

NEO4J SECURITY BENCHMARK

Version 4.3

11 October 2021

Contents

Introduction 4

Overview 4

Intended Audience 4

Changes in Neo4j 4.3 4

Installation and Patching 5

Ensure the appropriate Neo4j software version/patches are installed 5

Authentication 6

Ensure that authentication is enabled for Neo4j databases 6

Ensure authentication is enabled in the Neo4j cluster 7

Use Neo4j with an LDAP Auth Provider 8

Use Neo4j with Kerberos (Optional) 10

Configure User Authentication Maximum Failed Attempts 11

Configure User Authentication Lockout Time 12

Securing Configuration Settings 13

USE neo4j-ADMIN start --expand-commands 13

Encryption 14

Require TLS for Bolt Connections 14

Install a signed TLS certificate 15

OCSP Stapling 16

Disable HTTP Port 17

Neo4j Browser 18

Ensure Neo4j Browser does not cache login information 18

Implement Neo4j Browser Time-Out 19

Configure CORS Headers 20

Set HTTP Strict-Transport-Security (HSTS) Response Header 21

Data Encryption 22

Ensure TLS protects all network communications (cluster, browser and bolt) 22

Encrypt Data Backups 24

Change default backup port 26

Use Volume Encryption 27

Activity Logging 28

Ensure that system activity is audited 28

Operating System Hardening 30

Neo4j Database Running with Least Privileges 30
Ensure that Neo4j uses non-default ports 32
Ensure that operating system resource limits are set for Neo4j 33
Disable Deprecated Neo4j Utilities 34
JMX 34
Docker 36
Encryption 36
Sandbox Extensions 37
Sandboxing 37
Enabling Extensions 38
Role Based Access Control Authorization 39
RBAC 39
System Database 40
RBAC 40
FIPS 140.2 Support 41
FIPS 140.2 Configuration 41
Deprecated Settings 42
Deprecations 42

Introduction

Overview

This document, Neo4j Security Benchmark, provides prescriptive guidance for establishing a secure configuration posture for Neo4j Enterprise Edition versions 4.3.0 and higher. This guide was tested against Neo4j 4.3.0 running on Ubuntu Linux 18.04 and Microsoft Windows Server 2016. This guide applies to other Linux distributions. If you have questions, comments, or have identified ways to improve this guide, please write us at feedback@neo4j.com.

(This document should integrate and incorporate present information on the public website available here: https://neo4j.com/docs/operations-manual/current/security/checklist/)

Intended Audience

This document is intended for system and application administrators, security specialists, auditors, help desk, and platform deployment personnel who plan to develop, deploy, assess, or secure solutions that incorporate Neo4j Enterprise Edition.

Changes in Neo4j 4.3

Neo4j 4.3 provides the following enhancements regarding security configuration:

- Add a new setting **dbms.security.ldap.authentication.attribute**, used to configure a custom attribute for looking up LDAP users. This functionality earlier required an external plugin.
- Add a setting **dbms.security.ldap.authentication.search_for_attribute** to enable looking up LDAP users. This new setting replaces **dbms.security.ldap.authentication.use_samaccountname**, which has been deprecated.
- Added **dbms.security.ldap.authorization.access_permitted_group** to restrict LDAP authorization to a particular group

Installation and Patching

This section provides guidance on ensuring that the Neo4j software is up to date to eliminate easily avoidable vulnerabilities.

Ensure the appropriate Neo4j software version/patches are installed

DESCRIPTION:

The Neo4j installation version, along with the patch level, should be the most recent that is compatible with the organization's operational needs.

RATIONALE:

Using the most recent Neo4j software version along with all applicable patches helps limit the possibilities for vulnerabilities in the software. The installation version and/or patches applied should be selected according to the needs of the organization. At minimum, the software version should be supported. The latest releases can be found at: https://neo4j.com/download-center/. As of October 2021, Neo4j 3.5.28, Neo4j 4.0.12, Neo4j 4.2.11 and Neo4j 4.3.5 are the latest supported versions.

REMEDIATION:

Upgrade to the latest version of the Neo4j software. Refer to the online operations manual for the recommended upgrade in

https://neo4j.com/docs/upgrade-migration-guide/current/supported-paths/

DEFAULT VALUE:

None

REFERENCES:

1. https://neo4j.com/docs/operations-manual/current/installation/
2. https://support.neo4j.com/hc/en-us/articles/115013134648-Neo4j-Supported-Versions
3. https://support.neo4j.com/hc/en-us/articles/203307636-Downloads-for-Commercial-Customers

Authentication

This section contains recommendations for requiring authentication before allowing access to the Neo4j database.

Ensure that authentication is enabled for Neo4j databases

DESCRIPTION:

This setting ensures that all clients, users, and/or servers are required to authenticate prior to being granted access to the Neo4j database.

RATIONALE:

Failure to authenticate clients, users, and/or servers can enable unauthorized access to the Neo4j database and can prevent tracing actions back to their sources.

REMEDIATION:

The authentication mechanism should be implemented before anyone accesses the Neo4j Server.

To enable the authentication mechanism: In the **neo4j.conf** file, uncomment the **dbms.security.auth_enabled** setting:

```
# Whether requests to Neo4j are authenticated.
# To disable authentication, uncomment this line
dbms.security.auth_enabled=true
```

Modify the Neo4j user password, ensuring that its password meets organizationally defined password complexity requirements.

