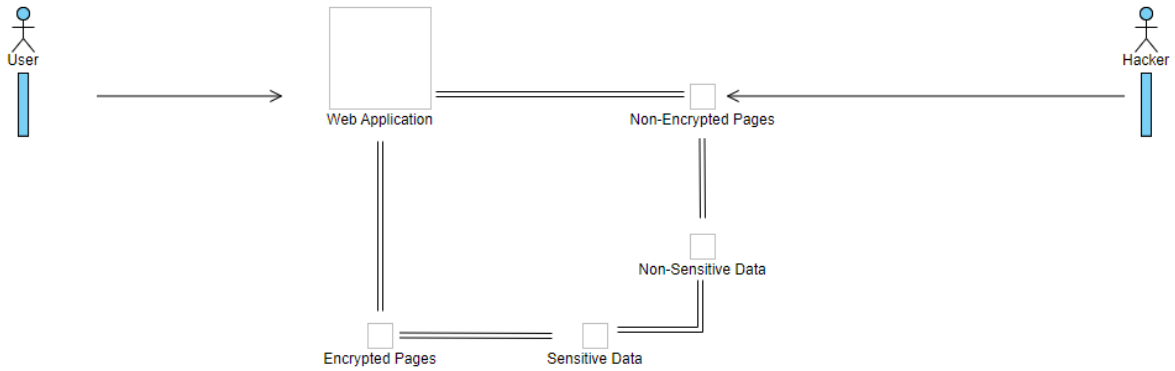
Decrypt Sensitive Data

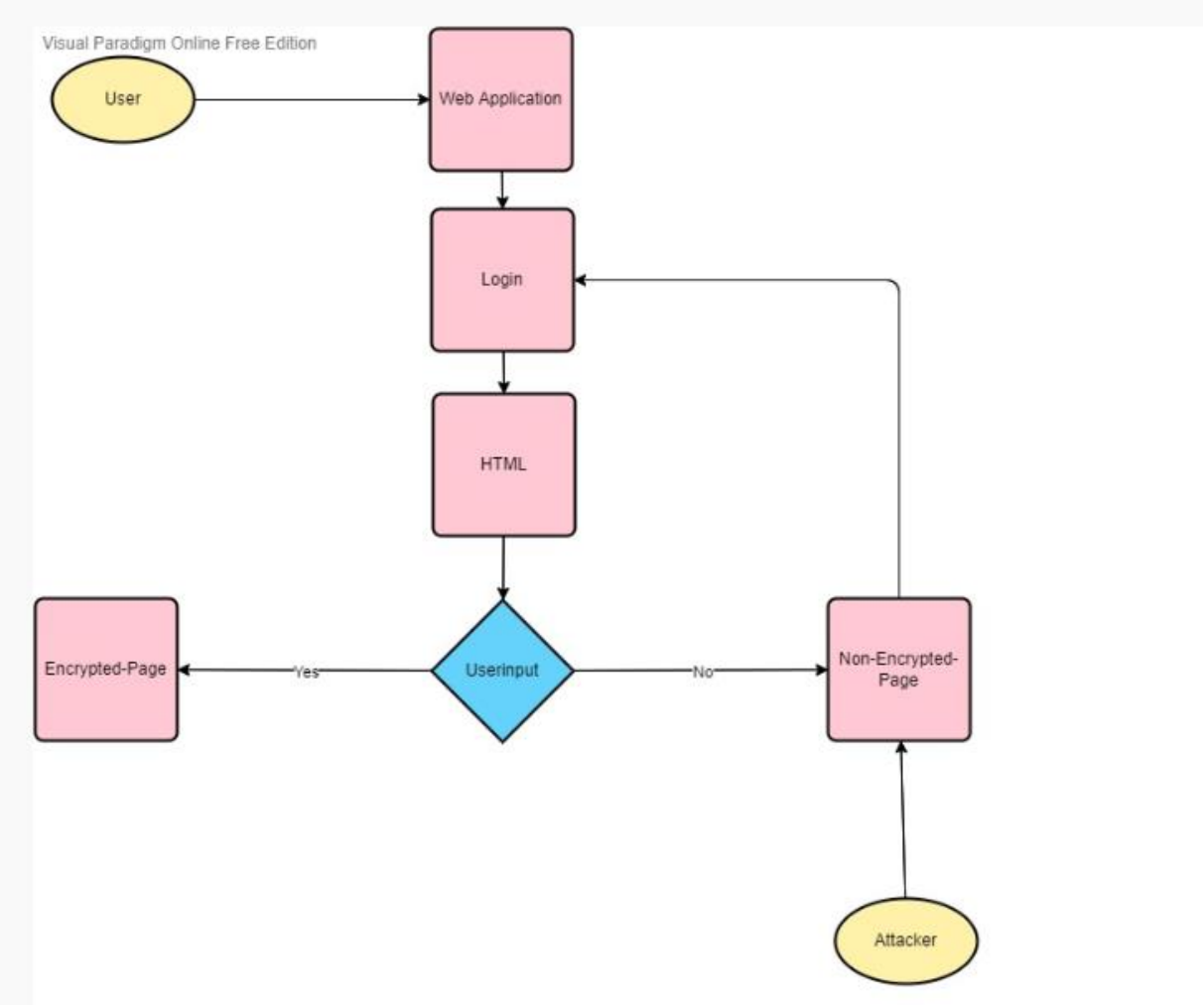
Nowadays, it is common praxis to encrypt sensitive data while building a web application. Often the encryption (and also decryption) occurs at the moment when the data is entering (or leaving) the connected database (F5DevCentral, 2018). This means, if an attacker would have access to the database, he also had to know how to decrypt the data. However, sometimes developer doesn't use encryption on every page. Imagine a case where one page doesn't have any input field but only uses images and the app only checks whether the user is allowed to enter the page (checking username and password). Now, if an attacker gets access to the whole system (server of web application, the local machine of the user and the database where the web application is connected to), then he can connect a "real" person to the user and therefore to his data in the database. The final step would then be to find the key to encrypt the cipher text and the whole database would be insecure (perhaps the attacker already found useful information on the local machine of the user that can help to decipher the encryption key for some rows, columns or tables) (F5Dev Central, 2018) (Mitre, 2017).

