ENSE 375 Group Project Step #4 Report

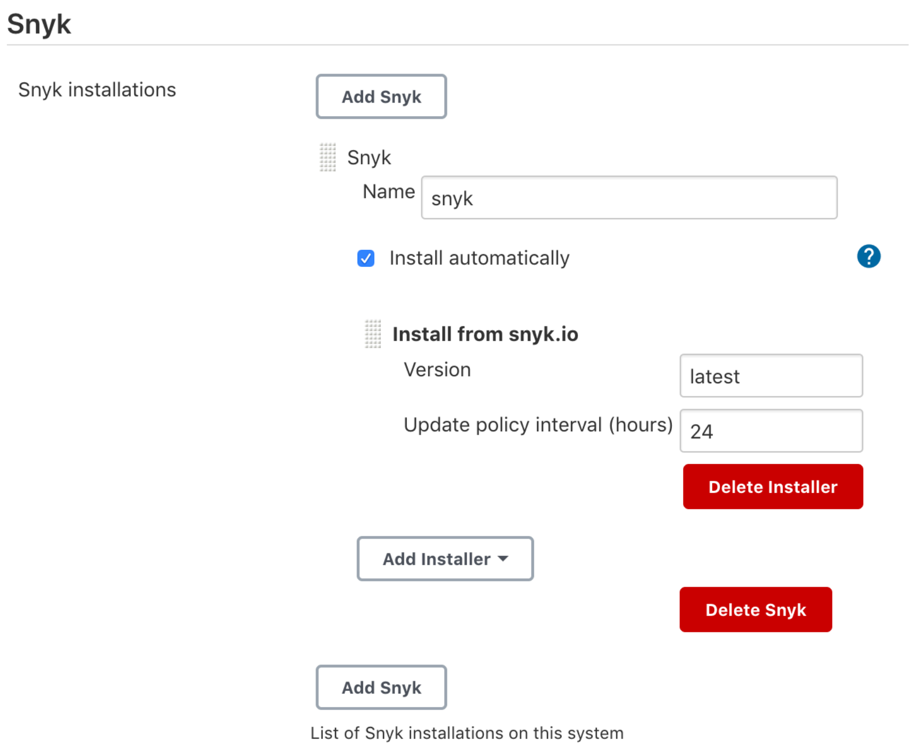
Carter Brezinski, Li Pan, Yash Patel, Abdelrahman Rabaa, Jacob Sauer

Security Plugin – Snyk

Snyk is an open-source security platform that is capable of detecting various types of issues in a code repository. Its hallmark feature is a dependency scanner, which detects and vulnerabilities in container images and open-source dependencies, and categorizes them based on severity. We chose to integrate Snyk into our project because of this dependency scanner, as our project repository now contains multiple dependencies, including JUnit, JMeter, and Selenium. At the time of integrating Snyk into the repository, however, only Selenium was properly configured; this is reflected in the screenshots depicting our repository's vulnerabilities.

The process of configuring Snyk in a Jenkins repository is outlined below:

1. On the Jenkins dashboard, navigate to *Configure Jenkins –> Manage Plugins*.
2. Install "Snyk Security Plugin" from the *Available* tab.
3. Return to the Jenkins dashboard and navigate to *Configure Jenkins –> Global Tool Configuration*.
4. Scroll to the "Snyk" subheading, and configure the installation settings so that they match the screenshot below (the *Name* value has no restrictions).


Snyk

Snyk installations

Snyk
Name: snyk

[✓] Install automatically

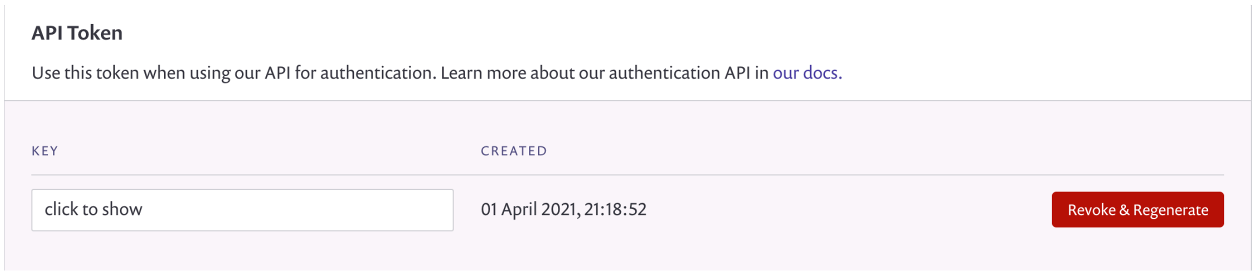
Install from snyk.io
Version: latest
Update policy interval (hours): 24

