Austin Cupp

CS 462

10/26/2022

1. Resource leak- Resource leaks are bugs that arise when a program doesn't release the resources it has acquired. Typically, this happens when there is a bug in the program. The types of resource leaks include memory leak and handle leak, particularly file handle leaks, though memory is often considered separately from other resources. Resource leaks can be prevented or fixed by using resource management, in that programming techniques or language constructs may prevent leaks by releasing resources promptly, while a separate process may reclaim resources that have been leaked. Most resource leaks are fixed by resource reclamation by the main operating system after the process terminates and then makes an exit system call.

6. Buffer overrun- occurs when the size of information written to a memory location exceeds what memory was allocated in the system. Buffer overruns can cause data to become corrupt, programs to crash, or even the execution of malicious code. Fixes for buffer overruns include using an interpreted language which isn't prone to buffer overrun issues, using a compiler that can help identify errors as well as unsafe functions, avoid using functions that do not perform buffer checks, for example in the C language, use fgets()) function instead of gets(), and also re-arranging the local variables so scalar variables which are individual fixed-size data objects, are above array variables, which contain multiple values.

8. Error handling issues- These occur when error messages contain details that might be useful to an attacker. When the information is present, this would be evidence of improper error handling. In most cases, improper error handling occurs as a result of default error handlers being used. An important first step to having a secure system is finding and replacing the error handlers with more secure approaches. When detailed error messages are required for developers, a secure error handler writes the error details to a log while providing a message to users that is more understandable, as well as it avoids revealing sensitive information that could be used by an attacker.

10. Concurrency issues- occur when updates are lost, for example when a row is deleted from an application, and when one application updates a value and the other reads it before it is committed, leaning to invalid data if there are changes. Fixes include generating a timestamp or hash every time a request is made, setting the request id as an additional variable with the suggestion list, and using read before write.

11. Insecure data handling- It occurs when data is not securely stored on the device, and when data is accessible to an attacker. One fix for insecure data is to actually not store data unless it is completely necessary, in that as soon as data becomes stored on a device, it becomes vulnerable to being attacked. Another fix includes not storing log in credentials in the system. Instead, make it so the user must identify themselves with some form of login procedure each time the application is opened and that the appropriate session timeouts are put into place. Also, adding another layer of encryption beyond the default encryption methods provided by the operating system is a good way to help keep data secured.

References:

Kaczanowski, M. (2021, April 28). *What is a buffer overflow attack – and how to stop it*. freeCodeCamp.org. Retrieved October 26, 2022, from https://www.freecodecamp.org/news/buffer-overflow-attacks/

Pentoma. (2022, July 21). *Understanding insecure data storage & how to approach it*. APP SECURITY INSIGHTS. Retrieved October 26, 2022, from https://blog.se.works/2017/05/25/understanding-insecure-data-storage-how-to-approach-it/