Other OWASP Top 10 Security Vulnerabilities which can only be explained Verbosely

1. Broken Access Control:

a. Access control refers a system that controls access to information or functionality. Broken access controls allow attackers to bypass authorization and perform tasks as though they were privileged users such as administrators. For an example a web application could allow a user to change which account they are logged in as simply by changing part of a url, without any other verification.

b. Access controls can be secured by ensuring that a web application uses authorization tokens* and sets tight controls on them.

c. *Many services issue authorization tokens when users log in. Every privileged request that a user makes will require that the authorization token be present. This is a secure way to ensure that the user is who they say they are, without having to constantly enter their login credentials.

d. In PHP, make sure you use SESSIONS for every logged in user, and let each user group has its own privileges, privileges should be secured from manipulations.

e. This vulnerability depends on the web project's logic of user access control, and it is almost a manual vulnerability to be checked and tested.

To learn more:

https://www.youtube.com/watch?v=P38at6Tp8Ms
https://cheatsheetseries.owasp.org/cheatsheets/Access_Control_Cheat_Sheet.html
https://owasp.org/www-project-top-ten/OWASP_Top_Ten_2017/Top_10-2017_A5-Broken_Access_Control

2. Insecure deserialization:

a. This threat targets the many web applications which frequently serialize and deserialize data. Serialization means taking objects from the application code and converting them into a format that can be used for another purpose, such as storing the data to disk or streaming it. Deserialization is just the opposite: converting serialized data back into objects the application can use. Serialization is sort of like packing furniture away into boxes before a move, and deserialization is like unpacking the boxes and assembling the furniture after the move. An insecure deserialization attack is like having the movers tamper with the contents of the boxes before they are unpacked.

b. An insecure deserialization exploit is the result of deserializing data from untrusted sources, and can result in serious consequences like DDoS attacks and remote code execution attacks. While steps can be taken to try and catch attackers, such as monitoring deserialization and implementing type checks, the only sure way to protect against insecure deserialization attacks is to prohibit the deserialization of data from untrusted sources.

c. This vulnerability depends on the web project's logic of user input and object handling, for example: `id` property is a number, check if it is not a number or if it does not exist or if it belongs to another user! so every column has its own logic of usage, developers shouldn't trust the input object from users, and must validate them all.

To learn more:

https://www.youtube.com/watch?v=nkTBwbnfesQ
https://cheatsheetseries.owasp.org/cheatsheets/Deserialization_Cheat_Sheet.html
https://owasp.org/www-project-top-ten/OWASP_Top_Ten_2017/Top_10-2017_A8-Insecure_Deserialization

3. Insufficient Logging and Monitoring:

a) Many web applications are not taking enough steps to detect data breaches. The average discovery time for a breach is around 200 days after it has happened. This gives attackers a lot of time to cause damage before there is any response. we recommend web developers to implement logging and monitoring as well as incident response plans to ensure that they are aware of attacks on their applications.

b) Logging and monitoring are features based on the hosting server and web project's code.

1. Some other Advices:
    i. Do logging of important situations such as login and logout and while error happens
    ii. Inner error reporting tool, run technical error reporting function in critical places in your code, where errors shouldn't be happening or user shouldn't be there etc..
    iii. Limit logging per sometime so logging system does not overflow, if DDoS attack occurred.

To learn more:

https://www.youtube.com/watch?v=IFF3tkUOF5E
https://cheatsheetseries.owasp.org/cheatsheets/Logging_Cheat_Sheet.html
https://owasp.org/www-project-top-ten/OWASP_Top_Ten_2017/Top_10-2017_A10-Insufficient_Logging%252526Monitoring