A3 - Sensitive Data Exposure

When an application does not sufficiently protect sensitive information the Sensitive Data Exposure will occur. The data that can be exposed are the passwords, credit card info and other sensitive data. When user input any information on a web application the server will be the one who will protect the sensitive data that the user entered. When a web application fails to protect sensitive information from being revealed to illegal users this is where hackers will have an access to the sensitive data.

Medication

What are the ways to prevent this kind of threat? In this kind of threat you need effective implementation of security. Encrypting the data and defining accessibility, whether in stored and transit form it is important to encrypt the data. Limit the user who can access you web application, by enforcing key based encryption. Secure authentication gateways, to ensure that all the data passed between the browser and web server will remain private use SSL or TSL for security. Prevent password attacks, when you allow the user to use weak password it may be the cause of security break. Apply hashing function algorithm to make sure that the password is strong. Have a backup plan, keeping a secure and protected backup of the sensitive data can help minimize these damages.

Example

An application encrypts credit card numbers in a database using automatic database encryption. However, this data is automatically decrypted when retrieved, allowing an SQL injection flaw to retrieve credit card numbers in clear text.

The password database uses unsalted or simple hashes to store everyone's passwords. A file upload flaw allows an attacker to retrieve the password database. All the unsalted hashes can be exposed with a rainbow table of pre-calculated hashes. Hashes generated by simple or fast hash functions may be cracked by GPUs, even if they were salted.

References:

https://www.owasp.org/index.php/Top\_10-2017\_A3-Sensitive\_Data\_Exposure

# A8-Insecure Deserialization

When untrusted data is used to abuse the logic of an application, inflict a denial of service attack, or even execute arbitrary code upon it being deserialized. So in order for us to understand what is “Insecure Deserialization” let us understand first what is serialization and deserialization. Serialization is the process of converting an object into a format which can be persisted to disk, sent through streams, or sent over a network. Deserialization is the opposite of serialization. The impact of deserialization flaws cannot be overstated. These flaws can lead to remote code execution attacks, one of the most serious attacks possible. The business impact depends on the protection needs of the application and data.

Medication

What are the ways to prevent this kind of threat? Implementing integrity checks such as digital signatures on any serialized objects to prevent hostile object creation or data tampering. Isolating and running code that deserializes in low privilege environments when possible. Restricting or monitoring incoming and outgoing network connectivity from containers or servers that deserialize. Monitoring deserialization, alerting if a user deserializes constantly.

Example

A React application calls a set of Spring Boot microservices. Being functional programmers, they tried to ensure that their code is immutable. The solution they came up with is serializing user state and passing it back and forth with each request. An attacker notices the "R00" Java object signature, and uses the Java Serial Killer tool to gain remote code execution on the application server. A PHP forum uses PHP object serialization to save a "super" cookie, containing the user's user ID, role, password hash, and other state.

References:

<https://www.owasp.org/index.php/Top_10-2017_A8-Insecure_Deserialization>

<https://dzone.com/articles/what-is-insecure-deserialization>

# A9-Using Components with Known Vulnerabilities

This kind of threat occurs when the components such as libraries and frameworks used within the app almost always execute with full privileges. If a vulnerable component is exploited, it makes the hacker’s job easier to cause a serious data loss or server takeover. Vulnerabilities in third-party libraries and software are extremely common and could be used to compromise the security of systems using the software. Only some of these vulnerabilities are relevant to GitHub, but it can be quite a challenge to keep track of these on a day-to-day basis. Some vulnerable components (e.g., framework libraries) can be identified and exploited with automated tools, expanding the threat agent pool beyond targeted attackers to include chaotic actors.

Medication

To prevent this kind of threat you need to Identify all components and the versions that are being used in the webapps not just restricted to database/frameworks. Add security wrappers around components that are vulnerable in nature. Keep all the components such as public databases, project mailing lists etc. up to date. If it is unavoidable, then it should be done without involving user parameters in redirecting the destination. It is better to avoid using redirects and forwards.

Example

Let us say the application has a page - redirect.jsp, which takes a parameter redirectrul. The hacker adds a malicious URL that redirects users which performs phishing/installs malware. Remote-code execution with Expression Language injection vulnerability is introduced through the Spring Framework for Java based apps. Attackers can invoke any web service with full permission by failing to provide an identity token.

References:

<https://hdivsecurity.com/owasp-using-components-with-known-vulnerabilities>

<https://bounty.github.com/classifications/using-components-with-known-vulnerabilities.html>

https://www.tutorialspoint.com/security\_testing/unvalidated\_redirects\_and\_forwards.htm