

Laboratuvar: Statik Analiz Kullanımı

Gerekli tahmini süre: 30 dakika

Statik Analiz Kullanımı için uygulamalı laboratuvara hoş geldiniz! Statik analiz, yürütmeden önce kaynak kodunu otomatik olarak inceleyen bir hata ayıklama yöntemidir. Bu laboratuvara, statik analiz hakkında daha fazla bilgi edinecek ve projeniz için SonarQube'ü nasıl kurup yapılandıracağınızı ve kullanacağınızı öğreneceksiniz.

Öğrenme Hedefleri

Bu laboratuvarın sonunda, şunları yapabileceksiniz:

- Statik analiz kullanmanın faydalarnı tanımlamak
- SonarQube'u kurmak ve yapılandırmak
- Statik analiz taramaları gerçekleştirmek
- SonarQube'den gelen güvenlik raporlarını yorumlamak

Statik Kod Analizini Anlamak

Statik kod analizi, kaynak kodunu inceleyen bir hata ayıklama yöntemi olup, programı çalıştırmadan kodu incelemek anlamına gelen bir çalışma ortamında gerçekleştirili. Bu uygulama, kaynak kodu analizi olarak da bilinir.

Test işlemleri geleneksel olarak kodlayıcı tarafından program çalıştırılarak yapılırken, kaynak kodu analizi bir program tamamlanmadan bile yapılabilir. Bu, geliştiricilere hataları erken yakalama avantajı sağlar.

Kaynak kodu analizi araçları, **Statik Uygulama Güvenlik Testi (SAST)** araçları olarak da bilinir, kaynak kodunu güvenlik açıklarını bulmak için analiz eder. SAST araçları, IDE'meye eklenebilir. Yazılım geliştirme sürecinde sorunları tespit etmenize yardımcı olabilir ve özellikle geliştirme döngüsünün ilerleyen aşamalarında güvenlik açıklarını bulmaya kıyasla zaman ve çaba tasarrufu sağlayabilir.

Bir sonraki bölümde, sürekli kod kalitesi değerlendirme için SonarSource tarafından oluşturulmuş açık kaynaklı bir platform olan **SonarQube**'yi kullanacağınız. Proje dallarınız ve çekme istekleriniz arasında sürekli kod denetimi sağlamak için mevcut iş akışını entegre edilebilir.

Başlarken

CI/CD hatlarında özel bir SonarQube aşaması kurulu olsa da, SonarQube'u yerel olarak çalıştırma bilmek faydalıdır. Bunu yaparak, kodu değiştirme, tarama yapma ve sonuçları görüntüleme geri bildirim döngüsünü kısaltabilirsiniz.

Bu laboratuvar boyunca, SonarQube'u kurma ve Docker ile Cloud IDE'deki bir uygulamanın analizini yürütme konusunda rehberlik alacaksınız. Her şey sağ paneldeki terminalde yapılabilir. Docker yüklü olan herhangi bir ortamda bu laboratuvari kolayca tekrarlayabileceksiniz, bu da kendi geliştirici çalışma istasyonunuza içerir.

Bir SonarCube sunucusu kurmak için şunları yapacağınız:

1. SonarQube ve PostgreSQL'in iletişim kurması için bir Docker ağı oluşturun
2. Docker konteynerinde PostgreSQL'i çalıştırın
3. Docker konteynerinde SonarQube'u çalıştırın
4. SonarQube tarayıcı Docker görüntüsünü indirin
5. sonar-scanner -c ile Docker konteynerini kullanarak tarama yapmak için bir takma ad oluşturun
6. Bazi kodları tarayın ve sonuçları yorumlayın

Başlamak için **İleri**'ye tıklayın.

