Dependency Check using Gradle in Go

Adding Gradle Dependency Check

The following build.gradle will perform the vulnerability check against our Go project.

build.gradle

```
apply plugin: 'org.owasp.dependencycheck'

buildscript {
    repositories {
        mavenCentral()
    }
    ext {
        owaspPluginVersion = '7.0.3'
    }
    dependencies {
        classpath "org.owasp:dependency-check-gradle:${owaspPluginVersion}"
    }
}

dependencyCheck {
    failOnError = false
    format = 'ALL'
    outputDirectory = "${buildDir}/reports/owasp"
    suppressionFile = "${projectDir}/owasp-suppressions.xml"

    analyzers {
        experimentalEnabled = true
        golangModEnabled = true
        pathToGo = '/usr/local/go/bin/go'
    }
}

task build() {
    // empty
}

build.dependsOn (dependencyCheckAnalyze)
```

Running the Dependency Check

Running the build will perform the dependency check

```
$ ./gradlew build
```

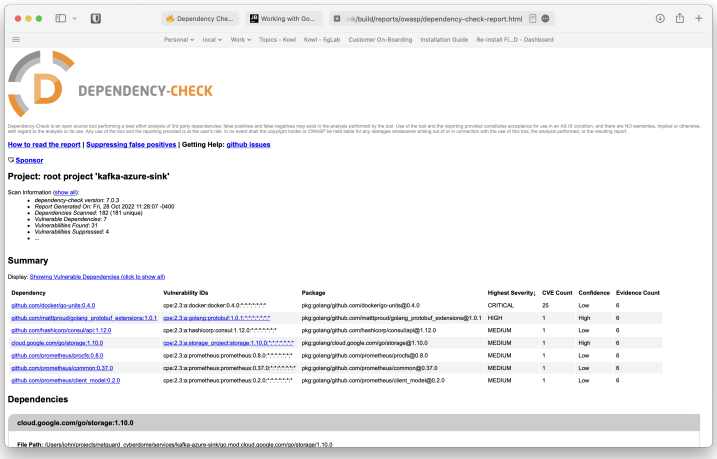
Sample Vulnerability Report

```
...
> Task :dependencyCheckAnalyze
Verifying dependencies for project kafka-azure-sink
Checking for updates and analyzing dependencies for vulnerabilities
Generating report for project kafka-azure-sink
Found 31 vulnerabilities in project kafka-azure-sink


One or more dependencies were identified with known vulnerabilities in kafka-azure-sink:

go.mod (pkg:golang/cloud.google.com/go/storage@1.10.0, cpe:2.3:a:storage_project:storage:1.10.0:*:*:*:*:*:*:*)
: CVE-2021-20291
go-units@v0.4.0 (pkg:golang/github.com/docker/go-units@0.4.0, cpe:2.3:a:docker:docker:0.4.0:*:*:*:*:*:*:*) :
CVE-2014-0047, CVE-2014-0048, CVE-2014-5277, CVE-2014-5278, CVE-2014-5282, CVE-2014-6407, CVE-2014-8178, CVE-
2014-8179, CVE-2014-9356, CVE-2014-9358, CVE-2015-3627, CVE-2015-3630, CVE-2015-3631, CVE-2016-3697, CVE-2017-
14992, CVE-2019-13139, CVE-2019-13509, CVE-2019-15752, CVE-2019-16884, CVE-2019-5736, CVE-2020-27534, CVE-2021-
21284, CVE-2021-21285, CVE-2021-3162, CVE-2022-25365
go.mod (pkg:golang/github.com/hashicorp/consul/api@1.12.0, cpe:2.3:a:hashicorp:consul:1.12.0:*:*:*:*:*:*:*) :
CVE-2022-40716
golang_protobuf_extensions@v1.0.1 (pkg:golang/github.com/matttproud/golang_protobuf_extensions@1.0.1, cpe:2.3:a:
golang:protobuf:1.0.1:*:*:*:*:*:*:*) : CVE-2021-3121
client_model@v0.2.0 (pkg:golang/github.com/prometheus/client_model@0.2.0, cpe:2.3:a:prometheus:prometheus:0.2.0:
*:*:*:*:*:*:*) : CVE-2019-3826
go.mod (pkg:golang/github.com/prometheus/common@0.37.0, cpe:2.3:a:prometheus:prometheus:0.37.0:*:*:*:*:*:*:*) :
CVE-2019-3826
go.mod (pkg:golang/github.com/prometheus/procfs@0.8.0, cpe:2.3:a:prometheus:prometheus:0.8.0:*:*:*:*:*:*:*) :
CVE-2019-3826


See the dependency-check report for more details.
```

Addressing Vulnerabilities

Open the report in your browser:

```
$ open build/reports/owasp/dependency-check-report.html
```



When a vulnerability is found we will need to do the following things:

Try to fix the vulnerability by updating the library

```
module kafka-azure-sink

go 1.18

require (
...
        github.com/docker/go-units v0.5.0
)
```

Issue the following commands to fix the go.mod/go.sum files.

```
$ go mod vendor
$ go mod tidy
```

Build the project to verify all is well

```
go build -o ./kafka-azure-sink ./src/cmd/svc
```

