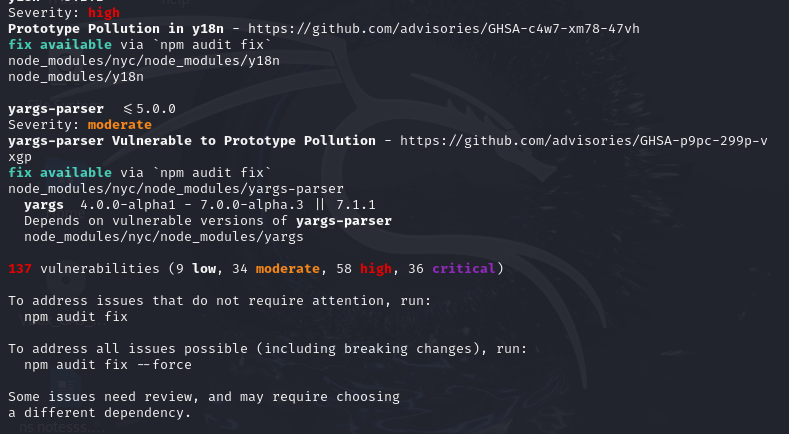
**So for the system on which we will deploy our web application we will integrate automatic security updates, this configures the server to download and install critical security patches automatically.:**

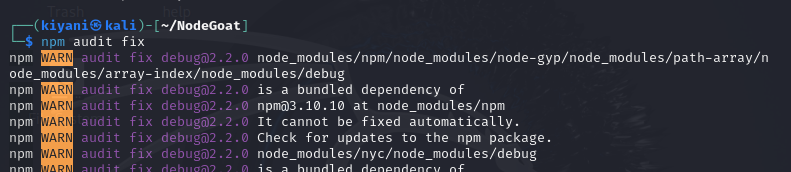




**And for the application we can use tools to scan the vulnerable packages, and timely fix or replace them:**



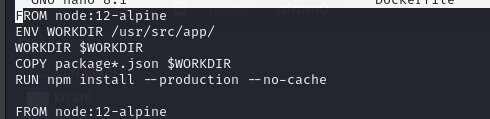




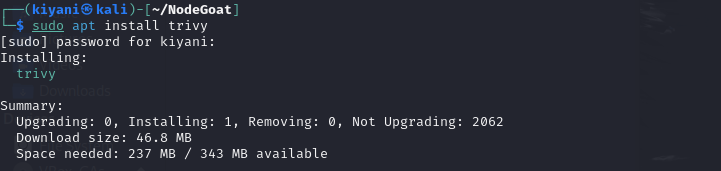
**A Docker is a tool that lets developers package the application in a container. And the container image security is extremely important. The best practices consist of:**

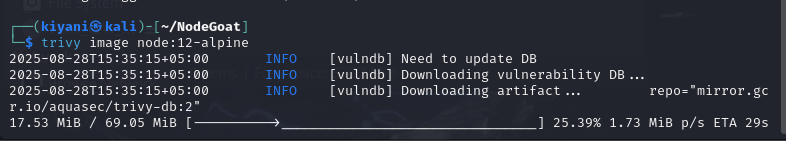
* **Use trusted base images (official ones, not random).**

**It has the “Official Image” badge on Docker Hub. Although it is a bit outdated, preferable are 18 and 20.**



* **Scan container images for vulnerabilities using tools like Trivy or Docker Scan.**