Adım 1: PostgreSQL veritabanını kurun

SonarQube, doğru çalışmak için bir veritabanına bağlıdır. SonarQube, bir veritabanı olmadan bir docker konteynerinde çalışabilse de, konteyner silindiğinde tüm veriler de silinir. Bu, hızlı geliştirme taramaları için uygun olabilir, ancak zamanla bir geçmiş oluşturmak istiyorsanız, yerel geliştirme için bile harici bir veritabanı kullanmak en iyi uygulamadır.

Bu laboratuvar çalışmasında, PostgreSQL veritabanını docker imajını çekerek kullanacağınız. Bunu yapmadan önce, veritabanını SonarQube ile ve daha sonra SonarScanner ile bağlamak için bir Docker ağına ihtiyacınız olacak. Bu, ilk göreviniz olacak.

Göreviniz

1. Üst menü çubuğundan Terminal -> Yeni Terminal seçeneğini seçerek bir terminal açın.
2. Ardından, terminal kabuğundan docker ağı oluşturmak için docker network komutunu çalıştırın ve bu ağa mynet adını verin.

```
docker network create mynet
```

Oluşturduğunuz docker ağı mynet olarak adlandırılacak ve bu, daha sonra oluşturacağınız PostgreSQL konteyneri ile SonarQube konteyneri arasında iletişim kurmak için kullanılacaktır.

3. Son olarak, bir PostgreSQL docker konteyneri oluşturmak için docker run komutunu kullanın:

```
docker run --name postgres -e POSTGRES_USER=root -e POSTGRES_PASSWORD=Test12345 -p 5432:5432 --network mynet -d postgres
```

Konteyneri çalıştırma isteğinde bulunarak, Docker'in imajı indirmesi gerektiğini anlayacağınızı göreceksiniz. postgres için Docker imajının katmanlarını indirme ve genişletme hakkında birçok çıktı göremelisiniz.

Sonuçlar

Son çıktılarınız aşağıdaki gibi görünmelidir:

```
theia@theiadocker-rofrano:/home/project$ docker run --name postgres -e POSTGRES_USER=root -e POSTGRES_PASSWORD=postgres -p 5432:5432 --network mynet -d postgres
45
Unable to find image 'postgres:latest' locally
latest: Pulling from library/postgres
7a6db449b51b: Pull complete
b4f184bc0704: Pull complete
606a73c0d34a: Pull complete
c39f1600d2b6: Pull complete
31f42f92b0fe: Pull complete
c8b67d2b0354: Pull complete
31107b8480ee: Pull complete
b26434cf8bfa: Pull complete
36220bd76bfa: Pull complete
b79e75c4a0c2: Pull complete
cc1ab699dda5: Pull complete
37312064dd9b: Pull complete
4bce56fcbe5: Pull complete
Digest: sha256:befb4cdc1d944bd89784b9caa287cf025f0720f9a02436038124163accd177dc
Status: Downloaded newer image for postgres:latest
6010f98f048f2eb59a066b887d6d96b6e60898328793f899c99848f3d613a277
```

Adım 2: SonarQube sunucusunu kurma

PostgreSQL veritabanımız çalıştırığınıza göre, SonarCube sunucusunu oluşturabilir ve veritabanına bağlayabiliriz. SonarQube'u kurmak ve çalıştırmak için uygun Java ortamına sahip olmak zorunda kalmamak için, SonarQube sunucusunu çalıştırınca bir Docker konteyneri kullanıcasınız. Neye ki, kullanabileceğiniz bir tane SonarQube sağlıyor.

Göreviniz

Aşağıdaki docker run komutunu kullanarak 9000 portunda bir SonarQube docker konteynerini çalıştırın:

```
docker run -d --name sonarqube -p 9000:9000 -e sonar.jdbc.url=jdbc:postgresql://postgres/postgres -e sonar.jdbc.username=root -e sonar.jdbc.password=Test12345 --network mynet sonarqube
```

Bu komut, SonarQube'un konteyneri dışındaki iletişimini için 9000 portunu açmak amacıyla -p bayrağını kullanır. Ayrıca, -e komutuyla birkaç ortam değişkeni ayarlar. Docker kullanarak sunucuyu masaüstüne çalıştırın, bir geliştirici olarak size muazzam bir özgürlük ve esneklik sağlar.

