Morpheus Hardening Guide – 8.x

Last Updated: March 11, 2025

INTRODUCTION

The information within this document is intended for Morpheus Data customers who wish to maintain a hardened full-HA deployment. This guide does not cover 3-node HA deployments. This information is written for experienced information technology security engineers and administrators who are familiar with basic security practices, cloud technologies, and the Linux operating system.

DOCUMENT GUIDELINES

This document was created based on component vendors hardening recommendations and/or security best practices.

See Appendix A for references.

APPLICABLE VERSIONS

The information within this document assumes the following Morpheus and Morpheus Component versions.

- Morpheus version == 8.0.4
- Tomcat version >= 9.0.98
- NGiNX version >= 1.26.2
- RabbitMQ version >= 3.13.7
- Elasticsearch version >= 8.11.2

COMPONENT HARDENING

Application Tier

Tomcat Hardening

The Tomcat component of Morpheus hosts the Morpheus Application. The Tomcat application is controlled by the Morpheus application and installer. Currently, Morpheus Engineering is working to incorporate an increased security posture for Tomcat by incorporating the items outlined below. At the time of the writing of this guide, any changes to Tomcat configuration files will be overwritten by any number of actions including but not limited to:

- Morpheus application updates
- Executing the `morpheus-ctl reconfigure` command

Note: Morpheus installations do not contain the Tomcat Manager web application or any additional web applications.

Server.xml Configuration

Configure the `server.xml` located in `/opt/morpheus/lib/tomcat/conf`

Server

Set the following attributes within the `Server` element:

```
port="-1"
<Server port="-1" shutdown="SHUTDOWN">
```

Connector

Set the following attributes within the Connector element:

```
discardFacades="true"

<Connector address="127.0.0.1" port="8080"
  protocol="org.apache.coyote.http11.Http11NioProtocol"
  connectionTimeout="20000" bindOnInit="false"
  redirectPort="8443"
  discardFacades="true" />
```

Host

Set the following attributes within the `Host` element:

```
deployXML="false"

autoDeploy="false"

<Host name="localhost"  appBase="webapps"
    unpackWARs="true" autoDeploy="false" deployXML="false">
</Host>
```

Note: Setting `autoDeploy="false"` may cause issues when applying updates.

Context

Set the following attributes within the `Context` element:

```
path=""

privileged="false"

crossContext="true"

allowLinking="false"
```

```
<Context path="" crossContext="true" privileged="false" allowLinking="false"/>
```

Valve

Set the following attributes within the `Valve` element:

```
showServerInfo="false"
```

```
<Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"
    prefix="localhost_access_log" suffix=".txt"
    pattern="%h %l %u %t &amp;quot;%r&amp;quot; %s %b"
    showServerInfo="false"/>
```

Web.xml

Configure the `web.xml` located in `/opt/morpheus/lib/tomcat/conf`

```
<servlet>
    <servlet-name>default</servlet-name>
    <servlet-class>org.apache.catalina.servlets.DefaultServlet</servlet-class>
    <init-param>
        <param-name>debug</param-name>
        <param-value>0</param-value>
    </init-param>
    <init-param>
        <param-name>listings</param-name>
        <param-value>false</param-value>
    </init-param>
    <init-param>
        <param-name>showServerInfo</param-name>
        <param-value>false</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
</servlet>
```

NGINX Hardening

The NGINX component of Morpheus provides an SSL termination and cache proxy for our Tomcat container. Increasing the security posture of NGINX to industry best practices has been done out-of-the-box within Morpheus. Below are several settings natively configured by Morpheus that enable the NGINX application to operate more securely. Additionally, recommendations for SSL certificates are provided.

HTTPS Encryption

- Specify port 443 for SSL
- Add certificates for TLS
- Only allow stronger specific ciphers
- Use TLSv1.2
- Set Session Timeouts and Cache
- Secure Diffie-Hellman for TLS

```
server {
  listen 443 ssl;
  server_name <servername>;
  ssl_certificate /etc/morpheus/ssl/<CA_Server_Bundle>.pem;
  ssl_certificate_key /etc/morpheus/ssl/<server_key>.key;
  ssl_ciphers ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-
ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-
POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-
GCM-SHA384;
  ssl_protocols TLSv1.2;
  ssl_session_timeout 5m;
  ssl_session_cache builtin:1000  shared:SSL:10m;
  ssl_session_tickets off;
  ssl_dhparam /etc/morpheus/ssl/dhparams/dhparams.pem;
}
```

Information Leakage

In a default NGINX installation, sensitive information can be revealed by your installation, which can assist attackers. To secure NGINX within Morpheus, the `server_tokens off;` setting should be configured. The configuration files are located in `/opt/morpheus/embedded/nginx` and the configuration change must be made in both `morpheus.conf` and `morpheus-ssl.conf`

Chaining CA and SSL Certificate

Morpheus recommends that if customers are using certificates, an internal or 3rd party, that NGINX should be secured with a certificate bundle that contains the full chain of certificates, root-level CA, any intermediate CAs, and the server certificate. The order of certificates in the bundle should be as follows: ServerCertificate > IntermediateCA > RootCA. All certificates should utilize Base64 encoding and be in .pem format.

