Dependency vulnerability check in The Nistagram

Golang application

We decided to manually check for dependencies vulnerabilities in The Nistagram Golang application. The decision was made to use the Sonatype OSS Index as a base tool. It is a free catalogue of open source components and scanning tools to help developers identify vulnerabilities, understand risk, and keep their software safe.

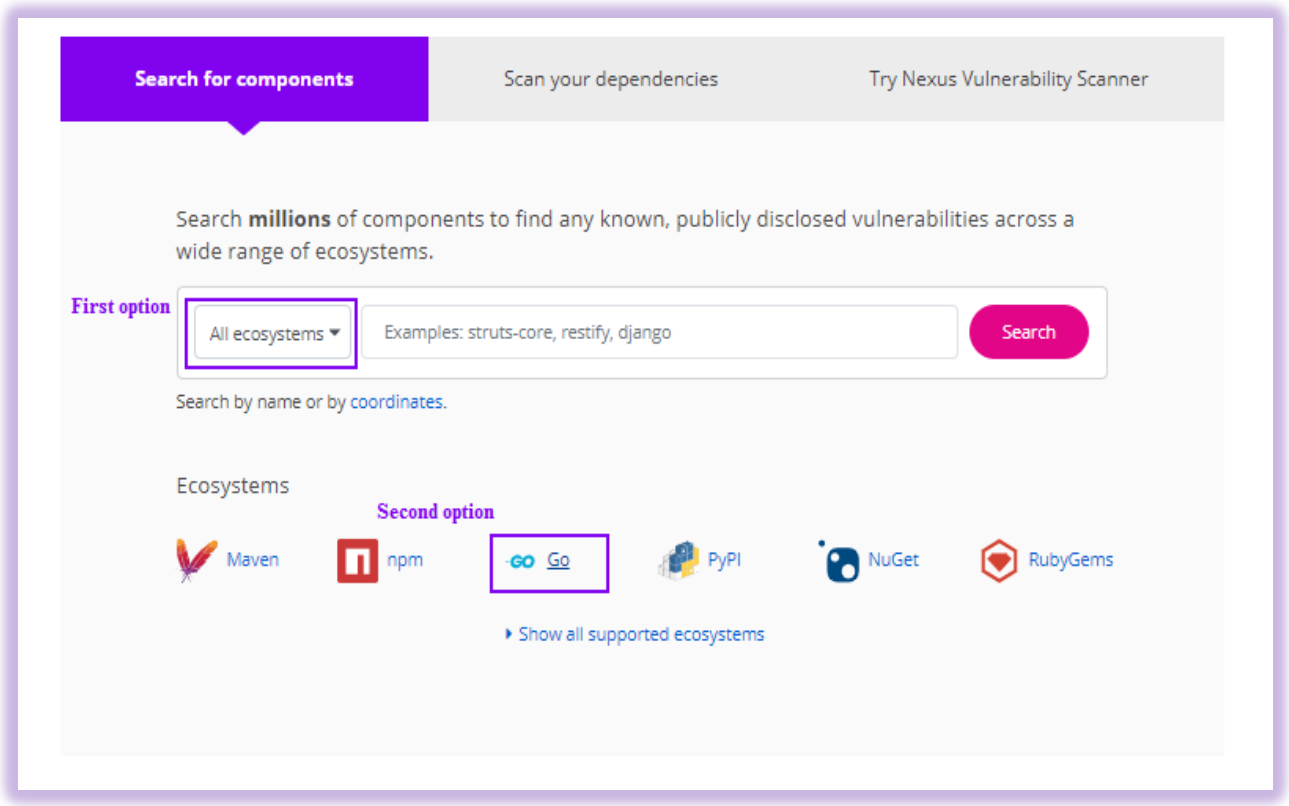
Steps for running a dependency vulnerability test

1. The initial step

Although it goes without saying, we will state that the initial step is to go to [The official website of the OSS index].