IMPORTANT: Use the **set-initial-password** command of **neo4j-admin** to define the password for the native user **neo4j**. This must be performed before starting up the database for the first time.

DEFAULT VALUE:

Authentication is enabled by default with the neo4j username and neo4j password.

REFERENCES:

1. https://neo4j.com/docs/operations-manual/current/reference/configuration-settings/#config_dbms.security.auth_enabled
2. https://neo4j.com/docs/operations-manual/current/configuration/set-initial-password/

Ensure authentication is enabled in the Neo4j cluster

DESCRIPTION:

Authentication is enabled in a Neo4j cluster when keyfiles are created and configured for all components. This ensures that every client that accesses the cluster must provide credentials, to include Neo4j instances that access each other within the cluster.

RATIONALE:

Enforcing a key on a sharded cluster prevents unauthorized access to the Neo4j database and provides traceability of database activities to a specific user or component.

REFERENCES:

1. https://neo4j.com/docs/operations-manual/current/clustering/intra-cluster-encryption/
2. https://neo4j.com/docs/operations-manual/current/security/ssl-framework/

Use Neo4j with an LDAP AuthENTICATION SERVICE

DESCRIPTION:

Configuring Neo4j to use an LDAP Auth provider offloads management of federated users to the LDAP service. The LDAP service facilities are used for administration.

RATIONALE:

Neo4j supports LDAP, which allows for integration with Active Directory (AD), OpenLDAP, or other LDAP-compatible authentication services. This means that you use the LDAP service for managing federated users, while the native Neo4j user and role administration are completely turned off.

An LDAP auth provider provides stronger user management (disable users, password changes, password complexity, etc.) than available with Neo4j native users. It also streamlines the user management across a cluster, as Neo4j does not centralize management of native users.

REMEDIATION:

The authentication mechanism should be implemented before anyone accesses the Neo4j Server.

All settings need to be defined at server startup time in the default configuration file **neo4j.conf**:

```
# Turn on security:
dbms.security.auth_enabled=true

# Choose LDAP connector as security provider for both authentication and
authorization:
dbms.security.authentication_providers=ldap
dbms.security.authorization_providers=ldap
```

See https://neo4j.com/docs/operations-manual/current/authentication-authorization/ldap-integration/ for details on configuring Neo4j to use an LDAP Auth provider.

In environments where you need both LDAP authentication as well as some native user accounts, there is a way to allow this in Neo4j 4.3. Use the configuration setting **dbms.security.auth_providers** instead of the singular version **dbms.security.auth_provider**. This will allow you to supply a list of providers to use for authentication.

DEFAULT VALUE:

Not configured.

REFERENCES:

1. https://neo4j.com/docs/operations-manual/current/authentication-authorization/ldap-integration
2. https://support.neo4j.com/hc/en-us/articles/360049019173-Neo4j-LDAP-Integration
3. https://support.neo4j.com/hc/en-us/articles/360000586268-Configure-Neo4j-to-authenticate-users-from-different-OUsusing-the-Active-Directory-attribute-samAccountName
4. https://support.neo4j.com/hc/en-us/articles/115013527168-How-do-I-allow-for-authentication-using-Active-Directoryattribute-samAccountName
5. https://support.neo4j.com/hc/en-us/articles/115008871987-How-to-configure-mixed-mode-security-native-and-LDAP-inNeo4j

Use Neo4j with Kerberos (Optional)

DESCRIPTION:

The official Neo4j Kerberos add-on can be used to extend Neo4j with Kerberos authentication.

RATIONALE:

The official Neo4j Kerberos add-on provides authentication and should be used in conjunction with another provider such as LDAP for authorization.

REMEDIATION:

Consult with Neo4j Support to configure the Kerberos add-on.

DEFAULT VALUE:

Not configured.

REFERENCES:

1. https://neo4j.com/download/neo4j-addon-kerberos/

Configure User Authentication Maximum Failed Attempts

DESCRIPTION:

When using Neo4j native user management, configure the maximum failed attempts to guard against brute force attacks.

RATIONALE:

Configuring the maximum failed attempts will help minimize the chance of a successful brute force attack (avoid an attacker guessing a password by constantly requesting authentication to the server).

REMEDIATION:

In the **neo4j.conf** file, add the **dbms.security.auth_max_failed_attempts** configuration parameter:

```
dbms.security.auth_max_failed_attempts=3
```

DEFAULT VALUE:

The default value for this parameter is 3.

NOTE:

Neo4j native user management does not have strong password requirements to include password length, complexity or expiration and may not meet stringent password requirements.

The settings are per-user, i.e. the server keeps a list of the number of consecutive failed attempts to login for each user.

The settings are global, i.e. they apply to all the users.

REFERENCES:

1. https://neo4j.com/docs/operations-manual/current/reference/configuration-settings/#config_dbms.security.auth_max_failed_attempts

Configure User Authentication Lockout Time

DESCRIPTION:

When using Neo4j native user management, configure the user lockout time after unsuccessful authentication attempts to guard against brute force attacks.

