**ASVS Analysis**

1. **– General Data Protection**

8.1.1 (not valid): Verify the application protects sensitive data from being cached in server components such as load balancers and application caches.  
Ensured that your server is sending appropriate cache control headers to instruct intermediaries) not to store sensitive data.  
Added the header functions to set cache control headers, ensuring that sensitive data is not cached by browsers or other intermediaries. The Cache-Control, Pragma, and Expires headers are set to instruct the browser not to store cached copies of the page.  
  
A black screen with green and white text

Description automatically generated

8.1.1 (not valid): Verify that all cached or temporary copies of sensitive data stored on the server are protected from unauthorized access or purged/invalidated after the authorized user accesses the sensitive data.  
I added unset($last\_payment); after processing the sensitive data in both the success and catch blocks. This ensures that the variable is cleared, reducing the chances of sensitive information being retained in memory.

8.1.2 (valid): Verify the application minimizes the number of parameters in a request, such as hidden fields, Ajax variables, cookies and header values.

8.1.3 (not valid): Verify the application can detect and alert on abnormal numbers of requests, such as by IP, user, total per hour or day, or whatever makes sense for the application.  
Added a modification that ensures that the request rate limiting and anomaly detection logic is executed at the beginning of your script, and if the threshold is exceeded, it redirects the user to an error page and terminates the script execution.

A screen shot of a computer program

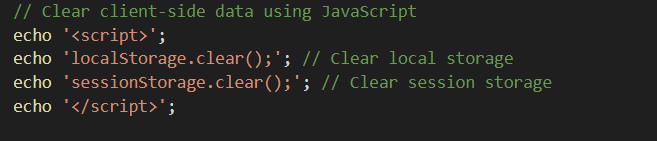
Description automatically generated

1. **– Client-side Data Protection**

8.2.1(Valid) Verify the application sets sufficient anti-caching headers so that sensitive data is not cached in modern browsers.

8.2.1(Non-Valid) Verify that authenticated data is cleared from client storage, such as the browser DOM, after the client or session is terminated.

Added JavaScript to clear client-side data



1. **– Sensitive Data Protection**

8.3.1(Valid) Verify that sensitive data is sent to the server in the HTTP message body or headers, and that query string parameters from any HTTP verb do not contain sensitive data.

8.3.2(Not Valid) Verify that users have a method to remove or export their data on demand.

Created a method for users to delete their data with the delete\_data button.

8.3.3(Not\_Valid) Verify that users are provided clear language regarding collection and use of supplied personal information and that users have provided opt-in consent for the use of that data before it is used in any way.

Created a new tab with the privacy policy.

8.3.5(Not\_Valid) Verify accessing sensitive data is audited (without logging the sensitive data itself), if the data is collected under relevant data protection directives or where logging of access is required.

I added log statements to capture user access and access to payment-related information without logging the actual sensitive data.

8.3.6(Not\_Valid) Verify that sensitive information contained in memory is overwritten as soon as it is no longer required to mitigate memory dumping attacks, using zeroes or random data.

In the "Payment completed" branch, sensitive data ($last\_payment) is being unset.

To more explicitly overwrite the sensitive data in memory I set the variable to a zero.

A screen shot of a computer code

Description automatically generated

8.3.7(Not\_Valid) Verify that sensitive or private information that is required to be encrypted, is encrypted using approved algorithms that provide both confidentiality and integrity.

I included a randomly generated encryption key for each session and uses AES-256 in CBC mode to encrypt the payment information.

8.3.8(Not\_Valid) Verify that sensitive personal information is subject to data retention classification, such that old or out of date data is deleted automatically, on a schedule, or as the situation requires.

I introduced a mechanism to automatically delete old session data by checking the time difference between the current time ($t) and the last activity time ($\_SESSION['last\_activity']). If the time difference exceeds the specified maximum session lifetime, it unsets and destroys the session data, logging the action.