Yine, SonarQube docker imajının katmanlarının indirildiğini ve çıkarıldığını göreceksiniz. Nihai çıktımız aşağıdaki benzer bir şey olmalıdır:

```
theia@theiadocker-rofrano:/home/project$ docker run -d --name sonarqube -p 9000:9000 -e sonar.jdbc.url=jdbc:postgresql://postgres/postgres -e sonar.jdbc.username=root -e sonar.jdbc.password=Test12345 --network mynet sonarqube
46
Unable to find image 'sonarqube:latest' locally
latest: Pulling from library/sonarqube
9621f1afde84: Pull complete
0da9106727c7: Pull complete
129c5a3f9c32: Pull complete
Digest: sha256:3fa9a76948fab6fafa41950bee256afea943773744723b5e4f38b340643516b9
Status: Downloaded newer image for sonarqube:latest
0be9c3c828e62f5e83d2d236f0fe32ce04e18ef355e5f41008c5680c228bdda8
```

Başarıyı Kontrol Et

Artık hem PostgreSQL hem de SonarQube imajları indirilmiş ve her iki konteyner de başlatılmış olduğuna göre, çalışıklarını kontrol etmek için docker ps komutunu kullanabilirsiniz.

Konteynerleri kontrol etmek için docker ps komutunu kullanın:

```
docker ps
```

postgres adında bir konteyner ve sonarqube adında bir konteyner ile aşağıdaki gibi bir çıktı görelimiz:

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
0be9c3c828e6	sonarqube	"/opt/sonarqube/bin/..."	4 minutes ago	Up 3 minutes	0.0.0.0:9000
000/tcp	sonarqube				
6010f98f048f	postgres	"docker-entrypoint.s..."	21 minutes ago	Up 20 minutes	0.0.0.0:5432
432/tcp	postgres				

Tebrikler! SonarQube'unuz artık çalışıyor!

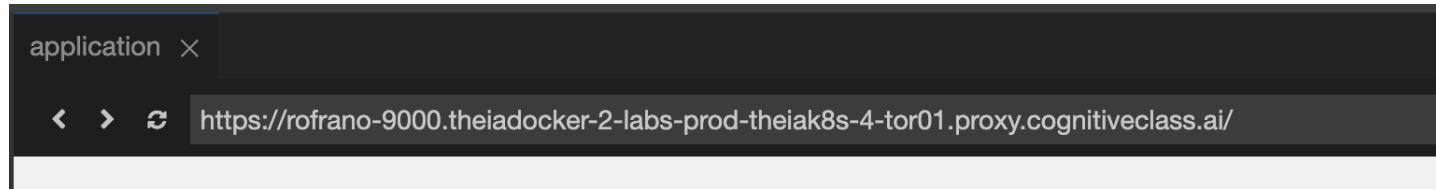
Artık kurulum kısmını tamamladınız ve projeniz için bir tarayıcı token'ı oluşturma aşamasına geçmeye hazırlız!

Adım 3: SonarQube'a Giriş Yapın

Artık SonarQube'u kullanmaya准备了。Aşağıdaki [SonarQube UI'yi Başlat] butonuna tıklayarak web arayüzünü başlatılabilir. SonarQube'un başlaması biraz zaman alabilir.

[SonarQube UI'yi Başlat]

Not: SonarQube UI oldukça büyük olduğu için, Cloud IDE yerine bu web sayfasını tarayıcınızda açmak isteyebilirsiniz. Cloud IDE tarayıcısını kullanarak gezinmek zor olabilir. **SonarQube UI'yi Başlat** butonuna tıkladıkten sonra, terminal panelinin sağ üst köşesinde yeni bir sayfa açacak bir ok simgesi olacak。



Kimlik bilgilerini değiştirin

SonarQube'a giriş yapmak için aşağıdaki varsayılan kimlik bilgilerini kullanın.

username: admin
password: admin

Sonra [Log in] butonuna tıklayın. Giriş yaptıktan sonra, şifrenizi değiştirmeniz istenecektir.