Messaging Tier

RabbitMQ Hardening

The RabbitMQ component of Morpheus provides an AMQP-based message queue within the application. The Morpheus Agent opens an outbound connection from the managed machine back to the Morpheus appliance allowing for a bidirectional command bus where the message queue is utilized to send messages to orchestrate the workload.

Increasing the security posture of RabbitMQ to industry best practices has been done out-of-the-box within Morpheus for single-server deployment however if deploying a distributed architecture the steps below should be taken in order to operate more securely.

CentOS and RedHat Linux Firewall

Local firewalls should only open ports for the appropriate RabbitMQ services being leveraged.

```
sudo firewall-cmd --zone=public --permanent --add-port=4369/tcp --add-
port=25672/tcp --add-port=5671-5672/tcp --add-port=15671-15672/tcp
sudo firewall-cmd --reload
```

SELinux

Configure SELinux with appropriate rules if SELinux is supported within the environment.

SELinux required ports and protocols for the RabbitMQ server:

```
semanage port --add 15671 -p tcp -t rabbitmq_port_t
semanage port --add 15672 -p tcp -t rabbitmq_port_t
semanage port --add 5671 -p tcp -t rabbitmq_port_t
semanage port --add 5672 -p tcp -t rabbitmq_port_t
semanage port --add 25672 -p tcp -t rabbitmq_port_t
semanage port --add 4369 -p tcp -t rabbitmq_port_t
```

Add new vHost

During the installation of RabbitMQ to support the Morpheus application, a new vHost should be created and the default vHost (/) should be deleted.

RabbtMQ create new vHost

```
sudo rabbitmqctl add_vhost morpheus
```

RabbitMQ delete default vHost

```
sudo rabbitmqctl delete_vhost /
```

Delete the "guest" account

RabbitMQ delete guest user

```
sudo rabbitmqctl delete_user guest
```

Create erLang cookie

The creation of the erLang cookie is be done during the installation process. The default algorithm for the creation of this file should NOT be used. The content should be randomly generated.

```
sudo sh -c "echo '7RSzaJe2y45S6JG' > /var/lib/rabbitmq/.erlang.cookie"
```

Enabling TLS

The use of TLS connections whenever possible to encrypt traffic is recommended

Prerequisites

- Certificate Authority available

Configure the CA template (Windows)

- Clone Template from Web Server
- Enable Signing and Encryption
- Add to CA available templates

Create a certificate request and private key

```
openssl req -new -newkey rsa:2048 -keyout rabbit-01.key -out rabbit-01.csr -nodes
```

Submit to CA and Import Completed Certificate to Server

- Submit the `rabbit-01.csr` file to your CA and return a DER formatted cert.
- Copy the completed certificate to the RabbitMQ server
- Rename the returned certificate to a filename with the `.pem` extension
- Examine the cert to ensure validity: `openssl x509 -noout -text -in /etc/pki/tls/certs/rabbit-01.pem`
- Change ownership of certificate and key files: `chown 0400 rabbitmq rabbit-01.pem; chown 0400 rabbitmq rabbit-01.key`

Move certificates

```
cp <certfile>.pem /etc/pki/tls/certs/
```

Move Keys

```
cp <keyfile>.pem /etc/pki/tls/certs/
```

RabbitMQ Server Configuration

Turning off RabbitMQ listeners that are not secured by TLS and the creation of idle timeouts, inactivity timeouts, and request timeouts should be configured in Rabbit.conf.

Note: The rabbit.conf does not exist, create it in `/etc/rabbitmq/`

```
ssl_options.cacertfile              = /etc/pki/tls/certs/$(cacert).pem
ssl_options.certfile                = /etc/pki/tls/certs/$(rabbit-node).pem
ssl_options.keyfile                 = /etc/pki/tls/private/$(rabbit-node).key
ssl_options.verify                  = verify_none
ssl_options.fail_if_no_peer_cert    = true
ssl_options.versions.1              = tlsv1.2
listeners.tcp                       = none
stomp.listeners.tcp                 = none
listeners.ssl.default               = 5671

#Timeout values in this section are in milliseconds
management.ssl.cacertfile           = /etc/pki/tls/certs/$(cacert).pem
```

```
management.ssl.certfile              = /etc/pki/tls/certs/$(rabbit-node).pem
management.ssl.keyfile               = /etc/pki/tls/private/$(rabbit-node).key
management.ssl.port                  = 15671
management.ssl.versions.1            = tlsv1.2
management.ssl.idle_timeout          = 120000
management.ssl.inactivity_timeout    = 120000
management.ssl.request_timeout       = 60000

#Timeout value in minutes below
management.login_session_timeout     = 5
```