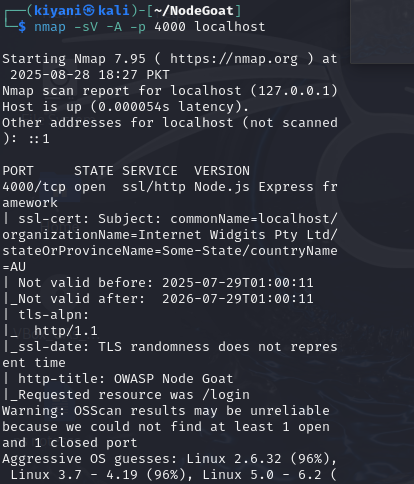


**Vulnerabilities in node:12-alpine:**

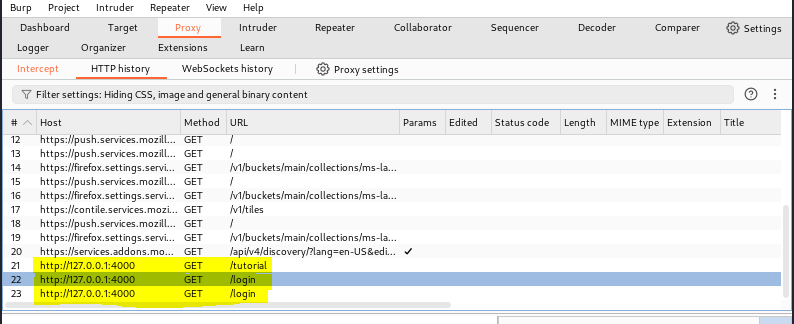
* **Outdated OpenSSL library with known CVEs allowing potential denial-of-service or data leakage.**
* **Weaknesses in the zlib compression library leading to possible denial-of-service attacks.**
* **Issues in Node.js 12 runtime itself, since it is End-of-Life and no longer receives security patches.**
* **Potential privilege escalation vulnerabilities due to older Linux kernel-related packages bundled in the image.**
* **Don’t run containers as root (use limited user permissions).**
* **Keep images updated with the latest patches.**
* **Avoid storing passwords or API keys inside container files (use secrets instead).**

***Final Penetration Testing***

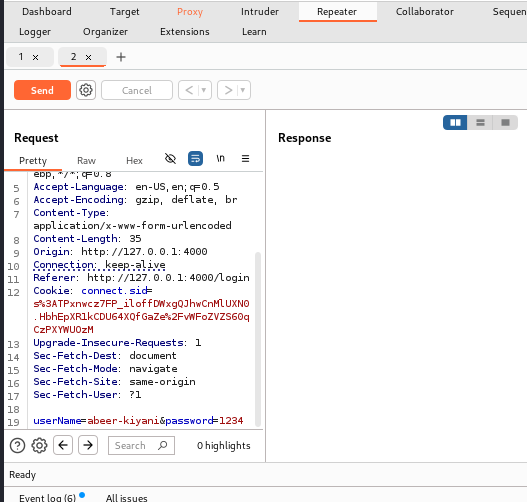
**Looking for the services available and running on the NodeGoat.**

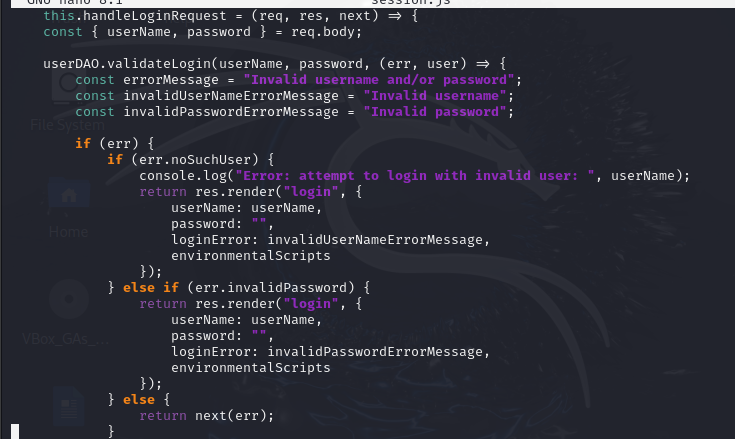


**Once i start the Burpsuite, as I open my application, I can see burp intercepting the requests and showing the details.**

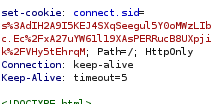


**I first send the request to the repeater as I login into the application. As you can see, the username and password were here shown and I wanted to test the website for SQL injection. By inserting Injection scripts or even bypassing the login by default credentials. Although my default credentials showed errors when I tried to test but surprisingly the SQL script did let the response be normal. For which I had to implement and check my input validation and sanitization.**