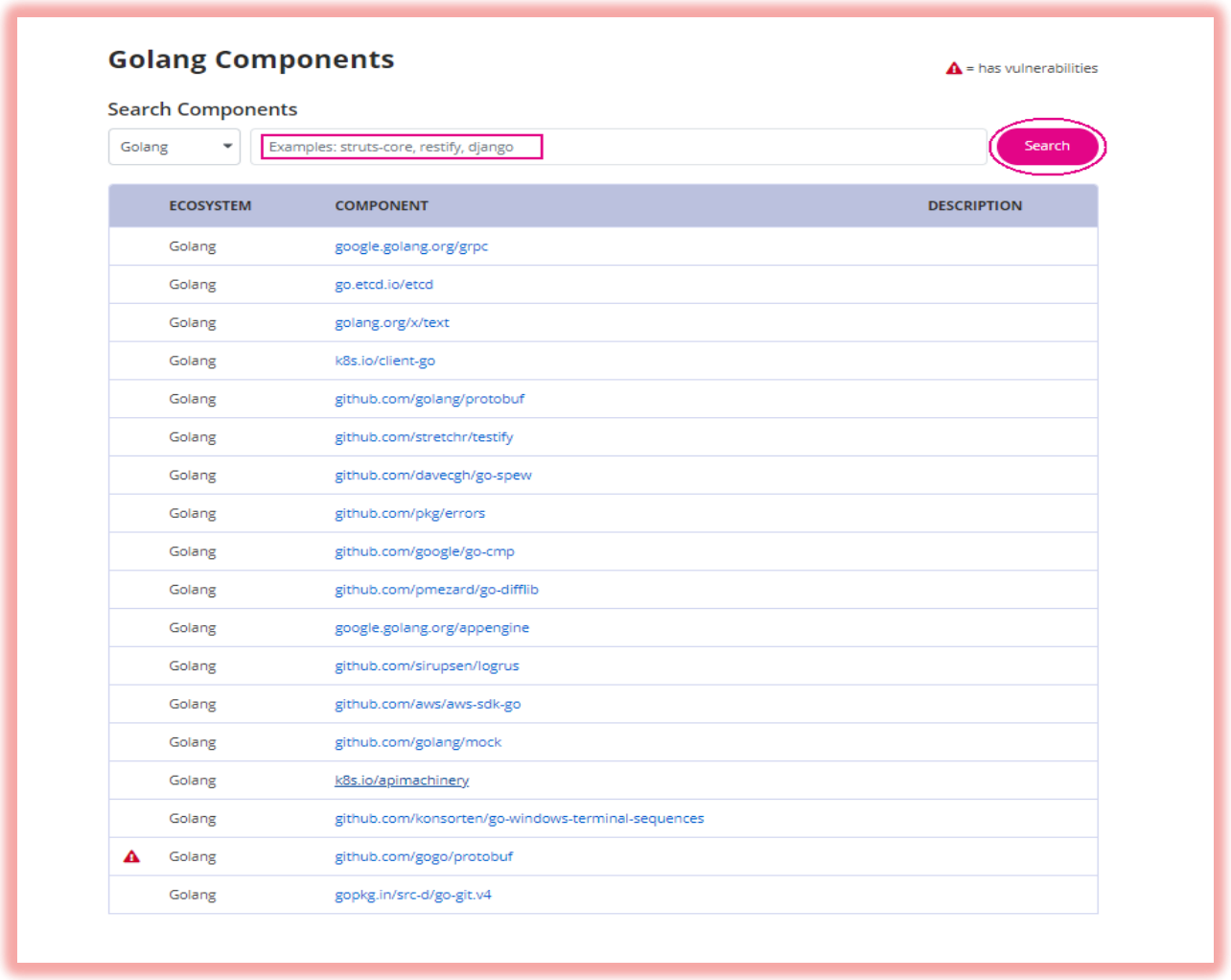
2. Ecosystem selection

Ecosystem selection facilitates the verification and search for dependency vulnerabilities, we can do this in two ways - by selecting an item from the combo box or simply clicking on the icon that represents our ecosystem in which we develop the application.