The figure below tries to depict that encrypted and non-encrypted are often related and should better be both encrypted. It uses a component diagram which reflects the complex idea of insecure (sensitive) data storage in using only a small number of components (Amber, 2003). Reading it from left to right and top to bottom, the user wants to use the web application. The application itself uses pages which can use both, encryption or not, depending on the kind of data that have been retrieved/sent. Nevertheless, if the user is still logged in while visiting the non-encrypted pages, Non-sensitive and Sensitive Data could be connected (see explanation in paragraph 1).

To access this cycle, the only thing a hacker needs to do, is to be present when a user gets on the Non-Encrypted page (F5DevCentral, 2018).



A flowchart can also help to understand the idea (Figure 2). The user uses a web application which asks for a login (username and password) and the user then has to choose between different Html's (for example in using buttons on the index page). After that, probably some pages ask the user for more inputs (which would perhaps transmit more sensitive data) whereas others don't and show only an output (an image for example). The problem then is that it can happen that a developer forgets that the user is already logged in and that there is therefore already a check of sensitive data in the background running logic file which makes it easier for an attacker to get access to when that page is not protected by encryption. If so, the hacker could use username and password of the user for further damages.

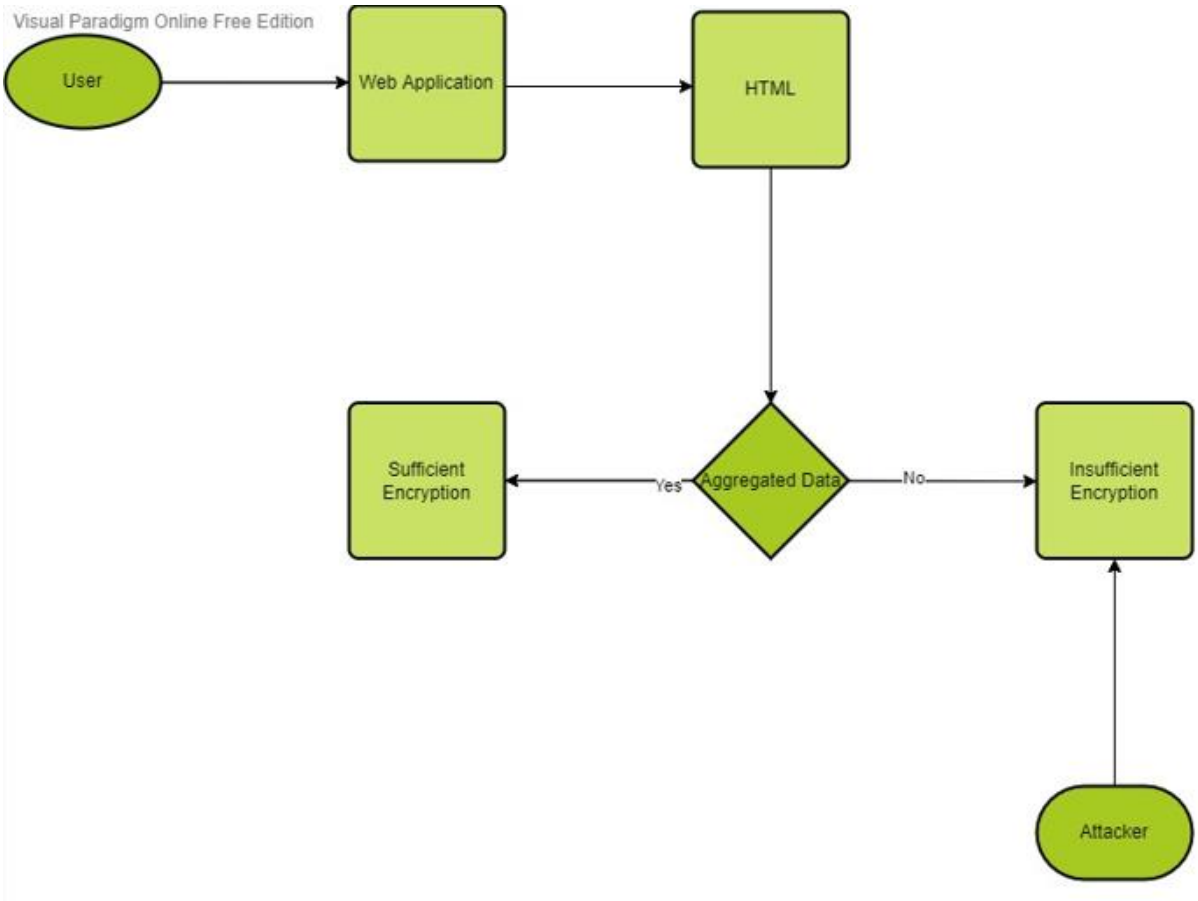


Final Reflection:

Looking back to the example chosen here, I can admit that the explanation wasn't that suited. After some reflection, what I tried to elaborate is the idea that when developing and encrypting an API (and I really mean here the encryption that occurs at the application logic files) that developers often think about encryption when dealing with sensitive data is obvious. A simple example is, that a social security number of a user has been sent to the database. However, sometimes sensitive data is also (or can be) captured in spaces where one would not imagine to see them, for example an image. Following that idea, if there is no (or not sufficient) encryption of an image file, then the attacker will find sensitive data of a user in simply going into

the details of that file. As an example, think about the idea of an image file that uses black space to hide sensitive data against an image that is not protected at all.

To illustrate this idea, I adapted the flowstart above to the follwing one (see below). Short explanation, the user uses a web application an chooses one page (or HTML). When opening the page, the important question is whether the sensitive data used are aggregate data or not. If not, the web page always uses sufficient encryption. If aggregated data are used, the encryption will be insufficient and the attacker can get access to the the system. As explained, this is only the example of a weak web page which doesn't use encryption sufficient enough.