RATIONALE:

Configuring the user lockout time will help minimize the chance of a successful brute force attack (avoid an attacker guessing a password by constantly requesting authentication to the server).

REMEDIATION:

In the **neo4j.conf** file, add the **dbms.security.auth_lock_time** configuration parameter:

```
dbms.security.auth_lock_time=5s
```

DEFAULT VALUE:

The default value for this parameter is 5s. The unit of measure for this parameter is seconds.

NOTE:

Neo4j native user management does not have strong password requirements to include password length, complexity or expiration and may not meet stringent password requirements.

The settings are per-user, i.e. the server keeps a list of the number of consecutive failed attempts to login for each user.

The settings are global, i.e. they apply to all the users.

REFERENCES:

1. https://neo4j.com/docs/operations-manual/current/reference/configuration-settings/#config_dbms.security.auth_lock_time

Securing Configuration Settings

This section describes the use of neo4j expand-commands utility to minimize the exposure of sensitive username / passwords that would be stored in the neo4j.conf file.

USE NEO4J-ADMIN START --EXPAND-COMMANDS

RATIONALE:

The --expand-commands settings allows settings to be retrieved from a common repository and added to multiple instances. This allows DBAs to remove sensitive information stored in plain text in neo4j.conf.

REMEDIATION:

In addition to providing values directly for settings, each setting can optionally execute a custom script to provide the value, using the syntax dbms.setting=$(script_to_execute).

For example:

```
dbms.max_databases=$(expr 10 * 10)

dbms.ssl.policy.private_key_password=$(curl -E example.cert
https://example.com/password)
```

DEFAULT VALUE:

This feature needs to be explicitly enabled by providing the **--expand-commands** argument to the Neo4j start-up script. The server will fail to start if the config contains scripts, but the correct argument is not provided.

REFERENCES:

1. https://neo4j.com/docs/operations-manual/current/configuration/neo4j-conf/index.html - neo4j-conf-expand-commands

Encryption

This section contains recommendations for configuring Neo4j to use encryption ensuring that sensitive data and passwords are not passed over the network in clear text.

Require TLS for Bolt Connections

RATIONALE:

The bolt protocol can operate either encrypted or unencrypted. This setting ensures that unencrypted mode may not be used by clients. This in turn ensures that unencrypted data and authorization information never crosses the wire.

REMEDIATION:

In the **neo4j.conf** file, configure the **dbms.connector.bolt.tls_level** configuration parameter:

```
dbms.connector.bolt.tls_level=REQUIRED
```

DEFAULT VALUE:

The default value for this parameter is DISABLED.

REFERENCES:

2. https://neo4j.com/docs/operations-manual/current/reference/configuration-settings/#config_dbms.connector.bolt.tls_level

Install a signed TLS certificate

RATIONALE:

The bolt protocol can operate either encrypted or unencrypted. This setting ensures that unencrypted mode may not be used by clients. This in turn ensures that unencrypted data and authorization information never crosses the wire.

REMEDIATION:

Use SSL certificates issued from a trusted Certificate Authority.

DEFAULT VALUE:

Not configured.

REFERENCES:

1. For configuring your Neo4j installation to use encrypted communication, refer to https://neo4j.com/docs/operations-manual/current/security/ssl-framework/
2. For configuring your Bolt and/or HTTPS connectors, refer to https://neo4j.com/docs/operations-manual/current/configuration/connectors/
3. If using LDAP, configure your LDAP system with encryption via StartTLS; see the section called "Use LDAP with encryption via StartTLS" (https://neo4j.com/docs/operations-manual/current/authentication-authorization/ldap-integration/#auth-ldap-encrypted-starttls).

OCSP Stapling

RATIONALE:

OCSP Stapling is a mechanism for managing certificate revocation. The protocol server must provide a proof of validity along with the certificate as part of TLS handshake. Most certificates and clients support OCSP, a protocol for querying a certificate's revocation status and confirming that it is still valid.

REMEDIATION:

In the **neo4j.conf** file, configure the **dbms.connector.bolt.ocsp_stapling_enabled** configuration parameter:

```
dbms.connector.bolt.ocsp_stapling_enabled=true
```

DEFAULT VALUE:

The default value for this parameter is FALSE.

REFERENCES:

1. https://neo4j.com/docs/operations-manual/current/reference/configuration-settings/#config_dbms.connector.bolt.ocsp_stapling_enabled
2. https://neo4j.com/docs/api/java-driver/current/org/neo4j/driver/Config.TrustStrategy.html

Disable HTTP Port

RATIONALE:

Data passed over the http port is unencrypted.

REMEDIATION:

You can disable the HTTP port by setting the **dbms.connector.http.enabled** parameter to false:

```
# HTTP Connector. There can be zero or one HTTP connectors.
dbms.connector.http.enabled=false
```

DEFAULT VALUE:

Default value for this parameter is **true**.

REFERENCES:

1. https://neo4j.com/docs/operations-manual/current/reference/configuration-settings/#config_dbms.connector.http.enabled

Neo4j Browser

This section contains recommendations for enhancing the Neo4j browser security and other http security measures.

Ensure Neo4j Browser does not cache login information

DESCRIPTION:

The Neo4j browser requires a username / password for login. This setting ensures that all clients using the Neo4j browser are required to login each time and that the login credentials are not maintained.