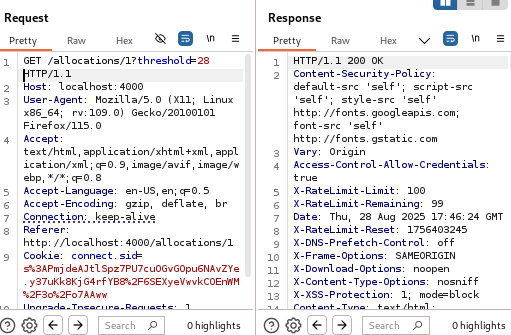


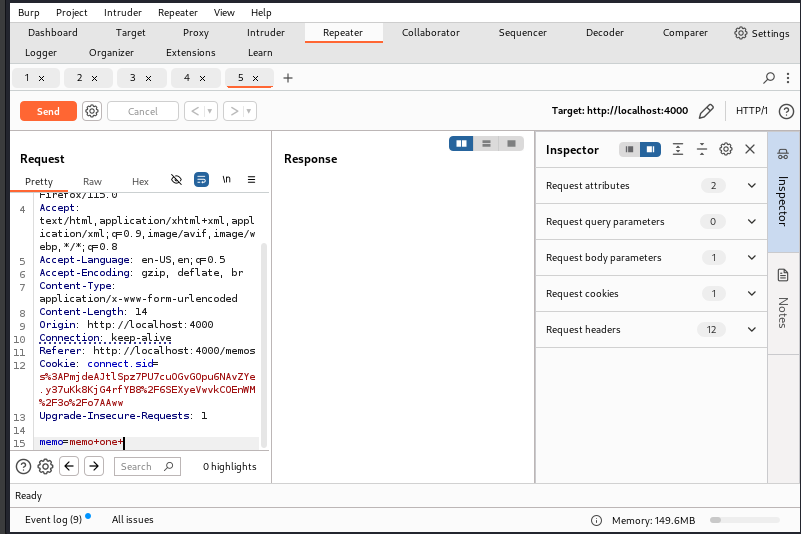
**The application uses a cookie named connect.sid for session management. However, it lacks important security attributes (HttpOnly, Secure, and SameSite). This happened because I ran my application as HTTP not HTTPS. Hence later switched to HTTPS and which resolved the issue.**



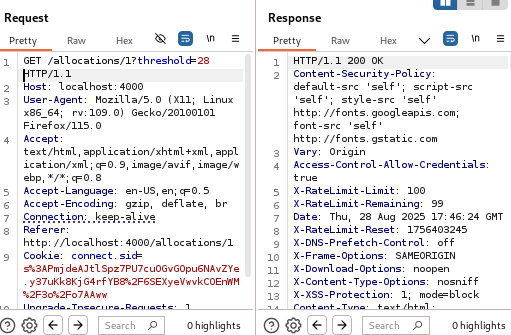
**The memo page and other pages such as collaboration, allocation, they did not have any sort of issue.**

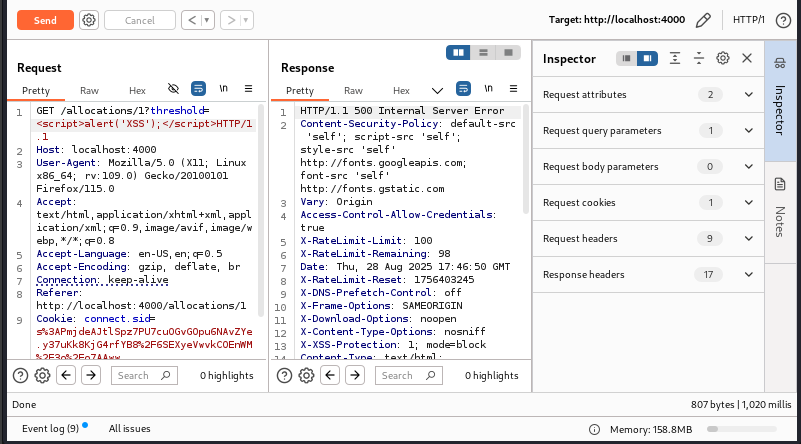




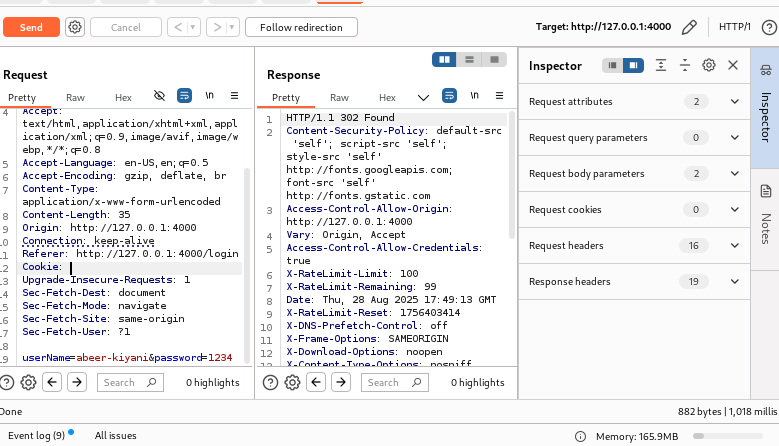


**Then I ran the XSS test on my allocations page form and it passed the test as gave the error. So the application was safe from this as well.**