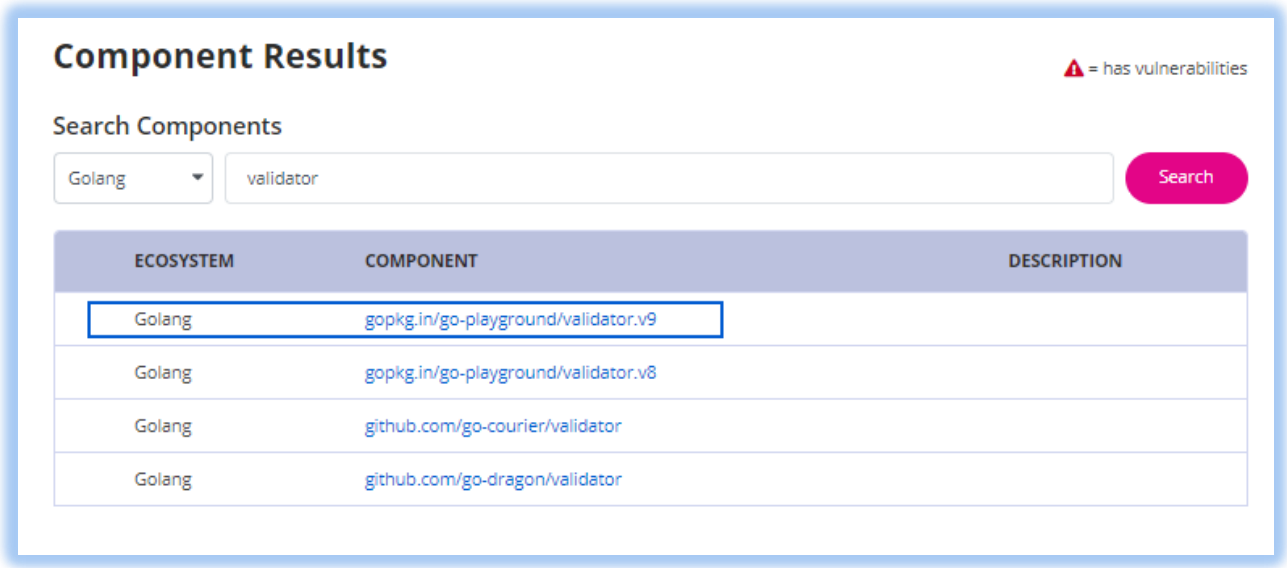
3. Search vulnerabilities for chosen dependency

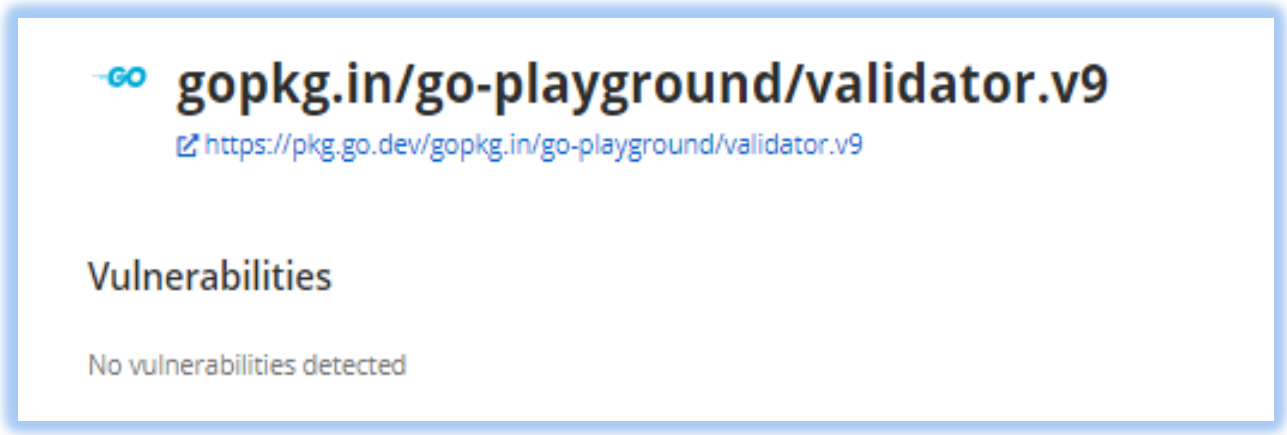
The search is very simple, all you have to do is enter the name of the dependency in the input field, and then click on the search button. After that display, you will see the search results, of course, there is a possibility that several other dependencies have the same name in the repository path, pay special attention to find your dependency.