RATIONALE:

Enforcing a login each time prevents unauthorized access to the Neo4j database and provides traceability of database activities to a specific user or component.

REMEDIATION:

The browser login should be implemented before anyone accesses the Neo4j Server.

To disable the browser from retaining connection credentials:

Configure the **browser.retain_connection_credentials** configuration parameter in the **neo4j.conf** file:

```
browser.retain_connection_credentials=false
```

DEFAULT VALUE:

Not configured.

REFERENCES:

1. https://neo4j.com/docs/operations-manual/current/reference/configuration-settings/#config_browser.retain_connection_credentials

Implement Neo4j Browser Time-Out

DESCRIPTION:

The Neo4j browser requires a username / password for login. This setting ensures that a client using the Neo4j browser will be logged out after a period of inactivity.

RATIONALE:

Enforcing a time-out after a period of inactivity prevents unauthorized access to the Neo4j database.

REMEDIATION:

The browser time-out setting should be implemented before anyone accesses the Neo4j Server.

To enable the browser time-out:

In the **neo4j.conf** file, add the **browser.credential_timeout** configuration parameter and configure it to the appropriate setting:

```
browser.credential_timeout=5m
```

DEFAULT VALUE:

Not configured.

REFERENCES:

1. https://neo4j.com/docs/operations-manual/current/reference/configuration-settings/#config_browser.credential_timeout

Configure CORS Headers

DESCRIPTION:

The Access-Control-Allow-Origin response header indicates whether the response can be shared with resources with the given origin. For requests without credentials, the server may specify "*" as a wildcard, thereby allowing any origin to access the resource.

It may be necessary to specify the CORS response header to limit access to a specific URI.

RATIONALE:

Enabling a specific CORS response header limits the ability to access the website.

REMEDIATION:

To specify a specific CORS response header:

In the **neo4j.conf** file, add the **dbms.connectors.access_control_allow_origin** parameter and configure it to the appropriate setting. The expected value is a string:

```
dbms.security.http_access_control_allow_origin=http://neo4j.com
```

DEFAULT VALUE:

dbms.security.http_access_control_allow_origin=*

REFERENCES:

1. https://neo4j.com/docs/operations-manual/current/reference/configuration-settings/#config_dbms.security.http_access_control_allow_origin

Set HTTP Strict-Transport-Security (HSTS) Response Header

DESCRIPTION:

The HTTP Strict-Transport-Security (HSTS) header tells browsers that a webpage should only be accessed using HTTPS instead of HTTP. It is attached to every HTTPS response.

RATIONALE:

Enabling the HSTS Response Header ensures that the browser is only available via HTTPS.

REMEDIATION:

To enable the HSTS header:

In the **neo4j.conf** file, add the **dbms.security.http_strict_transport_security** parameter and configure it to the appropriate setting.

```
dbms.security.http_strict_transport_security=Strict-Transport-Security:
max-age=31536000; includeSubDomains
```

DEFAULT VALUE:

Not set.

REFERENCES:

1. https://neo4j.com/docs/operations-manual/current/reference/configuration-settings/#config_dbms.security.http_strict_transport_security
2. https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Strict-Transport-Security
3. https://www.owasp.org/index.php/HTTP_Strict_Transport_Security_Cheat_Sheet

Data Encryption

Ensure TLS protects all network communications (cluster, browser and bolt)

DESCRIPTION:

Use TLSv1.2 to protect all incoming and outgoing connections. This should include using TLS to encrypt communication between a Neo4j cluster as well as between all applications and Neo4j.

RATIONALE:

This prevents sniffing of cleartext traffic between Neo4j servers or performing a man-in-the-middle attack of Neo4j.

AUDIT:

To verify that system activity is being audited for Neo4j, inspect the **neo4j.conf** file for the following settings:

```
nmap --script ssl-enum-ciphers -p 5000 localhost
nmap --script ssl-enum-ciphers -p 6000 localhost
nmap --script ssl-enum-ciphers -p 7000 localhost
nmap --script ssl-enum-ciphers -p 7473 localhost
```

REMEDIATION:

Configure Neo4j servers to require the use of SSL or TLS to encrypt all Neo4j network communications.

To implement SSL or TLS to encrypt all Neo4j network communication, review the referenced section of our manual.