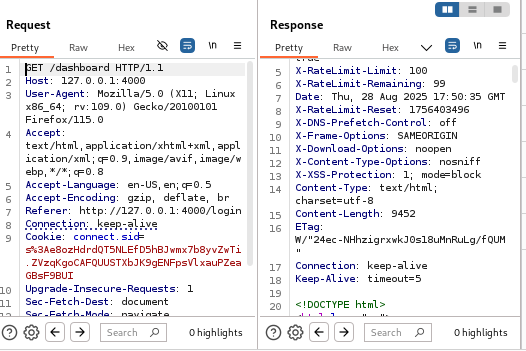




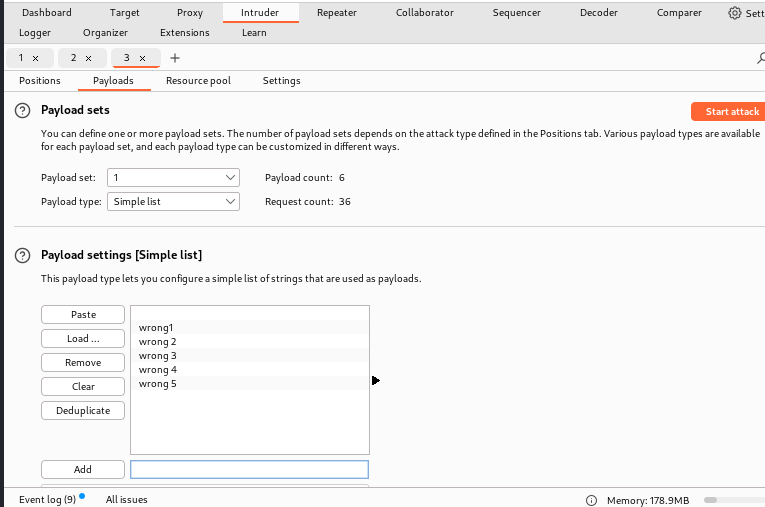
**Now ensuring that my application has secure session management hence when I remove the cookie on which session relies on, it showed me an error. Although again my testing was done on the HTTP version of my application since HTTPS wasn’t cooperating with burpsuite.**

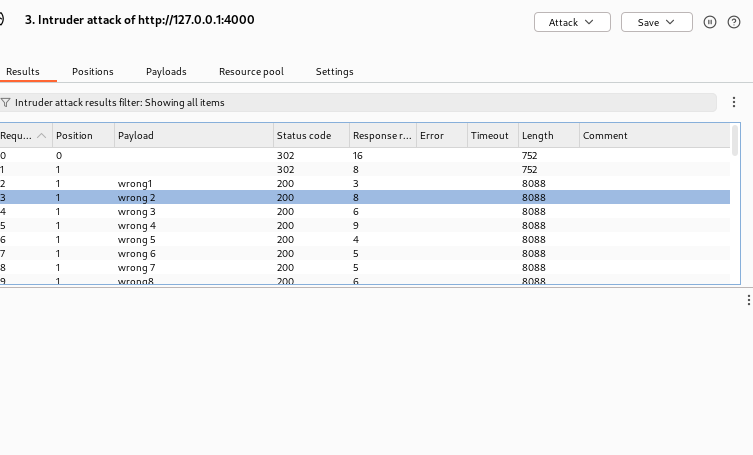


**During testing, it was verified that the application has all the necessary HTTP security headers implemented. The presence of these headers indicates that the application follows secure development practices to protect against common web vulnerabilities like clickjacking, MIME sniffing, and data exposure.**



**I also sent it to intruder to check the rate-limiting of my website and it is implemented as you can see but in one hour 100 requests are acceptable only.**





**During the assessment of the application, it was observed that several security mechanisms are already in place, such as the presence of common security headers and appropriate session management practices in some areas. These implementations demonstrate the developers consideration of baseline security measures. However, certain issues were also identified. Overall, while the application shows efforts toward secure design, it requires further improvements to mitigate critical risks and ensure robust protection of user data and system integrity.**