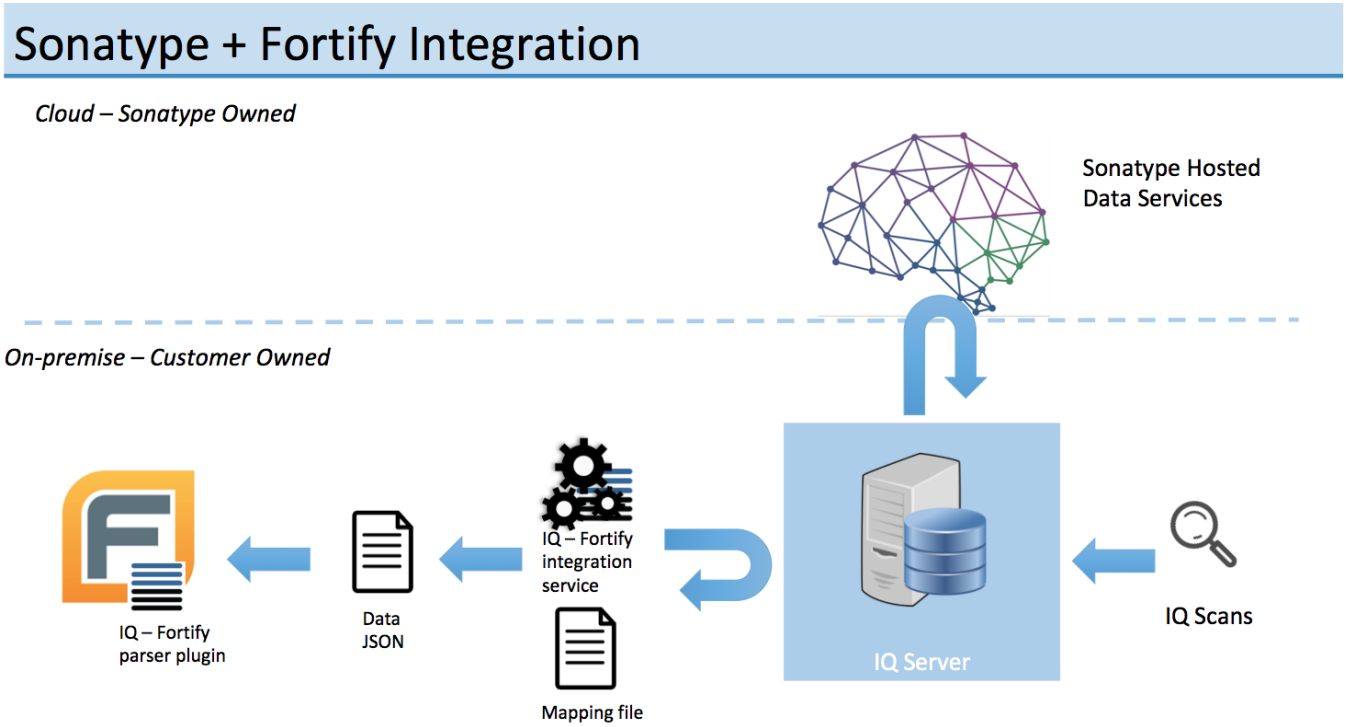
Installation Guide

Welcome to the Sonatype - Fortify Integration Service! This service has been tested against...

IQ Server 1.75 and above

Fortify SSC 19.10

Architecture View



This bundle is made of two parts:

A Parser Plugin for Fortify SSC and a stand-alone Integration Service that facilitates pulling data from the IQ Server and passing it to the SSC server.

The Integration Service has a built-in scheduler to control when it should run or can be configured to terminate after each run if you want to use your own scheduler. The service relies on Mappings file to which projects-phase in IQ map to which project-versions in SSC.

