**SONARQUBE SSL CONFIGURATION**

**This an an archived version of the documentation for SonarQube version 5.3.**  
**See**[**https://docs.sonarqube.org/display/SONAR/Documentation**](https://docs.sonarqube.org/display/SONAR/Documentation)**for current functionality**

[Running SonarQube Over HTTPS](https://docs.sonarqube.org/display/SONARQUBE53/Running+SonarQube+Over+HTTPS)

1. [Dashboard](https://docs.sonarqube.org/collector/pages.action?key=SONARQUBE53&src=breadcrumbs-collector)
2. **…**
3. [Installing the Server](https://docs.sonarqube.org/display/SONARQUBE53/Installing+the+Server?src=breadcrumbs-parent)

[Skip to end of banner](https://docs.sonarqube.org/display/SONARQUBE53/Running+SonarQube+Over+HTTPS#page-banner-end)

[Go to start of banner](https://docs.sonarqube.org/display/SONARQUBE53/Running+SonarQube+Over+HTTPS#page-banner-start)

[Skip to end of metadata](https://docs.sonarqube.org/display/SONARQUBE53/Running+SonarQube+Over+HTTPS#page-metadata-end)

* Created by [Ann Campbell](https://docs.sonarqube.org/display/~ann.campbell.2) on [Mar 09, 2016](https://docs.sonarqube.org/pages/viewpreviousversions.action?pageId=6953633)

[Go to start of metadata](https://docs.sonarqube.org/display/SONARQUBE53/Running+SonarQube+Over+HTTPS#page-metadata-start)

There are two ways to run SonarQube over HTTPS:

* By building a standard reverse proxy infrastructure
* By configuring the SonarQube server that natively supports SSL (since version 4.0)

Building a Standard Reverse Proxy Infrastructure

**Recommended Setup**

Using a reverse proxy infrastructure is the recommended way to set up your SonarQube installation on production environments which need to be highly secured. This allows to fully master all the security parameters that you want.

The reverse proxy must be configured to set the value "X\_FORWARDED\_PROTO: https" in each HTTP request header.

Without this property, redirection initiated by the SonarQube server will fall back on HTTP.

Configuring the SonarQube Server that Natively Supports SSL

**Known Limitations**

 Know limitations are:

* Because Tomcat is embedded in SonarQube in a "black box" mode, only few Tomcat parameters can be customised - which might be an issue for some security configuration it is not possible to specify which ciphers are allowed and which are banned to encrypt the SSL connection. See [~~SONAR-6140~~](https://jira.sonarsource.com/browse/SONAR-6140) - Ability to restrict HTTPS ciphers **CLOSED**
* The [Report Plugin](http://docs.sonarqube.org/display/PLUG/Report+Plugin) only supports TLSv1 connections. See [REP-61](https://jira.sonarsource.com/browse/REP-61?src=confmacro) - Jira project doesn't exist or you don't have permission to view it.

The configuration is pretty standard. The information to access the certificate must be provided in the configuration of the web server.

Here are two examples: Generating a certificate / Reusing an existing certificate.

Generating a Certificate

**Generate a RSA certificate**

Run the following command:

|  |
| --- |
| keytool -genkey -alias sonartomcat -keyalg RSA |



It adds the certificate to *USER\_HOME/.keystore.*

**Configure the SonarQube server**

Open the *SONARQUBE\_HOME/conf/sonar.properties* file and update it as below:

**sonar.properties**

|  |
| --- |
| ...  # TCP port for incoming HTTP connections. Disabled when value is -1.  sonar.web.port=-1    # TCP port for incoming HTTPS connections. Disabled when value is -1 (default).  sonar.web.https.port=8999    # HTTPS - the alias used to for the server certificate in the keystore.  sonar.web.https.keyAlias=sonartomcat    # HTTPS - the password used to access the server certificate from the  sonar.web.https.keyPass=changeit    # HTTPS - the password used to access the specified keystore file. The default  # value is the value of sonar.web.https.keyPass.  sonar.web.https.keystorePass=changeit  ... |

Restart the web server. You should now only be able to access the SonarQube server over HTTPS on port 8999.