Not: Yeni şifrenizi kaydetmemi unutmayın!

Log In to SonarQube

admin

•••••

Log in Cancel

Şifreyi sıfırlama işlemini tamamladıktan sonra, siz SonarQube'un ana sayfasına yönlendirilecektir.

Adım 4: Bir SonarQube Projesi Oluşturun

Kodunuz üzerinde SonarQube tarayıcısını çalıştırma için önce bir proje token'i oluşturmanız gerekecek. Bir token oluşturmanın birçok yolu vardır, ancak bu laboratuvar için **manuel** kurulumunu kullanacaksınız.

Sol alttaki **Manuel** simgesine tıklayın.

How do you want to create your project?

Do you want to benefit from all of SonarQube's features (like repository import and Pull Request decoration)? Create your project from your favorite DevOps platform.

First, you need to set up a DevOps platform configuration.

Import from Azure DevOps

[Setup](#)

Import from Bitbucket Cloud

[Setup](#)

Import from Bitbucket Cloud

Import from GitHub

[Setup](#)

Import from GitLab

[Setup](#)

Are you just testing or have an advanced use-case? Create a project manually.

[Create project manually](#)

Sonraki sayfada, aşağıdaki adımları izleyerek bir proje oluşturun:

1. Proje görüntü adını `temp` olarak ayarlayın.
2. Proje anahtarını `temp` olarak ayarlayın (bu varsayılan olarak gerçekleşecektir).
3. `main` dalının seçili olduğunu emin olun.
4. Devam etmek için `Next` butonuna basın.

Not: Eğer bir hata mesajı alırsanız, tekrar deneyin.

Create a project

Project display name *

Up to 255 characters. Some scanners might override the value you provide.

Project key *

The project key is a unique identifier for your project. It may contain up to 400 characters. Allowed characters are alphanumeric, '-' (dash), '_' (underscore), '.' (period) and ':' (colon), with at least one non-digit.

Main branch name *

The name of your project's default branch [Learn More](#)

[Next](#)



Embedded database should be used for evaluation purposes only

The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database.

Lütfen Küresel ayarı kullan seçeneğini seçin ve ardından Proje oluştur butonuna tıklayın.

Set up project for Clean as You Code

The new code definition sets which part of your code will be considered new code. This helps you focus attention on the most recent changes to your project, enabling you to follow the Clean as You Code methodology. Learn more: [Defining New Code](#)

Choose the baseline for new code for this project

Use the global setting

Previous version

Any code that has changed since the previous version is considered new code.
Recommended for projects following regular versions or releases.

Define a specific setting for this project

Previous version

Any code that has changed since the previous version is considered new code.
Recommended for projects following regular versions or releases.

Number of days

Any code that has changed in the last x days is considered new code. If no action is taken on a new issue after x days, this issue will become part of the overall code.
Recommended for projects following continuous delivery.

Reference branch

Choose a branch as the baseline for the new code.
Recommended for projects using feature branches.

Create project

Sonraki sayfada, deposunu nasıl analiz etmek istediğiniz soran bölümde Yerel seçeneğini seçin.

Analysis Method

Use this page to manage and set-up the way your analyses are performed.

How do you want to analyze your repository?

 With Jenkins

 With GitHub Actions

 With Bitbucket Pipelines

 With GitLab CI

 With Azure Pipelines

Other CI

SonarQube integrates with your workflow no matter you're using.

Locally

Use this for testing or advanced use-case. Other modes are recommended to help you set up your CI environment.

Sonraki adımda, projeniz için bir token oluşturacaksınız.