Example of positive outcome- dependency has no vulnerabilities

Fortunately, mostly in our dependencies, no vulnerabilities have been detected. We have now chosen, e.g. "gopkg.in/go-playground/validator.v9" dependency that we use to validate the data on the back-end of our application, just to show you one example of a positive outcome.





Example of negative outcome - dependency has vulnerabilities

One dependency we used in our project has a vulnerability. It is this "github.com/dgrijalva /jwt-go" dependency that we used to work with tokens.

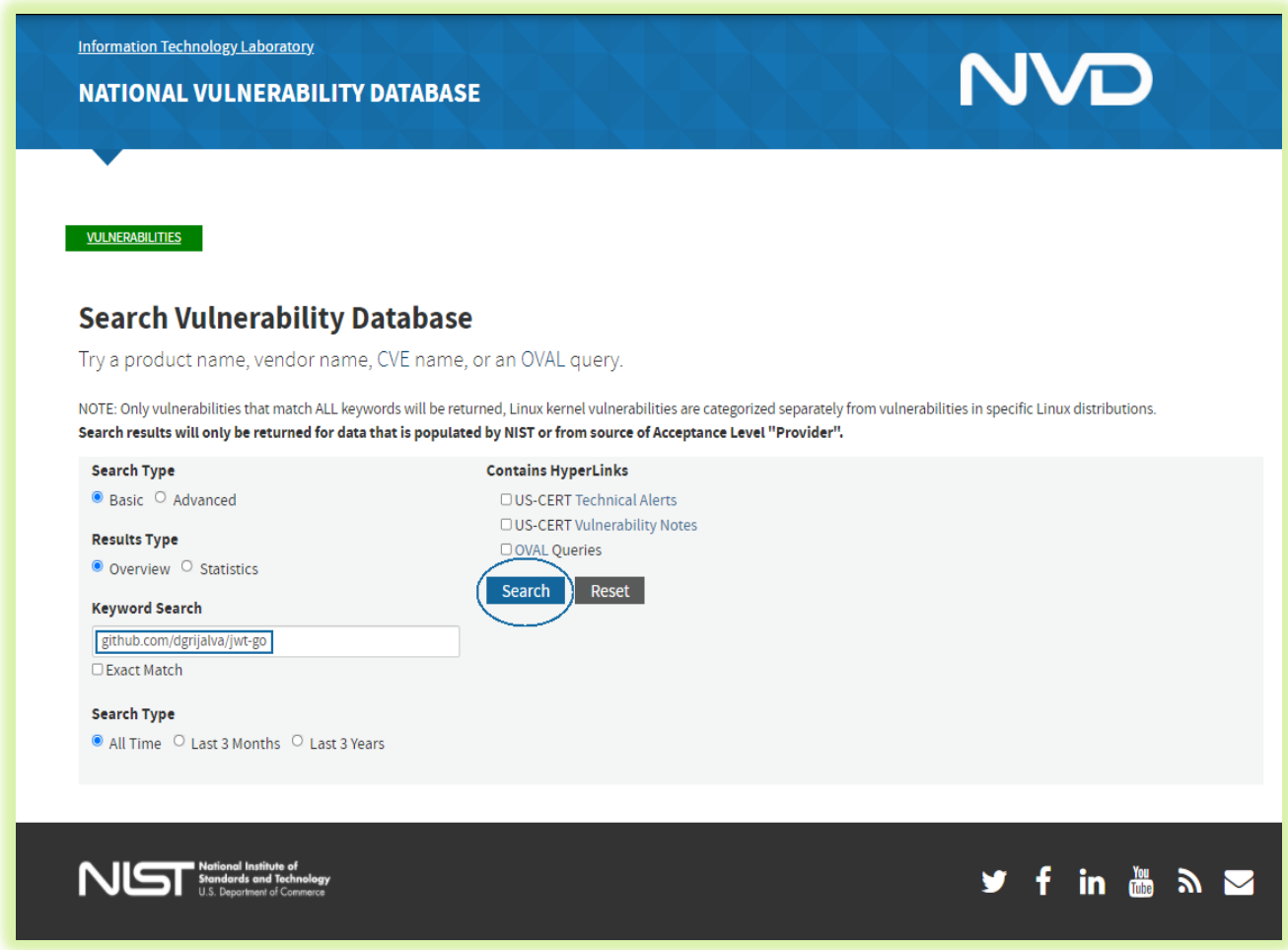
We will show you how we discovered exactly what that vulnerability is. We used the National Vulnerability Database (NVD), which is the U.S. government repository of standards-based vulnerability management, data represented using the Security Content Automation Protocol (SCAP). NVD includes databases of security checklists, security related software flaws, misconfigurations, product names, and impact metrics. In addition to providing a list of Common Vulnerabilities and Exposures (CVEs), the NVD scores vulnerabilities using the Common Vulnerability Scoring System (CVSS) which is based on a set of equations using metric such as access complexity. Using NVD, we will show you step by step in this example.

