Artemis Threat Modeling:

Tampering

Threat: Users sending messages to Artemis queues without having the permission

Mitigation: Prevent this by mapping roles to users to give certain permissions (using Artemis embedded security)

Damage Potential: 1

Reproducibility: 3

Exploitability: 3

Affected Users: 2

Discoverability: 2

Total: 1 + 3 + 3 + 2 + 2 = 11

Non-Repudiation

Threat: How do we know the person who sent/consumed a message really consumed that message? (Not being able to track users sending/consuming messages)

Mitigation: Audit all messages being sent/consumed

Damage Potential: 3

Reproducibility: 3

Exploitability: 3

Affected Users: 1

Discoverability: 1

Total: 3 + 3 + 3 + 1 + 1 = 11

Information Disclosure

Threat: Users consuming messages they are not authorized to consume

Mitigation: (Same as Tampering)

DREAD: (Same as Tampering)

Denial-Of-Service

Threat: If a poison message is sent to an Artemis queue, the message fails and attempts to be resent 10 times, thus slowing the system down especially if numerous poison messages are being sent

Mitigation: Input validation? Flow control by limiting memory for a set of queues? Using human interaction to notice that the DLQ is being hammered and taking action on the user sending poison messages? Needs more investigation.

Threat: A slow consumer can cause a DOS attack. The consumer can’t keep up with consuming messages on a queue therefore the Artemis queue backs up and eventually casing Artemis to die

Mitigation: Artemis knows who is a slow consumer and can kill their connection

Damage Potential: 3

Reproducibility: 3

Exploitability: 1

Affected Users: 3

Discoverability: 3

Total: 3 + 3 + 1 + 3 + 3 = 13

Can DDF get a DOS attack from Artemis?

Threat: A very large image gets published to Artemis, the message gets processed by some DDF service, but DDF doesn’t actually have the memory to process it, causing an OOM exception for DDF.

Mitigation: Reject the image if it’s too big? Think about how we can change the processing for DDF.