List of Snyk installations on this system

5. In the *pom.xml* file, create a new plugin section with the following attributes:

```
        </plugin>
          <plugin>
               <groupId>io.snyk</groupId>
               <artifactId>snyk-maven-plugin</artifactId>
               <version>2.0.0</version>
               <inherited>false</inherited>
               <executions>
                 <execution>
                   <id>snyk-test</id>
                   <goals>
                     <goal>test</goal>
                   </goals>
                 </execution>
                 <execution>
                   <id>snyk-monitor</id>
                   <goals>
                     <goal>monitor</goal>
                   </goals>
                 </execution>
               </executions>
               <configuration>
                 <apiToken>${env.SNYK_TOKEN}</apiToken>
                 <args>
                   <arg>--all-projects</arg>
                 </args>
               </configuration>
          </plugin>
```

6. Register for a Snyk account on https://snyk.io.
7. Navigate to *Account Settings*, and copy the API Token from the widget shown below:



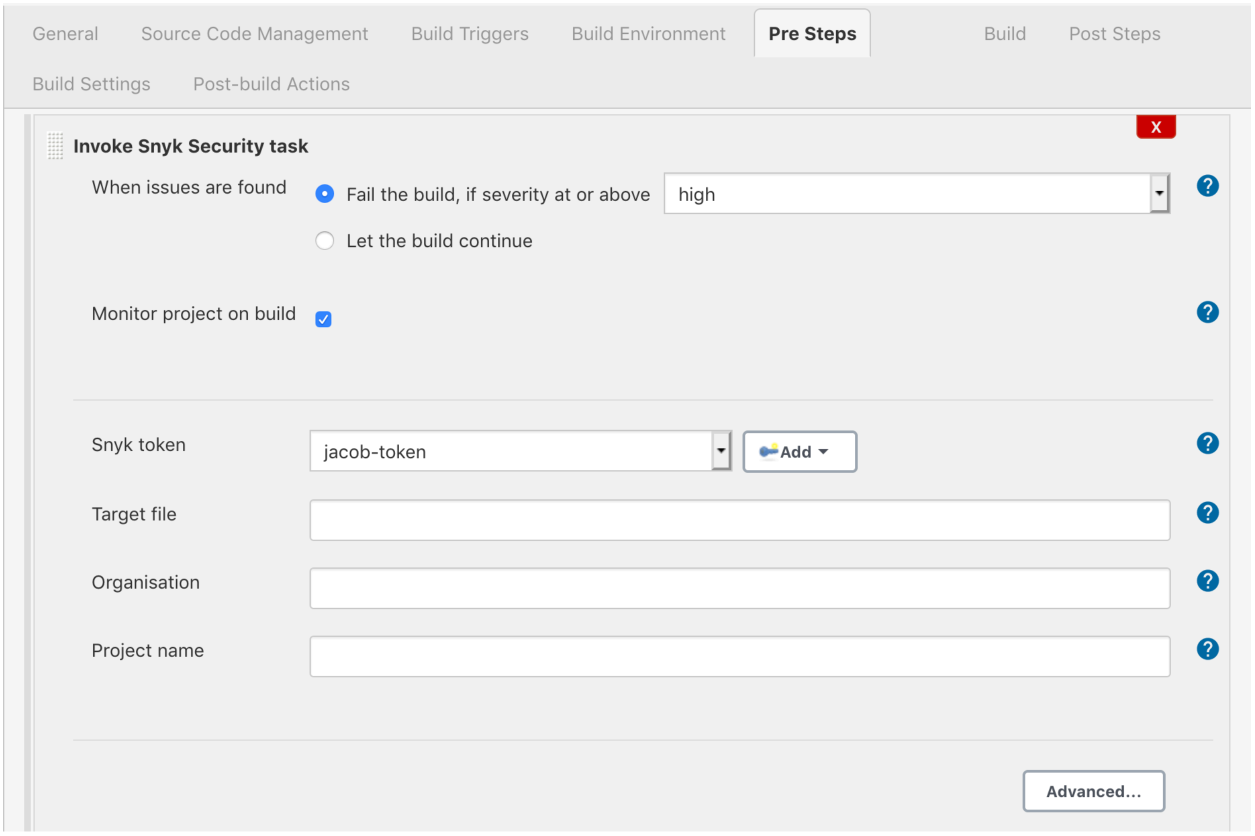
API Token

Use this token when using our API for authentication. Learn more about our authentication API in our docs.

