Zafiyet Analizinin Değerlendirilmesi

Gerekli tahmini süre: 20 dakika

Giriş

Zafiyet Analizinin Değerlendirilmesi için uygulamalı laboratuvara hoş geldiniz.

Güvenli bir uygulama geliştirme görevi verildiğinde, yazılım geliştirme yaşam döngüsüne güvenliği entegre etmek önemlidir. Ancak, bu, uygulamanızın dağıtım zamanı geldiğinde zafiyetlerden tamamen arınmış olacağı anlamına gelmez. Bir zafiyet analizi yapmak, uygulamanız canlıya geçmeden önce içindeki zafiyetleri belirlemenize yardımcı olabilir.

Zafiyet analizi, bir sistem veya uygulamadaki olası bilinen zayıflıkların veya zafiyetlerin sistematik ve kapsamlı bir incelemesini içerir. Zafiyetlerin ortaya çıkarılması, sorunun giderilmesinde ve güvenli bir sistem geliştirilmesinde atılacak ilk adımdır.

Öğrenme Hedefleri

Bu laboratuvarı tamamladıktan sonra şunları yapabileceksiniz:

- Jake aracı kullanarak bir örnek web uygulamasında zafiyet taraması gerçekleştirmek
- Zafiyet taraması sonuçlarını değerlendirmek
- Zafiyet analizlerinin güvenli yazılım uygulamaları geliştirmedeki önemini açıklamak

Laboratuvar Ortamını Kurma

Laboratuvara başlamadan önce biraz hazırlık yapmanız gerekiyor. Aşağıdaki adımlarda, bir güvenlik açığı tarayıcısı ve test etmek için bir örnek web uygulaması indirecek ve kuracaksınız.

Bu laboratuvar için kullanacağınız uygulamalar şunlardır:

- Jake, SonaType Nexus Topluluğu tarafından geliştirilen bir güvenlik açığı tarayıcısı
- Hit Counter (bir örnek web uygulaması)

Bu yazılımların her ikisi de Docker görüntüleri olarak mevcuttur.

Jake Nedir?

Jake, topluluk tarafından oluşturulmuş, açık kaynaklı bir araçtır ve Python ortamlarındaki güvenlik açıklarını kontrol etmek için tasarlanmıştır. Jake, Sonatype tarafından oluşturulmamış veya desteklenmemiştir, ancak Sonatype OSS Index'i kullanmaktadır.

Hit Counter Nedir?

Hit Counter, bir Python Flask uygulamasıdır. Uzun zamandır güncellenmemiştir. Bu, güvenlik açıkları için iyi bir uygulama kontrolü yapmasını sağlar. Şu anda eski bir Flask 1.1.4 sürümünü kullanmakta ve bağımlılıkları, maalesef, yıllar içinde güvenlik açığına dönüşmüştür.

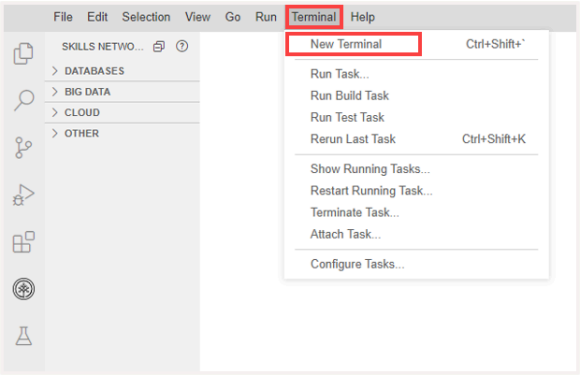
Adım 1: Jake aracını kurun

`jake` aracını Python paket yöneticisi (`pip`) kullanarak, komut satırı arayüzünde aşağıdaki komutu çalıştırarak kuracaksınız.

Göreviniz

1. Üst menüden yeni bir terminal açın: **Terminal Yeni Terminal** ve `cd` komutunu kullanarak `home/projects` dizinine gidin:

   ```
   cd /home/project
   ```



2. Python sanal ortam desteğini kurun:

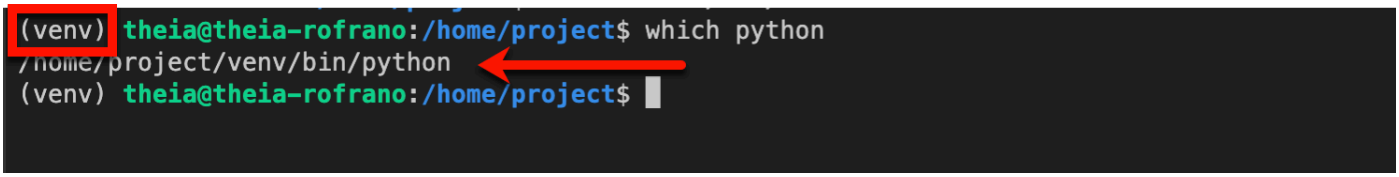
   ```
   sudo apt-get update && sudo apt-get install -y python3-venv
   ```

3. `venv` adında bir Python sanal ortamı oluşturun ve etkinleştirin:

   ```
   python3 -m venv venv
   source venv/bin/activate
   ```

4. Ortamın etkin olduğunu doğrulamak için `which python` komutunu çalıştırın:

   ```
   which python
   ```



İstemcinin önünde `(venv)` görmelisiniz ve `which python` komutu `/home/project/venv/bin/python` döndürmelidir. Her ikisini de görüyorsanız, her şey düzgün çalışıyor demektir.

