KIP-587: Suppress detailed responses for handled exceptions in security-sensitive environments

This KIP is aimed at providing an alternative error body that never includes a stack trace in the message passed in REST responses by Kafka Connect. For example, when Kafka Connect handles a catch-all exception, it prints the message of that exception, which could lead to future security issues and disclosures of information not desired in sensitive environments. In environments that require high levels of security, security teams have requested Kafka Connect intentionally obfuscate these messages. This KIP would allow these environments to anticipate this intentionally obfuscated error response in exchange for higher security.

Status

Current state: Under Discussion

Discussion thread: here

JIRA: KAFKA-9766

Please keep the discussion on the mailing list rather than commenting on the wiki (wiki discussions get unwieldy fast).

Motivation

It is expected that all unhandled exceptions be caught by a catch-all exception mapper. However, in some sensitive environments, it is more harmful than helpful to disclose what the nature of that unhandled exception is. In most environments, it is useful to know the nature of all exceptions, so that errors can be caught by monitoring systems. In this case, we are aware of customers who have had to scrutinize their use of Kafka Connect in these secure environments because of their inability to tune this response. It should be noted that we have not found a specific vulnerability in relation to KafkaConnect, but are attempting to reduce the ability for a theoretical attacker (or audit team) to fuzz APIs and produce more information that is acceptable in highly secure environments. Additionally, this KIP asks that we examine the code base for additional instances of stack traces in handled/OK/200 responses to provide an alternative body for the stack trace.

Public Interfaces

We are proposing an alternative expected value to a variety of REST responses. This means that the JSON schema for affected responses should not change, but the values in that schema, for example, a stack trace, would be replaced by a static string.

Proposed Changes

We are proposing a config-based behavior branch in the ExceptionMapper, which would have an alternative behavior of printing a static SUPRESSED_EXCEPTION_MESSAGE (Suggested: "Detailed exception information has been suppressed. Please see logs." in this PR) instead of reading the exception.getMessage(). Additionally, we would identify classes that print stack traces in their REST responses, such as `/connectors/{name}/status`, which prints the "trace" field, and implement an alternative behavior as in the ExceptionMapper.

This ExceptionMapper behavior change and other affected classes should be configured to be changed based on the worker config. Proposed is a new configuration called **enable.rest.response.stack.traces** which would default to true. Disabling message detail would enable the “hardened security mode.”

This KIP includes handling all exceptions handled by the ExceptionMapper, including ConnectRestExceptions. An example of this is here, in this PR on GitHub.

Compatibility, Deprecation, and Migration Plan

- If enabled, the output of the API on an error response will change. The schema of the REST response should be the same.
- The old behavior will remain the default, this KIP asks the community to support an additional way of reporting errors into the future
- No tools are needed for migration
- The existing behavior will not be deprecated

Rejected Alternatives

We have considered putting an NGINX reverse proxy in front of Kafka Connect in these environments, but we rejected this alternative because this kind of web application firewall becomes heavy for something as lightweight as Kafka Connect

Proposed by the community was a Connect REST extension as an alternative to an Nginx proxy. We had not considered using one. It was suggested that a REST extension shouldn't be necessary in order to lock down the REST API in the way proposed by the KIP. It was deemed an acceptable workaround in cases where users are stuck on a given version of Connect and can't upgrade to the upstream version of Kafka that this KIP is merged into.

Tuning ssl.* parameters to provide guarantee of privacy (encryption) and authenticity (PKI) does not provide us with authentication, which would provide a "gate" around the ability to fuzz, and an audit trail on "who is fuzzing," except by IP, which could then be combined with Fail2Ban or similar log analysis mitigations. Since authorization seems like a more major change, we're suggesting that we just remove the data we've considered "harmful."