1. The First step

The first step is going to the official website of NVD - Search and Statistics.

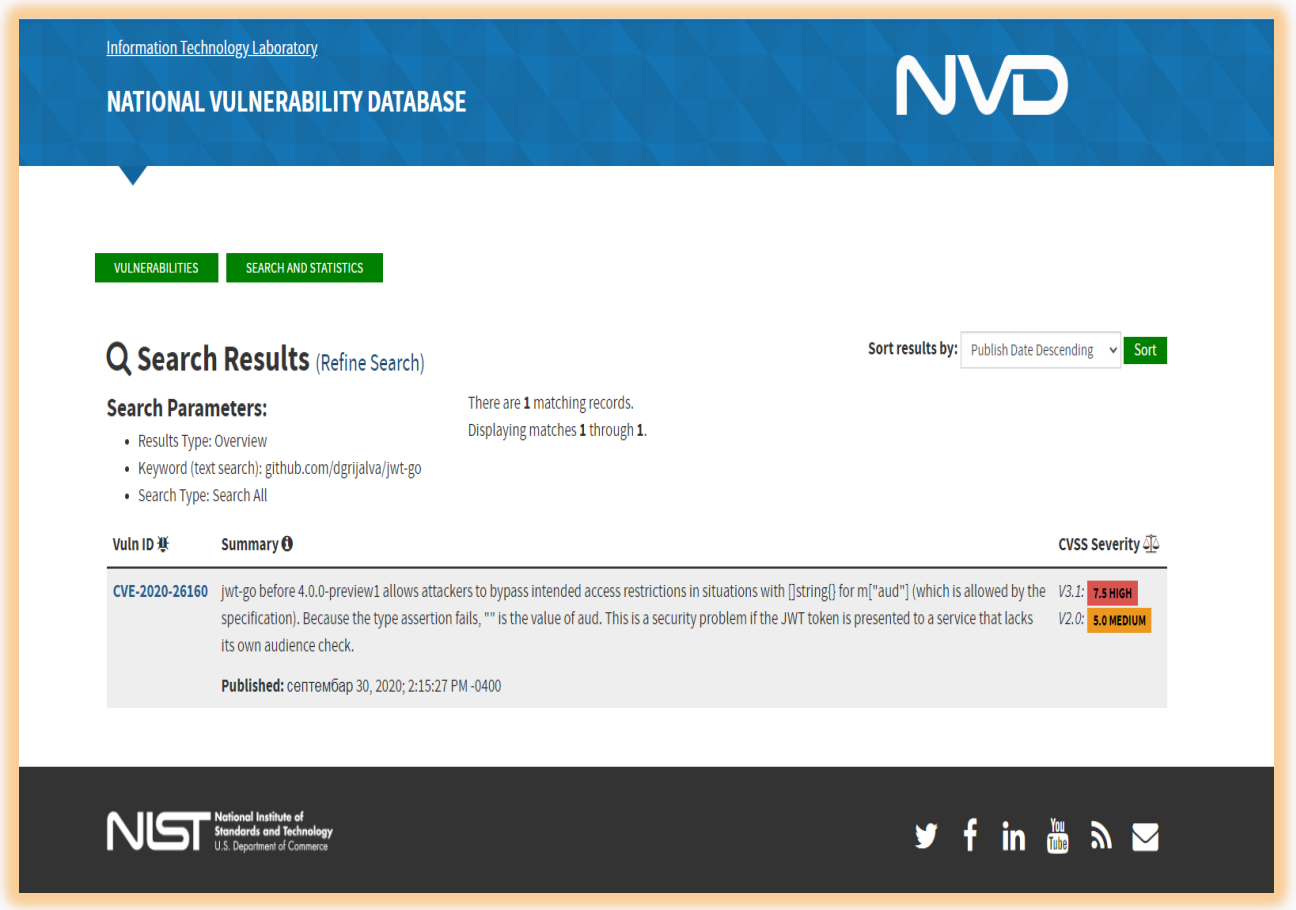
2. Search Vulnerability

We enter the keyword on the basis of which the search is performed, in our case "github.com/dgrijalva/jwt-go", and then click on the search button.