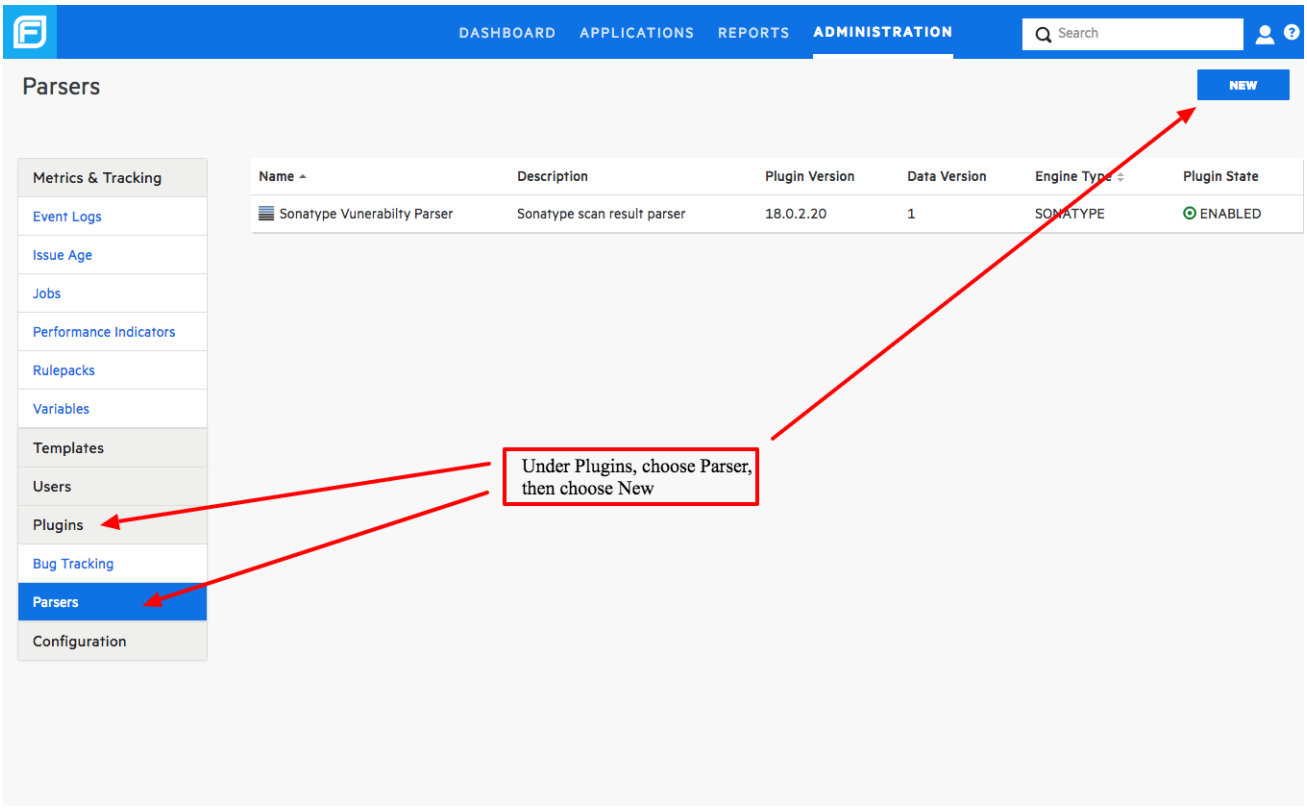
Integration service logic

1. Scheduler goes off or was manually started at /startScanLoad
2. Read Mapping file
3. Verify that the first entry has a report in IQ
    a. Get the evaluation date
    b. compare to the previous run if it exists
        i. IF no previous run, write out the vuln data file and upload to SSC
        ii. IF the evaluation date is not newer than the previous run, skip
        iii. IF the evaluation date is newer, overwrite the previous file and upload to SSC
    c. Uploads to SSC will create the project, or version, if they do not already exist in SSC
    d. Loop to next entry in Mapping file

