Security in Neo4J

Introduction

- To secure your application, you need to be concerned about 3 things:
  - Data security
  - Access control
  - User privacy

Securing a Neo4j application

- Neo4j applications consist of many parts, including
  - Databases
  - Static files like images and document scans
  - Application code
  - Application servers and
  - Web servers
- These all need to be secured

How to secure data-in-transit?

- If your application has users that access the application over the Internet, you should implement SSL/TLS

Securing data-at-rest

- Means securing private data - databases and backups
  - Securing files at the OS level
  - Ensure that Neo4j does not run as root
    - By default the user, neo4j owns the processes running that are related to the Neo4j instance.
    - You can determine which processes are owned by neo4j using
      - `ps -ef |grep -E "neo4j"`

Securing data-at-rest

- Neo4j file permissions
  - The following directories should be read-only:
    - conf
    - import
    - bin
    - lib
    - plugins
  - The following directories should be read/write:
    - data
    - logs
    - metrics
  - Should also set the log files to only be writable by neo4j and readable by root

Security auditing

- Must have a plan for
  - Analyzing a suspected security breach
- As part of that plan
  - Should implement a security audit trail that captures which users logged in successfully and those that failed
- Some recommended settings for security auditing in Neo4j:
  - dbms.directories.logs=logs   # root directory where the general log files are located
  - dbms.logs.security.level=INFO # DEBUG, INFO, WARN and ERROR.
  - dbms.logs.security.rotation.size=20m   # Threshold for rotation of the security log.
  - # Minimum time interval after last rotation of log before it may be rotated again.
  - dbms.logs.security.rotation.delay=300s
  - # Maximum number of history files for the security log. Sets # of historical log files kept
  - dbms.logs.security.rotation.keep_number=7