KEY	CREATED	
click to show	01 April 2021, 21:18:52	Revoke & Regenerate

8. In Jenkins, navigate to *<<Project Name>> -> Configure.*
9. Scroll down to *Pre Steps*, and add a Snyk Security task. It is possible to force all project builds to fail if issues of any severity are detected, or continue a build even if

severe issues are detected. Our group chose to force builds to fail if issues of high severity are detected, and continue any builds with issues of moderate severity or lower. The screenshot below depicts the Snyk Security task of a group member's Jenkins instance, with the Snyk API token being nicknamed "jacob-token":



Invoke Snyk Security task

When issues are found: [✓ Fail the build, if severity at or above] high / Let the build continue

Monitor project on build: ✓

Snyk token: jacob-token

Target file:

Organisation:

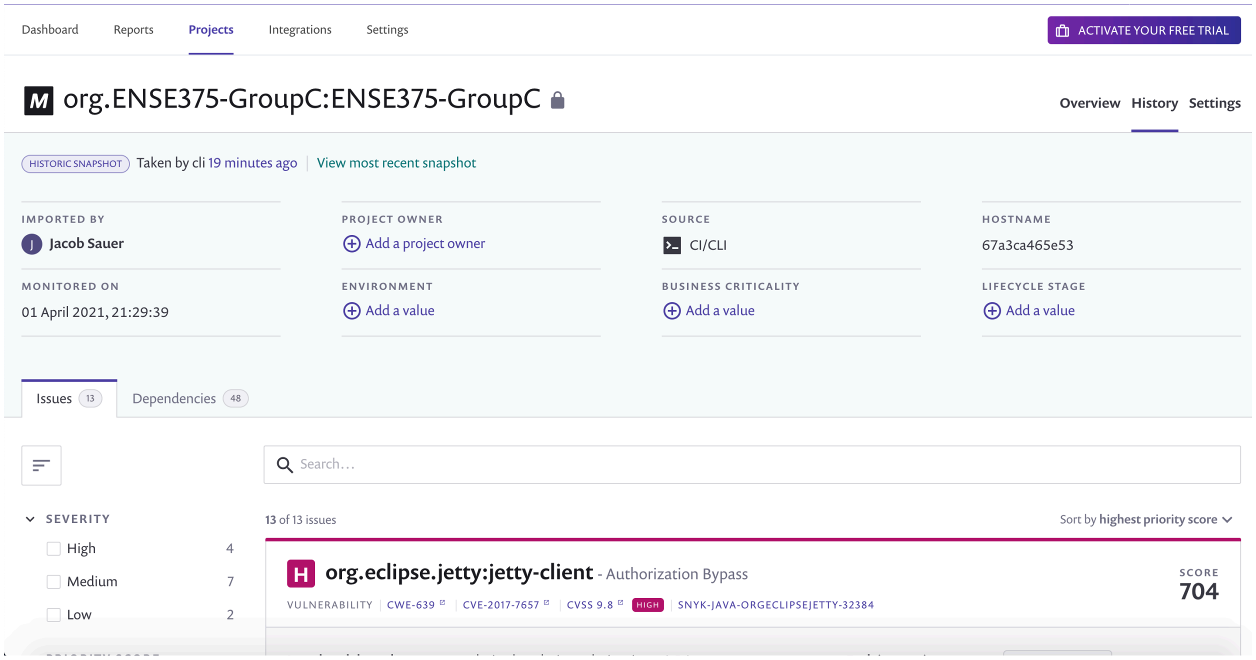
Project name:

At this point, it is possible to build a Jenkins project and observe the vulnerabilities detected by Snyk. After refactoring the *addPatient()* and *deletePatient()* functions in *App.java*, we tested our Risk Code Map repository from Step #3 for dependency-related security issues. Initially, Snyk detected 13 security issues in our repository; four were of high severity, seven were of medium severity, and the remaining two were of low severity. These issues included an authorization bypass, cache poisoning, information exposure, improper input validation, and denials of service. As a result of the four high-severity vulnerabilities, the Jenkins build was unsuccessful.

Fortunately, Snyk determined that 12 of the 13 vulnerabilities were easily fixable, as they had arisen due to the version of Selenium that was specified in *pom.xml*. Snyk's console window informed us that specifying version 3.141.0 of Selenium, rather than version 3.5.3, was the only necessary step in remedying these issues. Consequently, by changing just one line in *pom.xml*, our group was able to mitigate all security issues that Snyk deemed severe, and six of the seven issues that were deemed moderately severe.

Upon running a subsequent build of the Jenkins project, only one remaining vulnerability was detected; since this vulnerability was of moderate severity, the build succeeded.