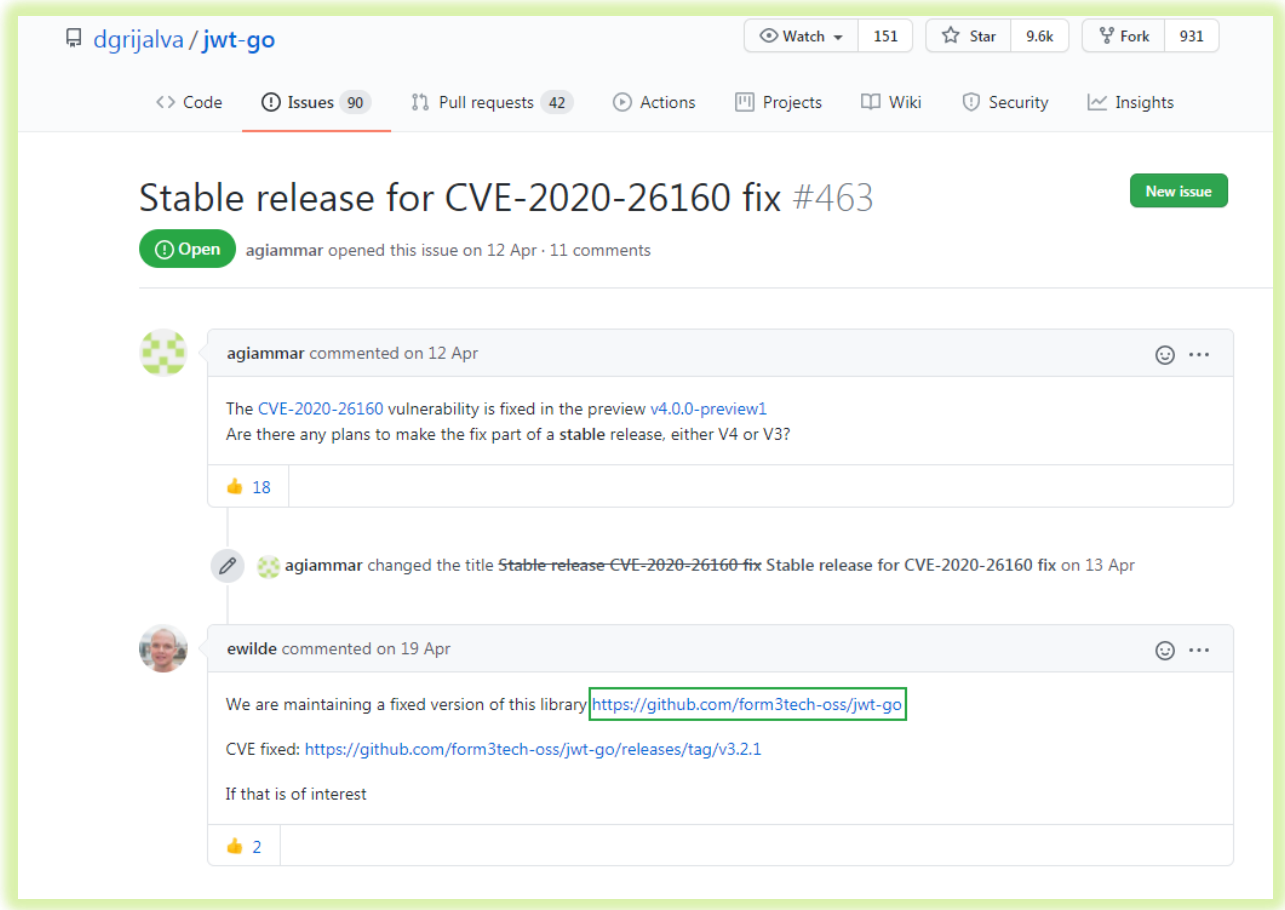
3. Vulnerability display

All vulnerabilities for the requested dependency are shown, in our case it is a vulnerability with identification code CVE-2020-26160. Below we will show you how we solved the found vulnerability.



Solving vulnerability

We looked for a solution to the found vulnerability and we found it. It was necessary to replace our vulnerable dependency with a new one. So instead of the critical dependency "github.com/dgrijalva/jwt-go", we now use "github.com/form3tech-oss/jwt-go v3.2.3" in The Nistagram application. We found an advice that solves the problem (photo below).



Checked Dependency lists

Dependency name	Vulnerabilities
github.com/fmtlib/fmt	No vulnerabilities detected
github.com/antchfx/xpath v1.1.11	No vulnerabilities detected
github.com/google/uuid	No vulnerabilities detected
github.com/mikespook/gorbac/v2	No vulnerabilities detected
gopkg.in/go-playground/validator.v9	No vulnerabilities detected
github.com/goccy/go-json	No vulnerabilities detected
github.com/json-iterator/go	No vulnerabilities detected
github.com/encoding/json	No vulnerabilities detected
net/http	No vulnerabilities detected

io/ioutil	No vulnerabilities detected
bytes	No vulnerabilities detected
os	No vulnerabilities detected
github.com/tdewolff/strconv	No vulnerabilities detected