Adım 5: SonarQube Tarayıcı Token'i Oluşturma

Kodunuzu taramadan önce bir token oluşturmanız gerekecek. Token'ı [Projenizi Analiz Edin](#) sayfasında **Token sağlayın** adımında oluşturabilirsiniz.

Oluştur butonuna tıklayın.

Analyze your project

We initialized your project on SonarQube, now it's up to you to launch analyses!

1 Provide a token

Generate a project token

Token name 

Expires in

Analyze "temp"

30 days

Generate



The token is used to identify you when an analysis is performed. If it has been compromised, you can revoke it at any point in time in your [user account](#).

2 Run analysis on your project

Sonra, oluşturulan token'ı göreceksiniz. Token metnini vurgulayıp kopyalayın ve güvenli bir yere yapıştırın. Daha sonra taramalarınızı göndermek için buna ihtiyacınız olacak.

Ardından **Devam** butonuna tıklayın.

Analyze your project

We initialized your project on SonarQube, now it's up to you to launch analyses!

1 Provide a token

Analyze "temp": **sqp_c02d45ac9a10af0999f559d5b6a8879051942d02** 

Copy

The token is used to identify you when an analysis is performed. If it has been compromised, you can revoke it at any point in time in your [user account](#).

Continue

2 Run analysis on your project

Bu, projenizin yapılandırmasıyla ilgili bazı soruları yanıtlamamanız gereken bir sayfaya yönlendirecek; örneğin, kullanılan dil ve işletim sistemi (OS).

2 Run analysis on your project

What option best describes your build?

Maven Gradle .NET Other (for JS, TS, Go, Python, PHP, ...)

1

What is your OS?

2 Linux Windows macOS

Download and unzip the Scanner for Linux

Visit the [official documentation of the Scanner](#) to download the latest version, and add the `bin` directory to the `PATH` environment variable.

Execute the Scanner

Running a SonarQube analysis is straightforward. You just need to execute the following commands in your project's folder:

```
sonar-scanner \
-Dsonar.projectKey=temp \
-Dsonar.sources=. \
-Dsonar.host.url=https://rofrano-9000.theiadocker-3-labs-prod-theiak8s-4-tor01.proxy.cognitiveservices.net \
-Dsonar.login=sqpc02d45ac9a10af0999f559d5b6a8879051942d02
```

Please visit the [official documentation of the Scanner](#) for more details.

Is my analysis done? If your analysis is successful, this page will automatically refresh in a few moments.

You can set up Pull Request Decoration under the project settings. To set up analysis with your favorite CI tool, see the [tutorial](#).

Check these useful links while you wait: [Branch Analysis](#), [Pull Request Analysis](#).

Seçimler için sunları seçmeliisiniz:

1. Diğer (JS, TS, Go, Python, PHP, ...)
2. Linux
3. Kopyala

Önemli!

Bu komut güvenli bir yere kopyalamak çok önemlidir! Kodunuz üzerinde tarayıcıyı çalıştırılabilmeniz için oluşturulan komutu göreceksiniz. Bu komut projenize özeldir.

O komutu bir yere kaydettiğinizden emin olun çünkü gelecekteki bir adımda buna ihtiyacınız olacak!

Adım 6: SonarQube Tarayıcısını Hazırlama

SonarQube sunucusunun, tarama sonuçlarını depolayan ayrı bir sistem olduğunu ve gerçek taramayı gerçekleştiren SonarQube tarayıcısından farklı olduğunu anlamak önemlidir. Şimdiye kadar, analiz sonuçlarını depolamak için bir veritabanı oluşturduk ve kullanıcı arayüzüni sunmak için bir SonarQube sunucusu sağladık.

SonarQube tarayıcısını Cloud IDE'de çalıştırmak için, ya yerel olarak kurabilir ya da docker imajını çekip docker konteynerini çalıştırabilirsiniz. Bu laboratuvar çalışmasında, docker imajını çekecek ve docker konteynerini çalıştıracağımız.