1. Read the following 2 sections in the Operations Manual: https://neo4j.com/docs/operations-manual/current/security/ssl-framework/ and then read: https://neo4j.com/docs/operations-manual/current/clustering/intra-cluster-encryption
2. Run **nmap --script ssl-enum-ciphers localhost** to determine what ciphers are available.
3. Configure intermediate certificates and server certificates per https://neo4j.com/docs/operations-manual/current/clustering/intra-cluster-encryption/
4. Configure **neo4j.conf** as follows:

```
dbms.connector.bolt.tls_level=REQUIRED

# Bolt SSL configuration
dbms.ssl.policy.bolt.enabled=true
dbms.ssl.policy.bolt.base_directory=certificates/default
dbms.ssl.policy.bolt.private_key=neo4jCert.key
dbms.ssl.policy.bolt.public_certificate=neo4jCert.crt
dbms.ssl.policy.bolt.ciphers=TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
```

```
dbms.ssl.policy.bolt.client_auth=NONE
dbms.ssl.policy.bolt.trusted_dir=certificates/bolt/trusted
dbms.ssl.policy.bolt.tls_versions=TLSv1.2

# Https SSL configuration
dbms.ssl.policy.https.enabled=true
dbms.ssl.policy.https.base_directory=certificates/default
dbms.ssl.policy.https.private_key=neo4jCert.key
dbms.ssl.policy.https.public_certificate=neo4jCert.crt
dbms.ssl.policy.https.trusted_dir=certificates/https/trusted
dbms.ssl.policy.https.ciphers=TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
dbms.ssl.policy.https.client_auth=NONE
dbms.ssl.policy.https.tls_versions=TLSv1.2

# Cluster SSL configuration
dbms.ssl.policy.cluster.enabled=true
dbms.ssl.policy.cluster.base_directory=certificates/default
dbms.ssl.policy.cluster.private_key=neo4jCert.key
dbms.ssl.policy.cluster.public_certificate=neo4jCert.crt
dbms.ssl.policy.cluster.ciphers=TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
dbms.ssl.policy.cluster.trusted_dir=certificates/bolt/trusted
dbms.ssl.policy.cluster.tls_versions=TLSv1.2
```

DEFAULT VALUE:

Not configured.

REFERENCES:

1. https://neo4j.com/docs/operations-manual/current/security/ssl-framework/
2. https://neo4j.com/docs/operations-manual/4.3/clustering/intra-cluster-encryption/
3. https://neo4j.com/docs/java-manual/current/client-applications/#java-driver-connection-uris

NOTES:

Neo4j drivers enable TLSv1.2 by default. Ciphers may be listed but are not required.

Encrypt Data Backups

DESCRIPTION:

Use TLSv1.2 to protect all data backups.

RATIONALE:

This prevents sniffing of cleartext traffic when performing a Neo4j backup.

AUDIT:

This prevents sniffing of cleartext traffic when performing a Neo4j backup:

```
nmap --script ssl-enum-ciphers -p 6362 localhost
```

NOTES:

Manage access to database dumps and to database backups.

The default backup port is different between a standalone instance and a causal cluster deployment.

REMEDIATION:

Configure Neo4j servers to require the use of TLS to encrypt all Neo4j network communications.

To implement TLS to encrypt all Neo4j network communication, review the referenced section of our manual.

1. Read the following 2 sections in the Operations Manual: https://neo4j.com/docs/operations-manual/current/security/ssl-framework/ and then read: https://neo4j.com/docs/operations-manual/current/clustering/intra-cluster-encryption/
2. Run **nmap --script ssl-enum-ciphers localhost** to determine what ciphers are available.
    a. Ciphers supported by the Oracle JRE can be found here: https://docs.oracle.com/en/java/javase/11/docs/specs/security/standard-names.html#jsse-cipher-suite-names
3. Configure intermediate certificates and server certificates per https://neo4j.com/docs/operations-manual/current/clustering/intra-cluster-encryption/
4. Enable encryption on the backup service for Causal Clustering instances.
5. Configure **neo4j.conf** Backup SSL Policy:

```
# Backup SSL configuration
dbms.ssl.policy.backup.enabled=true
dbms.ssl.policy.backup.base_directory=certificates/backup
dbms.ssl.policy.backup.private_key=private.key
dbms.ssl.policy.backup.public_certificate=public.crt
```

DEFAULT VALUE:

Not configured.

REFERENCES:

1. https://neo4j.com/docs/operations-manual/current/security/ssl-framework/
2. https://neo4j.com/docs/operations-manual/current/clustering/intra-cluster-encryption/
3. https://neo4j.com/docs/operations-manual/current/backup-restore/online-backup/#online-backup-ssl

Change default backup port

DESCRIPTION:

Modify backup port from the default setting of 6362.

RATIONALE:

This prevents access to the backup by sniffing for servers with ports 6362.

REMEDIATION:

Configure **neo4j.conf** to use a different backup port:

```
dbms.backup.address=0.0.0.0:6362
```

DEFAULT VALUE:

0.0.0.0:6362

REFERENCES:

1. https://neo4j.com/docs/operations-manual/current/configuration/ports

Use Volume Encryption

DESCRIPTION:

Use industry-standard methods for encrypting data when stored in disk ("data at rest").

REMEDIATION:

In Ubuntu, full-disk encryption is an option during the OS installation process. For other Linux distributions, the disk can be encrypted with dm-encrypt. For Windows, use volume encryption such as BitLocker.

For Cloud deployments (AWS, GCP, Azure), configure your storage so that the data is encrypted at rest using the Advanced Encryption Standard (AES).

REFERENCES:

1. https://neo4j.com/docs/operations-manual/current/security/checklist

Activity Logging

This section contains recommendations related to configuring activity logging in Neo4j.

Ensure that system activity is audited

DESCRIPTION:

Track access to Neo4j. Neo4j Enterprise includes a system auditing facility that can record system events (e.g.m user operations, connection events) on a Neo4j instance. These audit records permit forensic analysis and allow administrators to verify proper controls.

RATIONALE:

System level logs can be handy while troubleshooting an operational problem or handling a security incident.