**Running an Analysis**

Update the settings of your analyzer (*SONARQUBE\_HOME/conf/sonar-runner.properties* for SonarQube Runner or *settings.xml* for Maven...): sonar.host.url=https://localhost:8999

Analyze one of your projects to check that it works fine.

Use an Existing Certificate

# Generate a PKCS12 file with existing certificates and CAFile

$ openssl pkcs12 -export -in myserver.cert -inkey myserver.key -out myserver.p12 -name myserver -CAfile myserver.cert -caname root -chain

The **name** attribute value will be used for sonar.web.https.keyAlias**.**

The **export password** you have entered should be specify for sonar.web.https.keyPass.

Then copy your *myserver.p12* file in a secure place (e.g. */opt/sonar/conf*) and configure SonarQube to use it:

The **export password** you have entered should be specify for sonar.web.https.keyPass.

Then copy your *myserver.p12* file in a secure place (e.g. */opt/sonar/conf*) and configure SonarQube to use it:

|  |
| --- |
| ...  # TCP port for incoming HTTP connections. Disabled when value is -1.  sonar.web.port=-1    # TCP port for incoming HTTPS connections. Disabled when value is -1 (default).  sonar.web.https.port=443    # HTTPS - the alias used to for the server certificate in the keystore.  # If not specified the first key read in the keystore is used.  sonar.web.https.keyAlias=myserver    # HTTPS - the password used to access the server certificate from the  # specified keystore file. The default value is "changeit".  sonar.web.https.keyPass=mykeypass    # HTTPS - the pathname of the keystore file where is stored the server certificate.  # By default, the pathname is the file ".keystore" in the user home.  # If keystoreType doesn't need a file use empty value.  sonar.web.https.keystoreFile=/opt/sonarqube/conf/myserver.p12    # HTTPS - the password used to access the specified keystore file. The default  # value is the value of sonar.web.https.keyPass.  #sonar.web.https.keystorePass=    # HTTPS - the type of keystore file to be used for the server certificate.  # The default value is JKS (Java KeyStore).  sonar.web.https.keystoreType=PKCS12  ... |

openssl pkcs12 -export -out sonarqube.p12 \

-passout 'pass:DevOpsDev@2020' -inkey devopsdev.statebanktimes.in.key \

-in devopsdev.statebanktimes.in.crt -certfile devopsdev.statebanktimes.in.crt -name sonarqube.devopssbi.com

keytool -importkeystore -srckeystore sonarqube.p12 \

-srcstorepass 'DevOpsDev@2020' -srcstoretype PKCS12 \

-srcalias sonarqube.devopssbi.com -deststoretype JKS \

-destkeystore sonarqube.jks -deststorepass 'DevOpsDev@2020' \

-destalias sonarqube.devopssbi.com

openssl pkcs12 -export -in myserver.cert -inkey myserver.key -out myserver.p12 -name myserver -CAfile myserver.cert -caname root –chain

in sonar.sh

JAVA\_HOME /usr/java/jdk-11.0.9

In wrapper.conf

JAVA\_HOME /usr/java/jdk-11.0.9

In sonar.properties

JAVA\_HOME /usr/java/jdk-11.0.9

In wrapper.conf folder put below entry so that your sonarqube is running on different java version

**wrapper.java.command=/usr/java/jdk-11.0.9/bin/java**

**Step1:Go to the below configuration file**

vim /etc/httpd/conf.d/ssl.conf

</VirtualHost>

<VirtualHost \*:9100>

SSLEngine on

SSLProtocol all -SSLv2 -SSLv3

SSLCipherSuite HIGH:3DES:!aNULL:!MD5:!SEED:!IDEA

ProxyPreserveHost On

ProxyPass / http://10.191.159.48:9101/

ProxyPassReverse / http://10.191.159.48:9101/

SSLCertificateFile /opt/ssl\_cert/29122020/devopsdev.statebanktimes.in.crt

SSLCertificateKeyFile /opt/ssl\_cert/29122020/devopsdev.statebanktimes.in.key

</VirtualHost>

**Step2: /opt/sonarqube-7.9.1/conf/sonar.properties**

#sonar.web.host=0.0.0.0

sonar.web.host=10.191.159.48

#sonar.web.host=0.0.0.0

sonar.web.port=9101

Install mod\_ssl

yum install mod\_ssl

yum.repo.d

install httpd

systemctl restart httpd