Göreviniz

1. Öncelikle, `docker pull` komutunu kullanarak Docker hub'dan `sonarsource/sonar-scanner-cli` imajını indirip yerel olarak kullanabilir hale getireceğiz.

```
docker pull sonarsource/sonar-scanner-cli
```

Not: İmajı çekmeşeniz, tarayıcıyı ilk çalıştırığınızda otomatik olarak çekilecektir. Bunu şimdi yapıyoruz ki daha sonra zaman kazanmış olalım.

2. Terminalde aşağıdaki bash alias komutunu çalıştırın; bu, `scanner-cli` docker konteynerini kullanarak tarayıcıyı daha sonra çalıştırmak için `sonar-scanner` adında bir takma ad oluşturur:

```
alias sonar-scanner='docker run --rm -v "$(pwd)":/usr/src sonarsource/sonar-scanner-cli'
```

Not: Bu komut, mevcut çalışma dizinini konteyner içinde `/usr/src` konumuna bir hacim olarak bağlıyor; `sonar-scanner` burada kaynak kodunu arıyor. Bunu kendi bilgisayarınızda da ayarlayabilirsiniz.

`sonar-scanner` komutuna geçirdiğiniz herhangi bir argüman, konteyner versiyonuna da iletilicektir. Bu sayede, Docker konteynerlerinde komutları, sanki bilgisayarınıza gerçekten kuruluymuş gibi kolayca çalıştırılabilirsiniz.

Artık tarayıcıyı hazırladığımıza göre, üzerinde analiz yapacağımız bir proje edinelim!

Adım 7: Örnek Proje Alma

Taramanız için biraz koda ihtiyacınız var. Güvenlik açıklarını taramak için IBM CI/CD kursundan bir projeyi kullanacağız.

Terminalde, bir CI/CD python projesini GitHub [depo](#) adresinden klonlamak için `git clone` komutunu kullanın ve ardından `wtecc-CICD_PracticeCode` proje dizinine `cd` ile geçin.

```
git clone https://github.com/ibm-developer-skills-network/wtecc-CICD_PracticeCode.git  
cd wtecc-CICD_PracticeCode
```

`ls -l` komutunu kullanarak dosyaları görüntüleyin:

```
ls -l
```

Aşağıdakileri görmelisiniz:

```
theia@theiadocker-rofrano:/home/project/wtecc-CICD_PracticeCode$ ls -l  
total 44  
-rw-r--r-- 1 theia users 491 Sep 12 19:41 Dockerfile  
drwxr-sr-x 8 theia users 4096 Sep 12 19:41 labs  
-rw-r--r-- 1 theia users 11357 Sep 12 19:41 LICENSE  
-rw-r--r-- 1 theia users 72 Sep 12 19:41 Procfile  
-rw-r--r-- 1 theia users 957 Sep 12 19:41 README.md  
-rw-r--r-- 1 theia users 327 Sep 12 19:41 requirements.txt  
drwxr-sr-x 3 theia users 4096 Sep 12 19:41 service  
-rw-r--r-- 1 theia users 331 Sep 12 19:41 setup.cfg  
drwxr-sr-x 2 theia users 4096 Sep 12 19:41 tests  
theia@theiadocker-rofrano:/home/project/wtecc-CICD_PracticeCode$ █
```

Artık bazı taramalar yapmaya hazırız.

Adım 8: Tarayıcıyı Çalıştırma

Artık kod üzerinde statik analiz yapmak için ihtiyacınız olan her seye sahipsiniz. Terminalde, "Adım 5: SonarQube Token'ı Oluştur" dan kaydettiğiniz komutu çalıştırın. Aşağıdakine benzer görünmelidir:

```
## THIS IS AN EXAMPLE ONLY ## DO NOT PASTE THIS ##  
sonar-scanner \  
  -Dsonar.projectKey=temp \  
  -Dsonar.sources=. \  
  -Dsonar.host.url=https://{{YOUR SONARQUBE URL}} \  
  -Dsonar.login={{YOUR PROJECT TOKEN}}
```

Göreviniz

1. Adım 5'te oluşturduğunuz SonarQube komutunu buraya yapıştırın. Komutunuz, taramanın çalışabilmesi için proje jetonunu da içeren tüm parametreleri içermelidir. Yukarıdaki örnek kodu kullanmayın.