AUDIT:

To verify that system activity is being audited for Neo4j, inspect the **neo4j.conf** file for the following settings:

```
dbms.directories.logs=logs

# Log level for the security log. One of DEBUG, INFO, WARN and ERROR.
#dbms.logs.security.level=INFO

# Threshold for rotation of the security log.
#dbms.logs.security.rotation.size=20m

# Minimum time interval after last rotation of the security log before it may
be rotated again.
#dbms.logs.security.rotation.delay=300s

# Maximum number of history files for the security log.
#dbms.logs.security.rotation.keep_number=7

dbms.logs.security.format=json
```

REMEDIATION:

1. Set the value of **dbms.directories.logs.destination** to the appropriate destination.
2. Set the value of **dbms.logs.security.level** to the appropriate setting.
3. Set the value of **dbms.logs.security.rotation.keep_number** to the appropriate number of logs to keep.
4. Set the value of **dbms.logs.security.format** to json.

DEFAULT VALUE:

- dbms.directories.logs=logs
- dbms.logs.security.level=INFO

- dbms.logs.security.rotation.size=20m
- dbms.logs.security.rotation.delay=300s
- dbms.logs.security.rotation.keep_number=7
- dbms.logs.security.format=plain

REFERENCES:

1. https://neo4j.com/docs/operations-manual/current/configuration/file-locations/
2. https://neo4j.com/docs/operations-manual/current/monitoring/logging/#security-events-logging
3. https://neo4j.com/docs/operations-manual/current/monitoring/logging/#secutiry-log-config

Operating System Hardening

This section contains recommendations related to hardening the operating system running below Neo4j.

Neo4j Database Running with Least Privileges

DESCRIPTION:

These recommendations help ensure that the Neo4j service runs as a least privileged user.

RATIONALE:

Anyone who has been a victim of viruses, worms, and other malicious software (malware) will appreciate the security principle of “least privilege”. If all processes ran with the smallest set of privileges needed to perform the user’s tasks, it would be more difficult for malicious and annoying software to infect a machine and propagate to other machines.

REMEDIATION:

Create a user that is only used for installing and running neo4j and directly related processes.

This user must not have administrative rights to the system.

The user that Neo4j runs as must have the following permissions:

Read Only

- bin
- certificates
- conf
- import
- lib
- licenses
- plugins

Read and write

- data
- logs
- metrics
- run

Execute

- all files in bin

REFERENCES:

1. https://neo4j.com/docs/operations-manual/current/configuration/file-locations/#file-locations-permissions

Ensure that Neo4j uses non-default ports

DESCRIPTION:

Changing the ports used by Neo4j makes it harder for attackers to find the database and target it.

RATIONALE:

Standard ports are used in automated attacks and by attackers to verify which applications are running on a server.

REMEDIATION:

Change the default ports used by the Neo4j server.

DEFAULT VALUE:

- dbms.backup.listen_address=0.0.0.0:6362
- dbms.connector.bolt.listen_address=:7687
- dbms.connector.http.listen_address=:7474
- dbms.connector.https.listen_address=:7473
- dbms.routing.listen_address=:7688
- causal_clustering.discovery_listen_address=:5000
- causal_clustering.transaction_listen_address=:6000
- causal_clustering.raft_listen_address=:7000

AUDIT:

Review the **neo4j.conf** file for:

- dbms.backup.listen_address
- dbms.connector.bolt.listen_address
- dbms.connector.http.listen_address
- dbms.connector.https.listen_address
- dbms.routing.listen_address
- dbms.routing.advertised_address
- causal_clustering.discovery_listen_address
- causal_clustering.discovery_advertised_address
- causal_clustering.transaction_listen_address
- causal_clustering.transaction_advertised_address
- causal_clustering.raft_listen_address
- causal_clustering.raft_advertised_address

IMPACT:

Hackers frequently scan IP addresses for commonly used ports, so it is not uncommon to use a different port to "fly under the radar". This is just to avoid detection, other than that there is no added safety by using a different port.

REFERENCES:

1. https://neo4j.com/docs/operations-manual/current/configuration/ports/

Ensure that operating system resource limits are set for Neo4j

DESCRIPTION:

Operating systems provide ways to limit and control the usage of system resources such as threads, files, and network connections on a per-process and per-user basis.

RATIONALE:

These limits prevent a single user from consuming too many system resources.

AUDIT:

To verify the resource limits set for Neo4j, run the following commands.

Extract the process ID for Neo4j:

```
ps -ef | grep neo4j
```

View the ulimits associated with that process number:

```
cat /proc/1322/limits
```

REMEDIATION:

Every deployment may have unique requirements and settings. Recommended thresholds and settings are particularly important for Neo4j deployments:

- f (file size): unlimited
- t (cpu time): unlimited
- v (virtual memory): unlimited[1]
- n (open files): 40000
- m (memory size): unlimited[1 2]
- u (processes/threads): 64000

DEFAULT VALUE:

Not configured.

REFERENCES:

1. https://neo4j.com/developer/guide-performance-tuning/
2. https://neo4j.com/docs/operations-manual/current/installation/linux/tarball/#linux-open-files

Disable Deprecated Neo4j Utilities

This section contains recommendations related to disabling deprecated utilities.

JMX

DESCRIPTION:

This setting ensures that remote JMX monitoring is disabled.

