🛠️ AutoSRE - Incident Analyzer

Analyze incidents, fetch similar past issues, and recommend resolutions automatically!

Enter Incident Summary/Exception/Stacktrace:

```
com.auth.exceptions.AuthenticationException: An error occurred\n    at
com.example.module.Class.method(Class.java:25)\n    at
com.example.module.OtherClass.anotherMethod(OtherClass.java:50)\n    at
java.base/java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1128)\n    at
java.base/java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:628)\n    at
java.base/java.lang.Thread.run(Thread.java:829)
```

Analyze Incident

Root Cause Identified!

A nice Java stack trace!

Based on the error message and the call stack, I would identify the most likely root cause as:

Insufficient thread pool configuration or contention

The exception is an `AuthenticationException`, which suggests that the issue is related to authentication or authorization. The fact that it's occurring in a `ThreadPoolExecutor` (specifically, in a worker thread) implies that there might be contention or insufficiency in the thread pool configuration. This could lead to delays or timeouts during the authentication process, resulting in an exception being thrown.

Additional insights: It's possible that the application is trying to perform an authentication operation concurrently with another long-running task, leading to resource competition and eventually throwing this exception.

Similar Past Incidents

Incident 1: Date: 2025-01-14 Incident_Number: INC1741 Incident_Category: Security Incident_Summary: Suspicious login detected Incident_Details: Multiple failed login attempts from unusual IP addresses. Incident_RCA: Brute force attack from external IP Incident_Impact: Potential unauthorized access risk Incident_Resolution: Blocked IP and enforced MFA Incident_Errors: Login attempts exceeded Incident_Exception: SecurityBreachException Incident_Full_Exception: com.security.exceptions.SecurityBreachException Incident_Stacktrace: com.security.exceptions.SecurityBreachException: An error occurred at com.example.module.Class.method(Class.java:25) at com.example.module.OtherClass.anotherMethod(OtherClass.java:50) at java.base/java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1128) at java.base/java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:628) at java.base/java.lang.Thread.run(Thread.java:829)

Recommended Resolution

Here are the resolution steps:

Step 1: Check thread pool configuration

- Verify that the thread pool size is set correctly in your application. If it's too small, consider increasing it to handle concurrent requests.
- Check if there are any long-running tasks or blocking operations that might be consuming resources and causing contention.

Sample Code Snippet:

```
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

// Create a thread pool with a reasonable size (e.g., 5-10 threads)
ExecutorService executor = Executors.newFixedThreadPool(5);

// Submit tasks to the executor
executor.submit(task1);
executor.submit(task2);

// Shutdown the executor when done
executor.shutdown();
```

Step 2: Review affected files and check for changes

- Check the GitHub commit history to see if any relevant code changes have been made recently.
- If yes, review the file diff (using `git diff` or a GUI client like GitHub Desktop) to identify potential issues.

File Diff Example:

```
--- a/path/to/AffectedClass.java
+++ b/path/to/AffectedClass.java
@@ -10,7 +10,8 @@
public class AffectedClass {
    // Original code
-   private static final int THREAD_POOL_SIZE = 5;
+   private static final int THREAD_POOL_SIZE = 2;  // reduced pool size?
    // ...
```

Step 3: Troubleshoot and optimize

- If the issue persists, try increasing the thread pool size to see if it resolves the problem.
- If the error still occurs, consider using a more advanced concurrency framework like Java Concurrency API (java.util.concurrent) or Apache Commons Lang's `ThreadUtils` class.

Remember to always monitor your application's performance and resource utilization after making changes.