2. Enter tuşuna bastığınızda, tarayıcı mevcut proje dizininizde statik bir analiz başlatır ve bir süre çalışır.

Tarama tamamlandığında, çıktıının sonu aşağıdaki gibi görünmeliidir:

```
roxy.cognitiveclass.ai/dashboard?id=temp  
INFO: Note that you will be able to access the updated dashboard once the server has processed  
report  
INFO: More about the report processing at https://rofrano-9000.theiadocker-3-labs-prod-theiak8s  
class.ai/api/ce/task?id=AYMzJIZaolQa5I6QBptT  
INFO: Analysis total time: 9.773 s  
INFO: -----  
INFO: EXECUTION SUCCESS  
INFO: -----  
INFO: Total time: 30.262s  
INFO: Final Memory: 16M/57M  
INFO: -----  
theia@theiadocker-rofrano:/home/project/sampleproject$ █
```

Adım 9: Tarama sonuçlarını yorumlama

Kod taraması ve analizi tamamlandığında, sonuçları SonarQube arayüzünde görebilirsiniz. Sonuçları yeni bir tarayıcı penceresinde görmek daha kolay olabilir. Sonuçlarla birlikte yeni bir tarayıcı penceresi açmak için [SonarQube UI'yi Tarayıcıda Başlat] butonuna basabilirsiniz.

SonarQube UI'yi Tarayıcıda Başlat

Raporu görüntülediğinizde, genel raporun geçerli olduğunu ancak bir güvenlik maddesinin işaretlendiğini göreceksiniz.

The screenshot shows the SonarQube interface. At the top, there's a navigation bar with links for Projects, Issues, Rules, Quality Profiles, Quality Gates, and Administration. Below the navigation bar, there are two project cards: 'temp' (starred) and 'master'. A message box at the top says, 'To benefit from more of SonarQube's features, set up analysis in your favorite CI.' On the left, a large green box indicates 'Passed' status with the message 'All conditions passed.' In the center, under 'MEASURES', there are four categories: 'New Code' (0 Bugs), 'Overall Code' (0 Vulnerabilities), 'Security Hotspots' (1), and 'Debt' (0.0%). A red arrow points to the 'Security Hotspots' section, which is highlighted with a red border. The 'Security Hotspots' section contains a link labeled 'Güvenlik Sıkak Noktaları'.

QUALITY GATE STATUS ?

MEASURES

New Code	Overall Code
0 Bugs	
0 Vulnerabilities	
1 Security Hotspots ?	C
0 Debt	C
0.0%	

Güvenlik Sıkak Noktaları yanındaki **1** bağlantısına tıkladığınızda bununla ilgili bazı bilgileri göreceksiniz.

Üst sekmeleerde aşağıdaki etiketleri görebilirsiniz:

- Risk nerede?
- Risk nedir?
- Riski değerlendirin
- Bunu nasıl düzeltebilirim?

🛡 1 Security Hotspots to review

Review priority: **HIGH**

Cross-Site Request Forgery (CSRF)

1



Make sure disabling CSRF protection is safe here.

service/__init__.py

1 of 1 shown

Make sure disabling CSRF protection is safe here.

Disabling CSRF protections is security-sensitive [python:S4502](#)

Status: **TO REVIEW**

This security hotspot needs to be reviewed to assess whether the code poses a risk.

[Change st:](#)

Where is the risk?

What's the risk?

Assess the risk

How

service/__init__.py

[Open in IDE](#)

```
1     ....  
2     Service Package  
3     ....  
4     from flask import Flask  
5  
6     app = Flask(__name__)  
  
7     # This must be imported after the Flask app is created  
8     from service import routes          # pylint:  
9     from service.common import log_handlers # pylint:  
10    log_handlers.init_logging(app, "gunicorn.error")  
11  
12
```

Make sure disabling CSRF protection is safe here.

