Holmes Security/Vulnerability Threat Impact Analysis

This template is intended to be used to document the outcome of the impact analysis related to the known vulnerability reported by Nexus-IQ (CLM tab in Jenkins). Nexus-IQ can identify the known vulnerabilities contained in the components use by onap components.

This table will be presented to TSC at Code Freeze milestone (M4) to the TSC.

It is recommended to **first update to the latest version** of the third party components available. In case the latest third party components still reports some vulnerabilities, you must provide an impact analysis as illustrated in the example below.

In the case where you have nested third party components (a third party component embedding another third party component) and there is **NO CVE** number for the upstream third party component (meaning the third party component you are embedding), it is recommended to open a vulnerability issue on the upstream third party component.

Usage

Please make a **Copy** of this template into your project wiki space. Be sure to make a Copy (not a Move) by using the ... on the top right corner of this page

Within the M4 checklist create a link toward your copy of this template

Once this template has been copied into your project wiki space, you can delete this "Tip" section as well as the "Sample of CLM Report" screenshot. This screenshot is just an example.

The following table is addressing 2 different scenarios:

- Confirmation of a vulnerability including an action
- False Positive

The information related to Repository, Group, Artifact, Version and Problem Code are extracted from the CLM report (see the below screenshot)

Repository	Group	Impact Analysis	Action
holmes-common	com.fasterxml.jackson.core	False Positive Explaination: This vulnerability issue only exists if com.fasterxml.jackson.databind.ObjectMapper.setDefaultTyping() is called before it is used for deserialization. holmes-common does not use ObjectMapper for serialization/deserialization of JSON objects. Instead, Holmes uses GSON to avoid the vulnerability issues. The reason this is detected is that jackson-databind is introduced indirectly by msb-java-sdk. Also, the MSB team has declared this to be a false positive. ? Unknown Attachment	
holmes-dsa	com.fasterxml.jackson.core	False Positive Explaination: This vulnerability issue only exists if com.fasterxml.jackson.databind.ObjectMapper.setDefaultTyping() is called before it is used for deserialization. holmes-dsa does not use ObjectMapper for serialization/deserialization of JSON objects. Instead, Holmes uses GSON to avoid the vulnerability issues. The reason this is detected is that jackson-databind is introduced indirectly by msb-java-sdk. Also, the MSB team has declared this to be a false positive. ? Unknown Attachment	
holmes-engine-management	com.fasterxml.jackson.core	Explaination: This vulnerability issue only exists if com.fasterxml.jackson.databind.ObjectMapper.setDefaultTyping() is called before it is used for deserialization. holmes-engine-management does not use ObjectMapper for serialization/deserialization of JSON objects. Instead, Holmes uses GSON to avoid the vulnerability issues. The reason this is detected is that jackson-databind is introduced indirectly by dropwizard-core. ? Unknown Attachment To solve the problem, we have to replace the framework of Holmes or wait for updates from Dropwizard. From Homles perspective, we don't use Jackson for JSON data processing. So this is not a big deal for Holmes.	Need to update Dropwizard to check whether its new version has solved this problem. Otherwise, we have to switch to another framework.

holmes-rule-management	com. fasterxml .jackson. core	Explaination: This vulnerability issue only exists if com.fasterxml.jackson.databind. ObjectMapper.setDefaultTyping() is called before it is used for deserialization. holmes-rule-management does not use ObjectMapper for serialization/deserialization of JSON objects. Instead, Holmes uses GSON to avoid the vulnerability issues. The reason this is detected is that jackson-databind is introduced indirectly by dropwizard-core. To solve the problem, we have to replace the framework of Holmes or wait for updates from Dropwizard. From Homles perspective, we don't use Jackson for JSON data processing. So this is not a big deal for Holmes.	Need to update Dropwizard to check whether its new version has solved this problem. Otherwise, we have to switch to another framework.