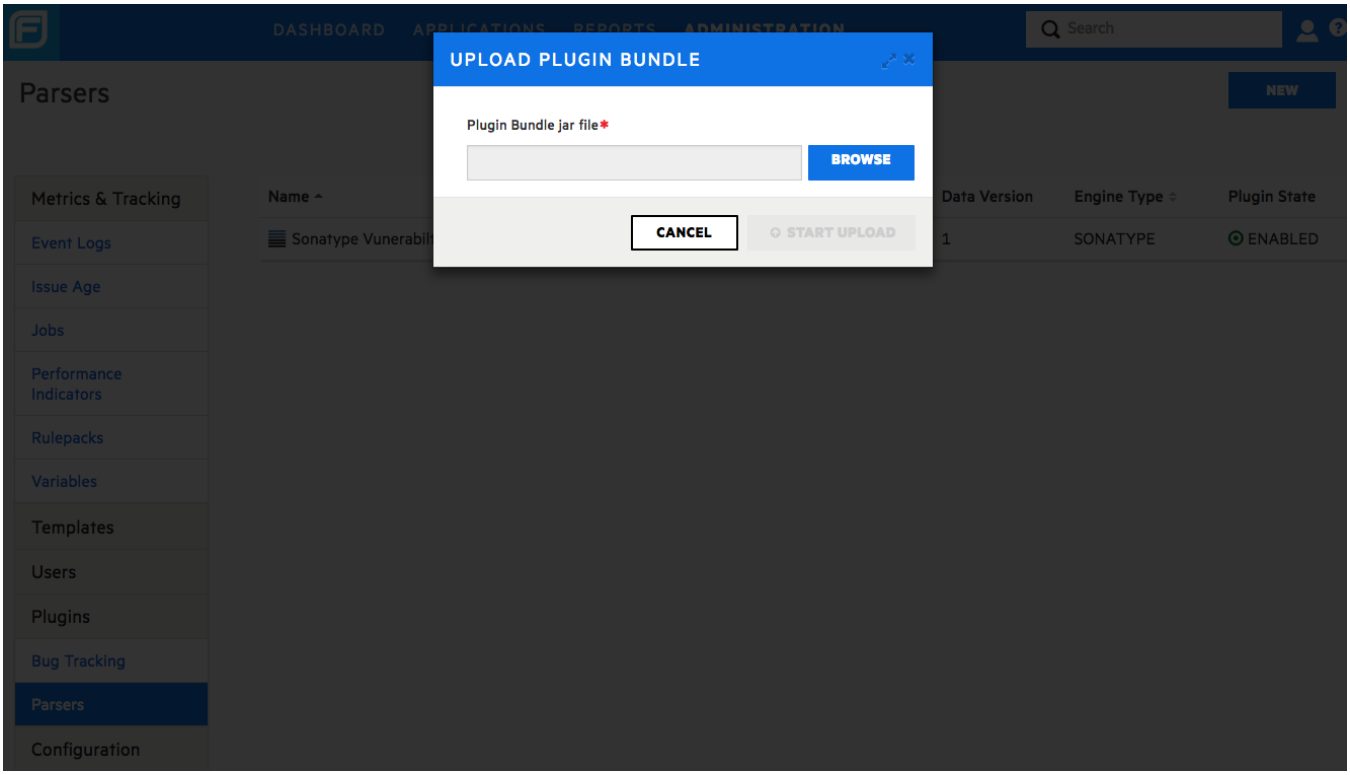
It should be added that vulnerabilities that have been triaged in IQ server (status set to Not applicable) will not be written into the data JSON file.

Installing the Parser Plugin

The plugin is installed through the administration configuration interface of Fortify SSC

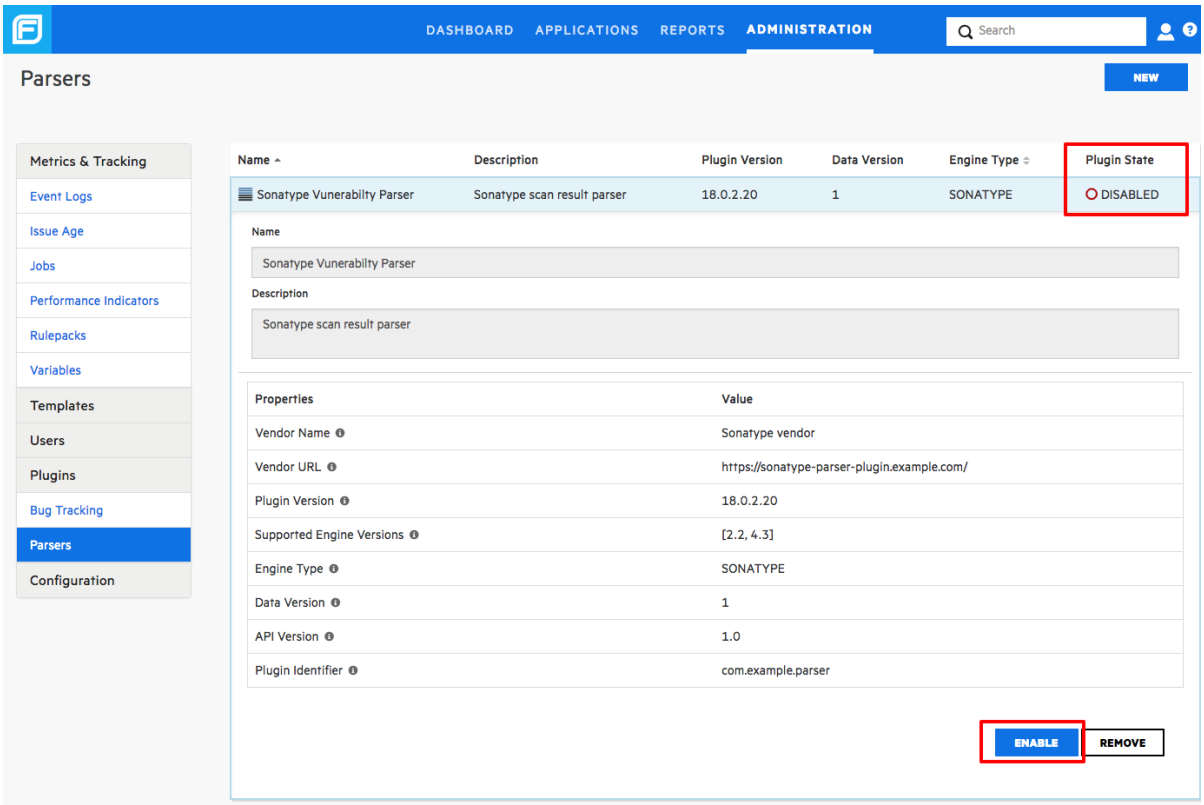


After accepting the warning about uploading Parsers you'll see a dialog to browse for a file:



Browse to the sonatype-plugin-*.jar file in this bundle and then 'Start Upload'

Once uploaded, be sure to 'Enable' the plugin.