Risk nerede? sekmesinin altında, kullandığımız örnek kodun uygun güvenlik önlemlerini içermediği için potansiyel bir Cross Site Request Forgery (CSRF) riski olduğunu belirtiyor.

Bir sonraki sorunuz “Bunu nasıl düzeltbilirim?” olabilir. Bunu öğrenmek için **Bunu nasıl düzeltbilirim?** sekmesine tıklayabilirsiniz.

Filters

Assigned to me All

Status

To review

Overall code

🛡 1 Security Hotspots to review

Review priority: **HIGH**

Cross-Site Request Forgery (CSRF)

1



Make sure disabling CSRF protection is safe here.

service/__init__.py

1 of 1 shown

For a [Flask](#) application,

- the `CSRFProtect` module should be used (and not disabled fu

```
app = Flask(__name__)
csrf = CSRFProtect()
csrf.init_app(app) # Compliant
```



- and it is recommended to not disable the CSRF protection on sp

```
@app.route('/example/', methods=['POST']) # Compliant
def example():
    return 'example'

class unprotectedForm(FlaskForm):
    class Meta:
        csrf = True # Compliant

    name = TextField('name')
    submit = SubmitField('submit')
```

See

- [OWASP Top 10 2021 Category A1 - Broken Access Control](#)
- [MITRE, CWE-352 - Cross-Site Request Forgery \(CSRF\)](#)
- [OWASP Top 10 2017 Category A6 - Security Misconfiguration](#)
- [OWASP: Cross-Site Request Forgery](#)
- [SANS Top 25 - Insecure Interaction Between Components](#)

Bunu nasıl düzeltelim? sekmesi, dikkat etmeniz gereken bazı noktaları veriyor.

Bu uygulama Flask framework'ü kullanılarak yazılmıştır. Flask ile ilgili bölümne indiğinizde, sorunu düzeltmek için `CSRFProtect` sınıfını nasıl kullanacağınızı tam olarak anlatıyor ve bazı diğer tavsiyeler veriyor.

```
app = Flask(__name__)
csrf = CSRFProtect()
csrf.init_app(app) # Compliant
```

Eğer bu sizin orijinal kodunuz olsaydı, önerilen değişiklikleri uygulamanız ve sorunun düzeltildiğinden emin olmak için taramayı tekrar çalıştırmanız gerekiyor.

Sonuç

Tebrikler! Statik analiz üzerinde bu laboratuvarı tamamladınız, bu da güvenli uygulama geliştirme sürecinin ayrılmaz bir adımındır. Artık uygulamalarınıza daha güvenli hale getirmek için onlara statik analiz güvenlik taramaları yapma yolunda ilerliyorsunuz.

Statik analizin bir projedeki zayıflıkları tespit etmek için nasıl kullanılabileceğini anlıyorsunuz. Ayrıca, statik analizler gerçekleştirmek için SonarQube gibi açık kaynak araçları nasıl başlayacağınızı da biliyorsunuz.

Sonraki Adımlar

Farklı türdeki zayıflıkları tespit etmek, güvenli uygulama geliştirmedeki ilk adımlardan biridir. Ayrıca, bu zayıflıkların arkasındaki anlamları anlamamanız gereklidir ki doğru eylemleri alabilin. Öğrenmenin en iyi yolu, uygulamaktır.

Bir sonraki meydan okumanız, geliştirme ortamınızda SonarQube'u kurmak, kodunuza güvenlik taramaları yapmak ve bulduğu sorunları düzeltmektir. Daha güvenli kod yazma yolunda ilerliyorsunuz!

Author(s)

[Roxanne Li](#)
[John J. Rofrano](#)

© IBM Corporation. Tüm hakları saklıdır.