Import Certificates

Import Certificates into Morpheus Java Keystore on all RabbitMQ Nodes.

```
sudo /opt/morpheus/embedded/java/bin/keytool -import -trustcacerts -file
<ca_certfile>.cer -alias <ca_alias> -keystore
/opt/morpheus/embedded/java/lib/security/cacerts
```

Import RabbitMQ node certificates.

```
sudo /opt/morpheus/embedded/java/bin/keytool -import -trustcacerts -file
<server_certfile>.pem -alias <server_alias> -keystore
/opt/morpheus/embedded/java/lib/security/cacerts
```

Repeat this process for all RabbitMQ node certificates

Edit Morpheus config file

Edit `/etc/morpheus.rb` to contain the following values

```
rabbitmq['enable'] = false
rabbitmq['use_tls'] = true
rabbitmq['vhost'] = '<morpheus_vhost_name>'
rabbitmq['queue_user'] = '<rabbitMQ_user>'
rabbitmq['queue_user_password'] = '<queueUserPassword>'
rabbitmq['host'] = 'x.x.x.x'
rabbitmq['port'] = '5671'
rabbitmq['stomp_port'] = '61614'
rabbitmq['heartbeat'] = 50
```

Run `morpheus-ctl reconfigure`

Upon completion, run

```
sudo morpheus-ctl restart morpheus-ui
```

Non-Transactional Database Tier

The Non-Transactional Database Tier consists of Elasticsearch (clustered, if redundancy is desired). The Elasticsearch component of Morpheus enables logs and metrics. Increasing the security posture of Elasticsearch to industry best practices has been done out-of-the-box within Morpheus for a single

server deployment however if deploying a distributed architecture, the steps below should be taken in order to operate more securely.

Elasticsearch Hardening

Enabling TLS for Inter-node Cluster communication and HTTP REST access

TLS requires X.509 certificates to perform encryption and authentication of the application that is being communicated with. In order for the communication between nodes to be truly secure, the certificates must be validated. The recommended approach for validating certificate authenticity in an Elasticsearch cluster is to trust the certificate authority (CA) that signed the certificate. By doing this, as nodes are added to your cluster they use a certificate signed by the same CA, and the node is automatically allowed to join the cluster. Additionally, it is recommended that the certificates contain subject alternative names (SAN) that correspond to the node's IP address and DNS name so that hostname verification can be performed.

If an internal CA is not available and utilization of an external CA is not desired, a CA can be created using the `elasticsearch-certutil` utility provided by ElasticSearch. By default, the output file from the utility is a PKCS#12 file, which contains the CA certificate and the private key for the CA. To be compliant with FIPS 140-2 compliance, export of the certificates will be in a PEM format. The `elasticsearch-certutil` is located in the `/usr/share/elasticsearch/bin` directory. Below is an example of the command that creates a CA bundle with a specified distinguished name for X number days expiration and the output file is in PEM format.

Note: If the use of elasticsearch-certutil is not desired, the certificates that are obtained must allow for both clientAuth and serverAuth if the extended key usage extension is present. The certificates need to be in `.pem` format for FIPS 140-2 compliance. Although not required, it is highly recommended that the certificate contains the DNS names and/or IP addresses of the node so that hostname verification can be used.

1. To start the process, create a YAML file with the hostnames and IP addresses of your ElasticSearch cluster nodes.

```
instances:
  - name: "hostname1"
    ip:
     - "192.168.1.10"
    dns:
     - "hostname1.morpheusdata.com"
  - name: "hostname2"
    ip:
     - "192.168.1.11"
    dns:
     -"hostname2.morpheusdata.com"
  - name: "hostname3"
    ip:
     - "192.168.1.12"
    dns:
```

```
-"hostname3.morpheusdata.com"
```

Note: The example YML file uses DNS names and IP addresses of the ElasticSearch cluster that will be used in the creation of the server certificate bundles for *Subject Alternative Names*, which is highly recommended.

2. Run the command below to create a CA certificate, server certificate and key bundle. The switches for the `elasticsearch-certutil` will use the YML file, created above, to output multiple server certificate bundles for use in the ElasticSearch cluster.

```
cd /usr/share/elasticsearch
bin/elasticsearch-certutil cert ca --pem --in </path/input_file.yml> --out
</path/file_path/bundle.zip
```

Note: The output .zip file will need to be extracted and the appropriate certificates and keys can be copied to the appropriate server's `/etc/elasticsearch/` directory.