And that's it for the Parser Plugin!

Configuring and Running the Integration Service

The Integration Service has a couple of properties files that you'll want to set prior to trying to run the service.

The iqapplication.properties file is where you put the URL's and user credentials both the IQ Server and the Fortify SSC application. The files are well documented and hopefully fairly self-evident we'll cover a few points here.

The service makes use of a 'work' directory, this is where it expects to write the JSON files that will be sent to SSC. These files also represent the 'previous run' and are used to determine if the data coming from the IQ Server is 'new'. These files also come in handy for troubleshooting or test data. The service also logs events which are defaulted to the Servicelog.log also in the 'work' directory.

```
# Default port to listen on, set as needed
server.port=8182

# URL and creds for IQ Server
iqserver.url=http://iq-server:8070/
iqserver.username=admin
iqserver.password=admin123

# URL and creds to SSC server
sscserver.url=http://ssc:8080/ssc/
sscserver.username=admin
sscserver.password=ch@ng3Me

# work directory where JSON files are stored
loadfile.location=./work/
mapping.file=mapping.json

# directory/file where log files are stored
logfile.location=./.work/Servicelog.log
logLevel=info

#cron expression; it consists of 7 fields
#<second> <minute> <hour> <day-of-month> <month> <day-of-week> <year>
#<year> field is optional. Rest all are required
#Some examples are as follows:
#Running every 12 hours starting at 6 - 0 0/720 6 * * ?
#Running every 12 hours starting at midnight - 0 0/720 0 * * ?
```

```
#Running every 6 hrs starting at 6 AM - 0 0/360 6 * * ?
#For more details please visit - https://www.baeldung.com/cron-expressions
# Currently scheduled to run at 6 AM and then every 6 hours.
scheduling.job.cron=0 0/360 6 * * ?

# Set it to true if wanted to close the process after next scheduled run or
# leave set to true if you want to use your own scheduler
# also http://localhost/killProcess will stop the process
KillProcess=false
```

The service has a built-in scheduler that understands cron-like expressions. We've defaulted it to run every 6 hrs.
You can also set the KillProcess flag to 'true' if you want to use your own scheduler as it will terminate the process after the run. Lastly, the process can be manually stopped via a REST endpoint.
The mapping.json file is used to connect applications at a given SDLC phose in the IQ server to an application version in SSC. If a project and sdlc phase defined in the mapping file doesn't exist in the IQ Server it will be skipped. If an application or version doesn't exist in SSC, it will be created and data transferred over. The sonatypeProject names are projectID's in IQ and not 'names'. Vulnerabilities with 'statuses' of 'Not Applicable' are not sent over to SSC.

```
[
        {
        "sonatypeProject": "petclinic",
        "sonatypeProjectStage": "release",
        "fortifyApplication": "PetClinic",
        "fortifyApplicationVersion": "1.0"
        },
        {
        "sonatypeProject": "petclinic",
        "sonatypeProjectStage": "build",
        "fortifyApplication": "PetClinic",
        "fortifyApplicationVersion": "2.0"
        },
        {
        "sonatypeProject": "struts-showcase",
        "sonatypeProjectStage": "stage-release",
        "fortifyApplication": "Struts Showcase",
        "fortifyApplicationVersion": "1.0"
        },
        {
        "sonatypeProject": "webgoat8",
        "sonatypeProjectStage": "release",
        "fortifyApplication": "Webgoat8",
        "fortifyApplicationVersion": "1.0"
        }
]
```

We'd recommend that you start with just a few to ensure everything is working. Then add more as needed. It is also possible to dynamically create a mapping.json file and then run the service in one-shot mode as part of a CI/CD pipeline.

Once the properties and mapping files are all set you can run the server using the Java -jar command as shown in the startup.sh file.

```
java -jar SonatypeFortifyIntegration-*.jar
```

That's it! Now your OSS vulnerability data is in alongside your other static and dynamic security testing data for a complete 360-degree view of your applications. More importantly, the OSS data comes from Sonatype's dedicated security research team so it is very accurate producing virtually no false positives. Sonatype also provides actionable recommendations to help arbitrate any dialog between security and development tohelp those two teams work together.

Enjoy!