The screenshot below depicts the 13 issues detected by Snyk with Selenium version 3.5.3 in our *pom.xml* file:



org.ENSE375-GroupC:ENSE375-GroupC

HISTORIC SNAPSHOT Taken by cli 19 minutes ago | View most recent snapshot

IMPORTED BY	PROJECT OWNER	SOURCE	HOSTNAME
Jacob Sauer	Add a project owner	CI/CLI	67a3ca465e53
MONITORED ON	ENVIRONMENT	BUSINESS CRITICALITY	LIFECYCLE STAGE
01 April 2021, 21:29:39	Add a value	Add a value	Add a value

Issues 13 | Dependencies 48

Search…

SEVERITY

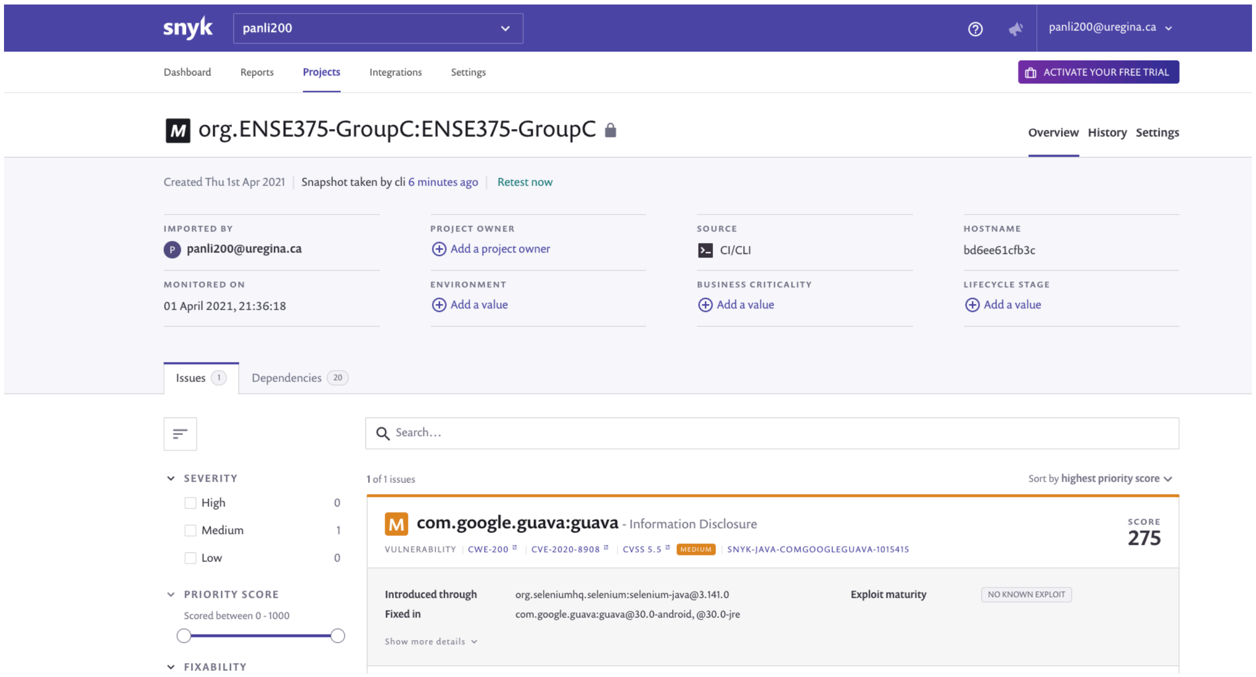
High	4
Medium	7
Low	2

13 of 13 issues — Sort by highest priority score

H org.eclipse.jetty:jetty-client - Authorization Bypass — SCORE 704

VULNERABILITY | CWE-639 | CVE-2017-7657 | CVSS 9.8 HIGH | SNYK-JAVA-ORGECLIPSEJETTY-32384

The screenshot below illustrates the 12 issues that were resolved by replacing version 3.5.3 with version 3.141.0:



org.ENSE375-GroupC:ENSE375-GroupC

Created Thu 1st Apr 2021 | Snapshot taken by cli 6 minutes ago | Retest now

IMPORTED BY	PROJECT OWNER	SOURCE	HOSTNAME
panli200@uregina.ca	Add a project owner	CI/CLI	bd6ee61cfb3c
MONITORED ON	ENVIRONMENT	BUSINESS CRITICALITY	LIFECYCLE STAGE
01 April 2021, 21:36:18	Add a value	Add a value	Add a value

Issues 1 | Dependencies 20

Search…

SEVERITY

High	0
Medium	1
Low	0

PRIORITY SCORE
Scored between 0 - 1000

FIXABILITY

1 of 1 issues — Sort by highest priority score

M com.google.guava:guava - Information Disclosure — SCORE 275

VULNERABILITY | CWE-200 | CVE-2020-8908 | CVSS 5.5 MEDIUM | SNYK-JAVA-COMGOOGLEGUAVA-1015415

Introduced through	org.seleniumhq.selenium:selenium-java@3.141.0	Exploit maturity	NO KNOWN EXPLOIT
Fixed in	com.google.guava:guava@30.0-android, @30.0-jre		

Show more details