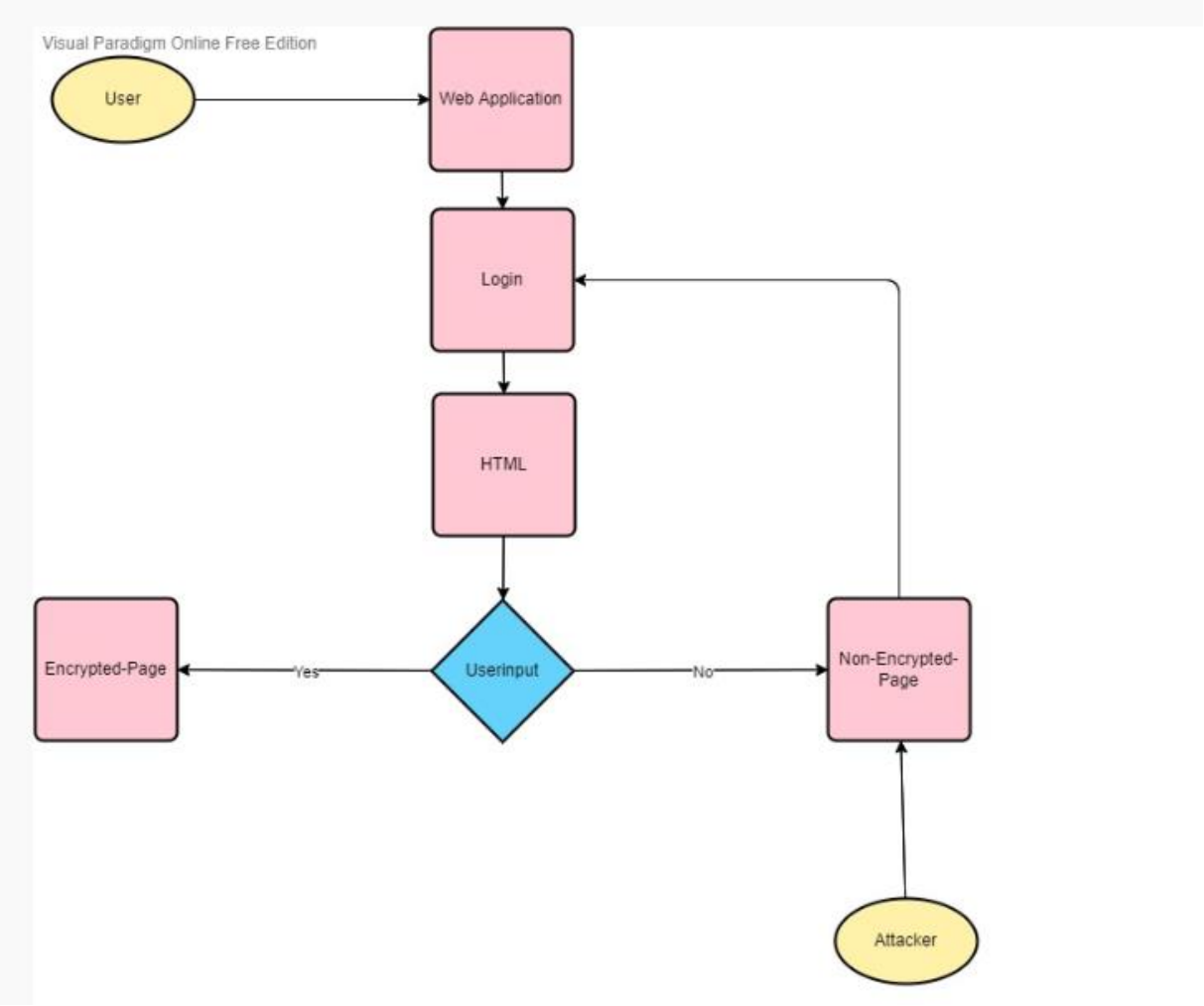


Reference List:

Ambler, S. (2003) *Elements of UML Style.* Cambridge: Cambridge University Press

F5DevCentral (2018) 2017 OWASP Top 10: Sensitive Data Exposure. Available from: https://www.youtube.com/watch?v=2RKbacrkUBU [Accessed 22 June 2022]

Mitre (2017) Weaknesses in OWASP Top Ten.



Hello Dr. Cathryn,

I hope this is better. The user uses a web application which asks for a login (username and password) and the user then has to choose between different Html's (for example in using buttons on the index page). After that, probably some pages ask the user for more inputs (which would perhaps transmit more sensitive data) whereas others don't and show only an output (an image for example). The problem then is that it can happen that a developer forgets that the user is already logged in and that there is therefore already a check of sensitive data in the background running logic file which makes it easier for an attacker to get access to when that page is not protected by encryption. If so, the hacker could use username and password of the user for further damages.

Reference List:

Ambler, S. (2003) *Elements of UML Style.* Cambridge: Cambridge University Press

F5DevCentral (2018) 2017 OWASP Top 10: Sensitive Data Exposure. Available from: https://www.youtube.com/watch?v=2RKbacrkUBU [Accessed 22 June 2022]

Mitre (2017) Weaknesses in OWASP Top Ten.