github.com/golage/strings	No vulnerabilities detected
github.com/golangplus/time	No vulnerabilities detected
github.com/gorilla/handlers v1.5.1	No vulnerabilities detected
github.com/gorilla/mux v1.8.0	No vulnerabilities detected
github.com/lib/pq v1.10.2	No vulnerabilities detected
gorm.io/driver/postgres v1.1.0	No vulnerabilities detected
gorm.io/gorm v1.21.10	No vulnerabilities detected
gopkg.in/mail.v2	No vulnerabilities detected
github.com/sirupsen/logrus v1.4.2	No vulnerabilities detected
gopkg.in/alexcesaro/quotedprintable.v3	No vulnerabilities detected
gopkg.in/go-playground/assert.v1	No vulnerabilities detected
golang.org/x/crypto v0.0.0-20210513164829-c07d793c2f9a	No vulnerabilities detected
github.com/leodido/go-urn v1.2.1	No vulnerabilities detected
github.com/go-playground/universal-translator v0.17.0	No vulnerabilities detected
github.com/form3tech-oss/jwt-go v3.2.3+incompatible	No vulnerabilities detected

Front-end Node Package Manager (npm) for the JavaScript programming language doesn't show any warnings, i.e. vulnerabilities, so we checked all dependencies and they are safe.

Conclusion

Except for solving the vulnerability that we presented in detail as an example, we had no others. We have achieved satisfactory results, and we have made our Nistagram application have a high security.