Artemis Threat Modeling

**Affects Processes:**

External Client 🡪 Artemis

Spoofing:

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 sent/consumed that message? (Not being able to track users sending/consuming messages)

Mitigation: Audit all messages being sent/consumed, signing messages

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 Spoofing) also using SSL connections

DREAD: (Same as Spoofing) also using SSL connections

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. Expire message to prevent them from filling up a queue? Using last value queues to maintain only one copy of the same message? Limiting the address size from the memory space?

Damage Potential: 3

Reproducibility: 3

Exploitability: 1

Affected Users: 3

Discoverability: 3

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

Tampering:

Threat: Consuming a message from a queue, modify the message maliciously and put it back on the queue

Mitigation: Artemis roles, use input and output address, people can only write to specific queues and people can only read from specific address

Using virtual queues/topics that DSD could send/consume messages to

Elevation of Privilege: Not really a problem for Artemis as there is no concept of a system user

Artemis 🡪 DDF

Spoofing:

Threat: Artemis sending messages to DDF without the permissions, how does DDF know that Artemis is who they say they are?

Mitigation: Using Artemis embedded security, sending and maintaining a subject across the broker to identify who is doing what when performing operations

Non-repudiation:

Threat: How do we know that when Artemis sends a message to DDF, they are who they say they are?

Mitigation: Auditing who is performing what operation, sending a subject across the broker, using an interceptor to grab the message to figure out who it came from

Information Disclosure: (Same as from an external client to Artemis)

Tampering: (Same as from an external client to Artemis)

Denial-Of-Service:

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

Elevation of Privilege: Trust the people who push to queues/topics

DDF 🡪 Artemis

Trusted and the interactions are the same as from an external client to Artemis

Outstanding Questions:

How do we check that we know who sent a message? And who wrote it the message out?

How to we protect from incoming (write) DOS attacks?

How do we perform authentication for the broker?

How do we maintain identity across the broker?