RATIONALE:

JMX has several drawbacks:

- It is based on the obsolete RMI protocol
- It can trigger harmful functions like garbage collection
- It can be used for nasty exploits like invoking arbitrary code (https://issues.apache.org/jira/browse/COLLECTIONS-580)

REMEDIATION:

JMX is not enabled by default. Neo4j provides other monitoring options which are described in our operations manual (https://neo4j.com/docs/operations-manual/current/monitoring/):

```
# Remote JMX monitoring, uncomment and adjust the following lines as needed.
Absolute paths to jmx.access and
# jmx.password files are required.
# Also make sure to update the jmx.access and jmx.password files with
appropriate permission roles and passwords, # the shipped configuration
contains only a read only role called 'monitor' with password 'Neo4j'.
# For more details, see:
http://download.oracle.com/javase/8/docs/technotes/guides/management/agent.html
# On Unix based systems the jmx.password file needs to be owned by the user
that will run the server,
# and have permissions set to 0600.
# For details on setting these file permissions on Windows see:
#
http://docs.oracle.com/javase/8/docs/technotes/guides/management/security-windows.html
#dbms.jvm.additional=-Dcom.sun.management.jmxremote.port=3637
#dbms.jvm.additional=-Dcom.sun.management.jmxremote.authenticate=true
#dbms.jvm.additional=-Dcom.sun.management.jmxremote.ssl=false
#dbms.jvm.additional=-Dcom.sun.management.jmxremote.password.file=/absolute/path/to/conf/jmx.password
#dbms.jvm.additional=-Dcom.sun.management.jmxremote.access.file=/absolute/path/to/conf/jmx.access
```

DEFAULT VALUE:

Not configured.

REFERENCES:

1. https://neo4j.com/docs/operations-manual/current/monitoring

Docker

This section contains recommendations related to running Neo4j securely in Docker.

Encryption

DESCRIPTION:

This section describes security in Neo4j when running in a Docker container.

RATIONALE:

The Docker image can be configured to expose Neo4j's native TLS support.

REMEDIATION:

Docker must be run as a user with read access to the SSL files. You must also publish port 7473 to access the HTTPS endpoint. Refer to the detailed instructions listed in the Neo4j Operations Manual (https://neo4j.com/docs/operations-manual/current/docker/security/) section on deploying Docker with SSL.

```
docker run \ --publish=7473:7473 --publish=7474:7474 --publish=7687:7687
--user="$(id -u):$(id -g)" -volume=$HOME/neo4j/certificates:/ssl
--volume=$HOME/neo4j/conf:/conf neo4j:4.3
```

DEFAULT VALUE:

Not enabled.

REFERENCES:

1. https://neo4j.com/docs/operations-manual/current/docker/security/
2. https://neo4j.com/docs/operations-manual/current/docker/security/#ssl-envar_example

Sandbox Extensions

This section describes how to ensure the security of custom-written additions in Neo4j.

SANDBOXING

DESCRIPTION:

Neo4j provides sandboxing to ensure that procedures do not inadvertently use insecure APIs.

For example, when writing custom code, it is possible to access Neo4j APIs that are not publicly supported, and these internal APIs are subject to change, without notice. Additionally, their use comes with the risk of performing insecure actions. The sandboxing functionality limits the use of extensions to publicly supported APIs, which exclusively contain safe operations, or contain security checks.

RATIONALE:

Neo4j can be extended by writing custom code which can be invoked directly from Cypher, as described in **Java Reference** → **Procedures and functions**. Custom code could perform insecure actions.

REMEDIATION:

The configuration setting **dbms.security.procedures.unrestricted** defines a comma-separated list of procedures and/or user- defined functions that are allowed full access to the database.

Configure this setting to allow the minimal set of procedures needed for the database:

```
dbms.security.procedures.unrestricted=apoc.*,my.extensions.example,my.procedures.*
```

DEFAULT VALUE:

Not enabled.

REFERENCES:

1. https://neo4j.com/docs/operations-manual/current/security/securing-extensions/

Enabling Extensions

DESCRIPTION:

We can configure Neo4j to only load a few extensions from a larger library.

RATIONALE:

Neo4j can be extended by writing custom code which can be invoked directly from Cypher, as described in **Java Reference → Procedures and functions**. Custom code could perform insecure actions. Allow listing manages the custom code that can be invoked directly from Cypher by allowing the loading of a subset of extensions from a larger library

REMEDIATION:

The configuration setting **dbms.security.procedures.allowlist** is used to name certain procedures that should be available from a library. It defines a comma-separated list of procedures that are to be loaded. Configure this setting to allow the minimal set of procedures or extensions needed for the database:

```
dbms.security.procedures.allowlist=apoc.coll.*,apoc.load.*
```

DEFAULT VALUE:

The default of the setting is *. This means that if you do not explicitly give it a value (or no value), all libraries in the plugins directory will be loaded.

REFERENCES:

1. https://neo4j.com/docs/operations-manual/current/security/securing-extensions/
2. https://neo4j.com/docs/operations-manual/current/security/securing-extensions/#allow-listing

Role Based Access Control Authorization

This section describes implementing Role Based Access Control (RBAC) Authorization in Neo4j 4.3.