3. The `elasticsearch.yml` file will need to be updated to allow the use of the certificate bundle to use TLS communication.

The example YML file below allows for TLS setup for internode cluster communication along with HTTP TLS using the same certificate bundle created above. Both settings are highly recommended.

```
# This security section allows for TLS enablement for internode cluster
communication with hostname verification enabled. This is highly recommended.
xpack.security.transport.ssl.enabled: true
xpack.security.transport.ssl.verification_mode: full
xpack.security.transport.ssl.key: </path/filename>.key
xpack.security.transport.ssl.certificate: </path/filename>.crt
xpack.security.transport.ssl.certificate_authorities: </path/filename>.crt
# This security section allows for TLS enablement on the HTTP REST interface for
ElasticSearch.
xpack.security.http.ssl.enabled: true
xpack.security.http.ssl.key: </path/filename>.key
xpack.security.http.ssl.certificate: </path/filename>.crt
xpack.security.http.ssl.certificate_authorities: </path/filename>.crt
```

4. Restart ElasticSearch: `sudo systemctl stop elasticsearch.service`

5. Set the passwords for the ALL built-in users

```
cd /usr/share/elasticsearch
bin/elasticsearch-setup-passwords interactive
```

Note: The command above will allow an *interactive* session to set up passwords on all default accounts. This is only allowed the first time you run the command. If you wish to change passwords again after this command is run, you'll need to use the GUI or API to change the default users password.

Configuring ElasticSearch for FIPS 140-2

Important Note: In order to run ElasticSearch with FIPS 140-2 compliance, the customer must purchase a Platinum level or above subscription from Elasticsearch.

The Federal Information Processing Standard (FIPS) Publication 140-2, (FIPS PUB 140-2), titled "Security Requirements for Cryptographic Modules" is a U.S. government computer security standard used to approve cryptographic modules. Elasticsearch offers a FIPS 140-2 compliant mode and as such can run in a FIPS 140-2 enabled JVM by configuring the following settings.

```
# FIPS 140-2 Enablement - with Platinum subscription or above from ElasticSearch
xpack.security.fips_mode.enabled: true
xpack.security.http.ssl.supported_protocols: TLSv1.2, TLSv1.3
xpack.security.ssl.diagnose.trust: true
xpack.security.authc.password_hashing.algorithm: pbkdf2
```

Configuring Morpheus for Hardened ElasticSearch

Settings within Morpheus will need to be configured for the consumption of the hardened ElasticSearch cluster. The `morpheus.rb` will need to be updated and the CA certificate and server certificate imported in the Morpheus keystore.

The `morpheus.rb` file will need the below settings. The `morpheus.rb` is located at `/etc/morpheus/`

```
elasticsearch['enable'] = false
elasticsearch['cluster'] = '<cluster_name>'
elasticsearch['es_hosts'] = {'<ip_address>' => 9200, '<ip_address>' => 9200,
'<ip_address>' => 9200}
elasticsearch['use_tls'] = true
elasticsearch['auth_user'] = '<elastic_user>'
elasticsearch['auth_password'] = '<password>'
```

Note: '<cluser_name>' is the name of the ElasticSearch cluster, '<es_host>' is the IP address of each node in the ElasticSearch cluster, '<auth_user>' is the elastic user who's password is being changed, and '<auth_password>' is the password for the *elastic* user.

Importing Certificates

Importing the CA and server certificates into the Java keystore is the final step.

```
sudo /opt/morpheus/embedded/java/bin/keytool -import -trustcacerts -file
<path/<filename>.crt -alias <alias_certname> -keystore
/opt/morpheus/embedded/java/lib/security/cacerts
```

Repeat the command for each server certificate in the ElasticSearch cluster and the CA certificate.

Reconfigure Morpheus

A Morpheus reconfigure must be run in order to read in the new ElasticSearch settings. Administrators may choose to perform this step at a later time: `sudo morpheus-ctl reconfigure`

Appendix A: References

- ElasticSearch Security: https://www.elastic.co/guide/en/elasticsearch/reference/current/certutil.html
- ElasticSearch for FIPS 140-2: https://www.elastic.co/guide/en/elasticsearch/reference/current/fips-140-compliance.html
- Morpheus Documentation: https://docs.morpheusdata.com/en/latest/getting_started/getting_started.html
- RabbitMQ TLS Support: https://www.rabbitmq.com/ssl.html
- RabbitMQ Production Checklist: https://www.rabbitmq.com/production-checklist.html#overview
- Tomcat Security: https://tomcat.apache.org/tomcat-9.0-doc/security-howto.html#Context