Re-run the Gradle Build

```
./gradlew build
```

Finding the Dependency

We can try to figure out who is linking the library in question using the go mod graph command.

```
$ go mod graph |grep <dependency>
```

Example - **Find out who is using storage@v1.14.0**

```
$ go mod graph |grep storage |grep 1.14.0

github.com/spf13/afero@v1.8.2 cloud.google.com/go/storage@v1.14.0                              <-- pulled in by
afero
cloud.google.com/go/storage@v1.14.0 cloud.google.com/go@v0.75.0
cloud.google.com/go/storage@v1.14.0 github.com/golang/protobuf@v1.4.3
cloud.google.com/go/storage@v1.14.0 github.com/google/go-cmp@v0.5.4
cloud.google.com/go/storage@v1.14.0 github.com/googleapis/gax-go/v2@v2.0.5
cloud.google.com/go/storage@v1.14.0 golang.org/x/mod@v0.4.1
cloud.google.com/go/storage@v1.14.0 golang.org/x/oauth2@v0.0.0-20210218202405-ba52d332ba99
cloud.google.com/go/storage@v1.14.0 golang.org/x/sys@v0.0.0-20210225134936-a50acf3fe073
cloud.google.com/go/storage@v1.14.0 golang.org/x/tools@v0.1.0
cloud.google.com/go/storage@v1.14.0 google.golang.org/api@v0.40.0
cloud.google.com/go/storage@v1.14.0 google.golang.org/genproto@v0.0.0-20210226172003-ab064af71705
cloud.google.com/go/storage@v1.14.0 google.golang.org/grpc@v1.35.0

$ go mod graph |grep afero

kafka-azure-sink github.com/spf13/afero@v1.8.2
github.com/spf13/afero@v1.8.2 cloud.google.com/go/storage@v1.14.0
github.com/spf13/afero@v1.8.2 github.com/googleapis/google-cloud-go-testing@v0.0.0-20200911160855-bcd43fbb19e8
github.com/spf13/afero@v1.8.2 github.com/pkg/sftp@v1.13.1
github.com/spf13/afero@v1.8.2 golang.org/x/crypto@v0.0.0-20211108221036-ceb1ce70b4fa
github.com/spf13/afero@v1.8.2 golang.org/x/oauth2@v0.0.0-20210218202405-ba52d332ba99
github.com/spf13/afero@v1.8.2 golang.org/x/text@v0.3.4
github.com/spf13/afero@v1.8.2 google.golang.org/api@v0.40.0
github.com/spf13/viper@v1.13.0 github.com/spf13/afero@v1.8.2                                <--- pulled in by
viper
```

Now that we have found the import that we are using, we can check to see if the library can be updated to a newer version. If not, we need to see if the error is really a problem.

Suppressing the Vulnerability

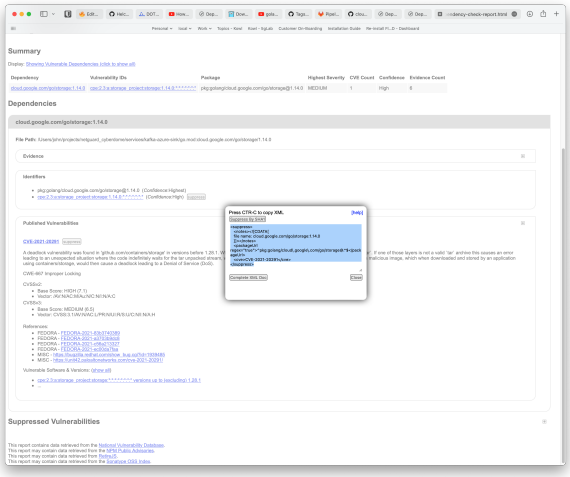
Vulnerability

```
File Path: /Users/john/projects/netguard_cyberdome/services/kafka-azure-sink/go.mod:cloud.google.com/go/storage
/1.14.0

CVE-2021-20291

A deadlock vulnerability was found in 'github.com/containers/storage' in versions before 1.28.1. When a
container image is processed, each layer is unpacked using `tar`. If one of those layers is not a valid `tar`
archive this causes an error leading to an unexpected situation where the code indefinitely waits for the tar
unpacked stream, which never finishes. An attacker could use this vulnerability to craft a malicious image,
which when downloaded and stored by an application using containers/storage, would then cause a deadlock
leading to a Denial of Service (DoS).
```

In the above section, we found that storage@1.14.0 is used by afero@v1.8.2 which is used by viper@v1.13.0.

We use viper for reading our environment variables. We do not use viper with an afero file system, so we can confidently suppress this vulnerability.

From the dependency-check-report.html we can click on the suppress button and copy the text to our **owasp-suppressions.xml** file.



It is best to add some information around why we suppressed the vulnerability.

owasp-suppressions.xml

```
<suppressions>
...
   <suppress>
      <notes><![CDATA[
      file name: cloud.google.com/go/storage:1.14.0
      github.com/spf13/afero@v1.8.2 included as port of github.com/spf13/viper@v1.13.0
      We are not using the afero filesystem with viper.
      ]]></notes>
      <packageUrl regex="true">^pkg:golang/cloud\.google\.com/go/storage@.*$</packageUrl>
      <cve>CVE-2021-20291</cve>
   </suppress>

</suppressions>
```