RBAC

DESCRIPTION:

Neo4j has a complex security model stored in the system graph, maintained in a special database called the system database. All administrative commands need to be executing against the system database.

Neo4j 3.1 introduced the concept of role-based access control. It was possible to create users and assign them to roles to control whether the users could read, write and administer the database.

In Neo4j 4.0 this model was enhanced significantly with the addition of privileges which are the underlying access-control rules by which the user's rights are defined. The original built-in roles still exist with almost the exact same access rights, but they are no-longer statically defined. Instead, they are defined in terms of their underlying privileges, and they can be modified by adding and removing these access rights. In addition, any new roles created can by assigned any combination of privileges to create the specific access control desired. A major additional capability is sub-graph access control whereby read-access to the graph can be limited to specific combinations of label, relationship-type and property.

RATIONALE:

RBAC can grant or deny access to portions of the graph ensuring users only see the elements that they are entitled to.

REMEDIATION:

Design and implement RBAC according to your data security requirements. Refer to the Neo4j manual for examples.

REFERENCES:

1. https://neo4j.com/docs/cypher-manual/current/access-control/
2. https://neo4j.com/docs/operations-manual/current/authentication-authorization/

System Database

This section outlines specific information related to the System Database.

RBAC

DESCRIPTION:

All Neo4j servers will contain a built-in database called **system** which behaves differently than all other databases. In particular, when connected to this database you can only perform a specific set of administrative functions, as described in detail in the section on administration (https://neo4j.com/docs/cypher-manual/current/databases/). Most of the available administrative commands can only be run by users with specific administrative privileges. Information on configuring the security privileges is described in the Operations Manual (https://neo4j.com/docs/operations-manual/current/) as well as in the section on security commands (https://neo4j.com/docs/operations-manual/current/authentication-authorization/).

REMEDIATION:

Configure and limit access to the System Database to those users with specific administrative privileges.

REFERENCES:

1. https://neo4j.com/docs/cypher-manual/current/databases/
2. https://neo4j.com/docs/operations-manual/current/security/
3. https://neo4j.com/docs/operations-manual/current/
4. https://neo4j.com/docs/cypher-manual/current/access-control/

FIPS 140.2 Support

This section outlines specific information related to FIPS 140.2 support.

FIPS 140.2 Configuration

DESCRIPTION:

The National Institute of Standards and Technology (NIST) developed the Federal Information Processing Standard (FIPS) Publication 140-2 as a security standard that sets forth requirements for cryptographic modules, including hardware, software, and/or firmware, for U.S. federal agencies. FIPS 140-2 Validated certification was established to aid in the protection of digitally stored unclassified, yet sensitive, information.

Neo4j can be configured to use Federal Information Processing Standard 140-2 (FIPS 140-2) cryptography for data transmitted across the network. This can be achieved by the use of a NIST-certified third-party OpenSSL cryptographic module and by using this module as the only provider of cryptographic services.

REMEDIATION:

1. Set SSL Policies on Bolt and HTTPS. See Section 5.1.
2. Installing a FIPS-certified Crypto Module.
    a. Configure **neo4.conf** to add:

```
dbms.netty.ssl.provider=OPENSSL
```

    b. Download https://search.maven.org/artifact/io.netty/netty-tcnative-boringssl-static/2.0.44.Final/jarand copy the appropriate jar to your Neo4j plugins folder.
    c. Restart Neo4j.

DEFAULT VALUE:

FIPS 140.2 is not enabled by default.

REFERENCES:

1. https://neo4j.com/docs/operations-manual/current/security/ssl-framework/#ssl-providers

Deprecated Settings

This section describes breaking changes to the Neo4j surface when migrating from Neo4j 3.5 to Neo4j 4.0 and beyond.

Deprecations

v3.x	v4.x
The configuration settings dbms.security.property_level.enabled and dbms.security.property_level.blacklist are used to disallow properties.	These configuration settings have been discontinued. The blocking functionality is replaced by the Cypher DENY command. Note that the DENY command must be applied while Neo4j is running. For details, see **Cypher Manual → Security → Graph and sub-graph access control**.
dbms.security.auth_provider	This setting is replaced by two new settings: **dbms.security.authentication_providers** and **dbms.security.authorization_providers**.
dbms.connector.https.enabled is set to true by default.	This setting is no longer true by default. To enable Neo4j to listen for incoming connections on the HTTPS port, you have to configure this setting to true.
The different communication channels are secured independently from each other, using the following configuration settings: bolt.ssl_policy=<policy name> https.ssl_policy=<policy name> causal_clustering.ssl_policy=<policy name> dbms.backup.ssl_policy=<policy name>	These settings have been replaced by the setting dbms.ssl.policy.<scope>.enabled=true, where <scope> substitutes the communication channel (bolt, https, cluster, and backup).
SSL support for Bolt and HTTPS using the legacy SSL system. The dbms.directories.certificates setting is used to explicitly configure the directory that stores the private key and certificate files.	The legacy SSL system has been deprecated and the dbms.directories.certificates setting has been removed. It is recommended to use the standard SSL configuration.
dbms.ssl.policy.*.allow_key_generation	This setting has been removed. Neo4j no longer automatically generates a self-signed certificate.
dbms.security.procedures.whitelist	This setting has been renamed to dbms.security.procedures.allowlist