5. `jake` aracını bu sanal ortama kurmak için aşağıdaki `pip install` komutunu çalıştırın:

```
pip install jake
```

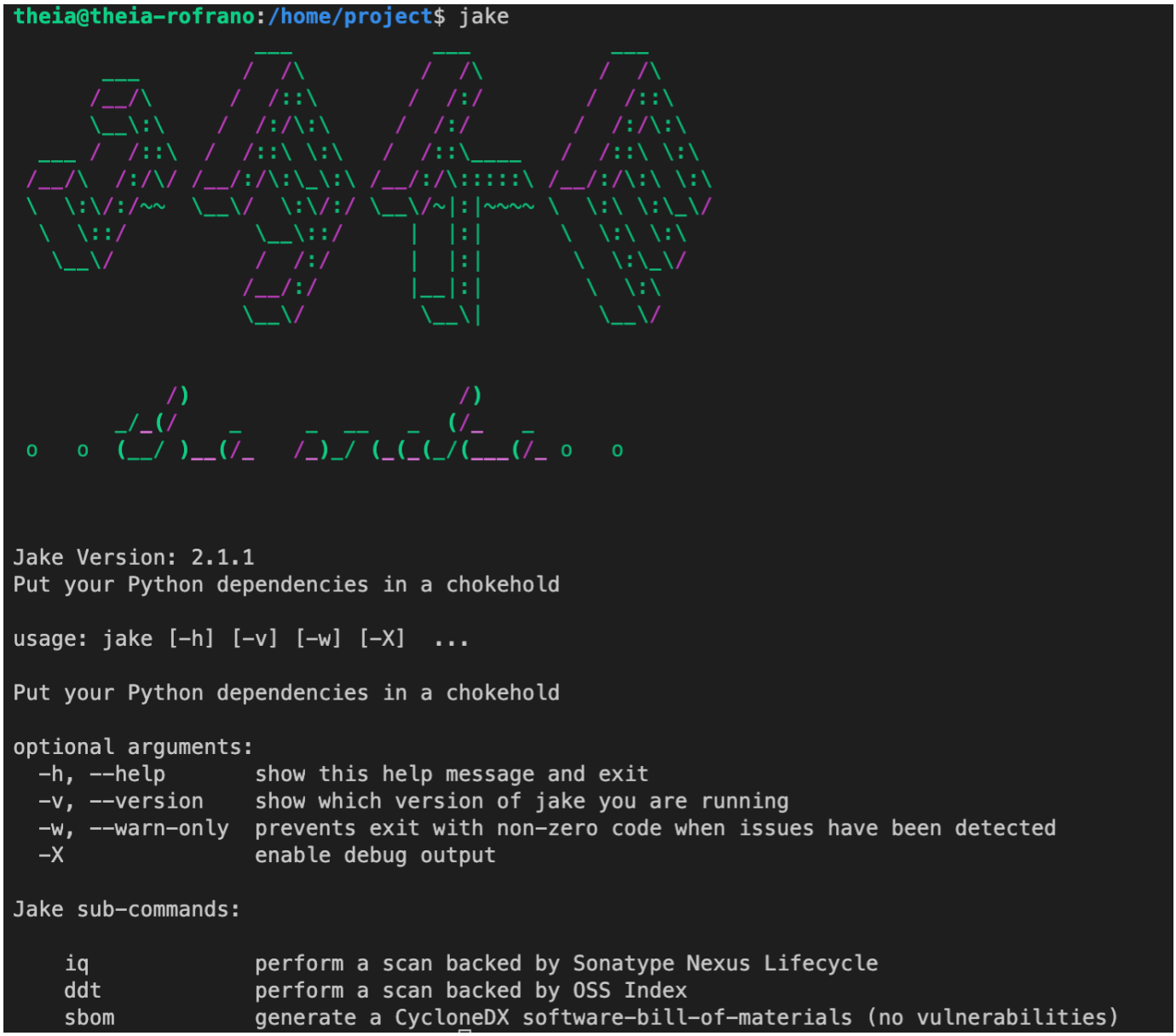
`jake`'in bağımlı olduğu çeşitli paketlerin kurulduğunu göreceksiniz.

6. `jake` aracını çalıştırarak düzgün çalıştığını kontrol edin:

```
jake
```

Sonuçlar

Aşağıdaki çıktıyı görmelisiniz:



Adım 2: Tarama için kodu yükleyin

Artık güvenlik açığı tarama aracınızı yüklediğinize göre, güvenlik açıklarını taramak için bir uygulamaya da ihtiyacınız olacak. **Hit Counter** adında bir Python Flask uygulaması kullanacaksınız. Eski bağımlılıkları olan bu uygulama, güvenlik açığı analizi için iyi bir örnek sunacaktır.

Göreviniz

Başlamak için, önce Docker imajını çekerek uygulamayı yükleyin.

1. Aşağıdaki `git clone` komutunu terminalde kullanarak kodu çekin:

```
git clone https://github.com/ibm-developer-skills-network/ycuer-flask-hitcounter.git
```

2. Ardından `ycuer-flask-hitcounter` klasörüne `cd` yapın.

```
cd ycuer-flask-hitcounter
```

3. Hit Counter için Python bağımlılıklarını `requirements.txt` dosyasından yüklemek için `pip install` komutunu kullanın:

```
pip install -r requirements.txt
```

Artık Hit Counter uygulamasını taramak için hazırsınız.

Adım 3: Bir Güvenlik Taraması Yapmak

Artık uygulamanın yerel bir kopyasına sahip olduğumuza göre, kod üzerinde `jake` komutunu çalıştırarak bulabileceğimiz güvenlik açıklarını görebiliriz. Jake birkaç modda çalışabilir. `ddt` modu, hedeflediğimiz zayıf bağımlılıkları bulmak için statik tarama yapar.

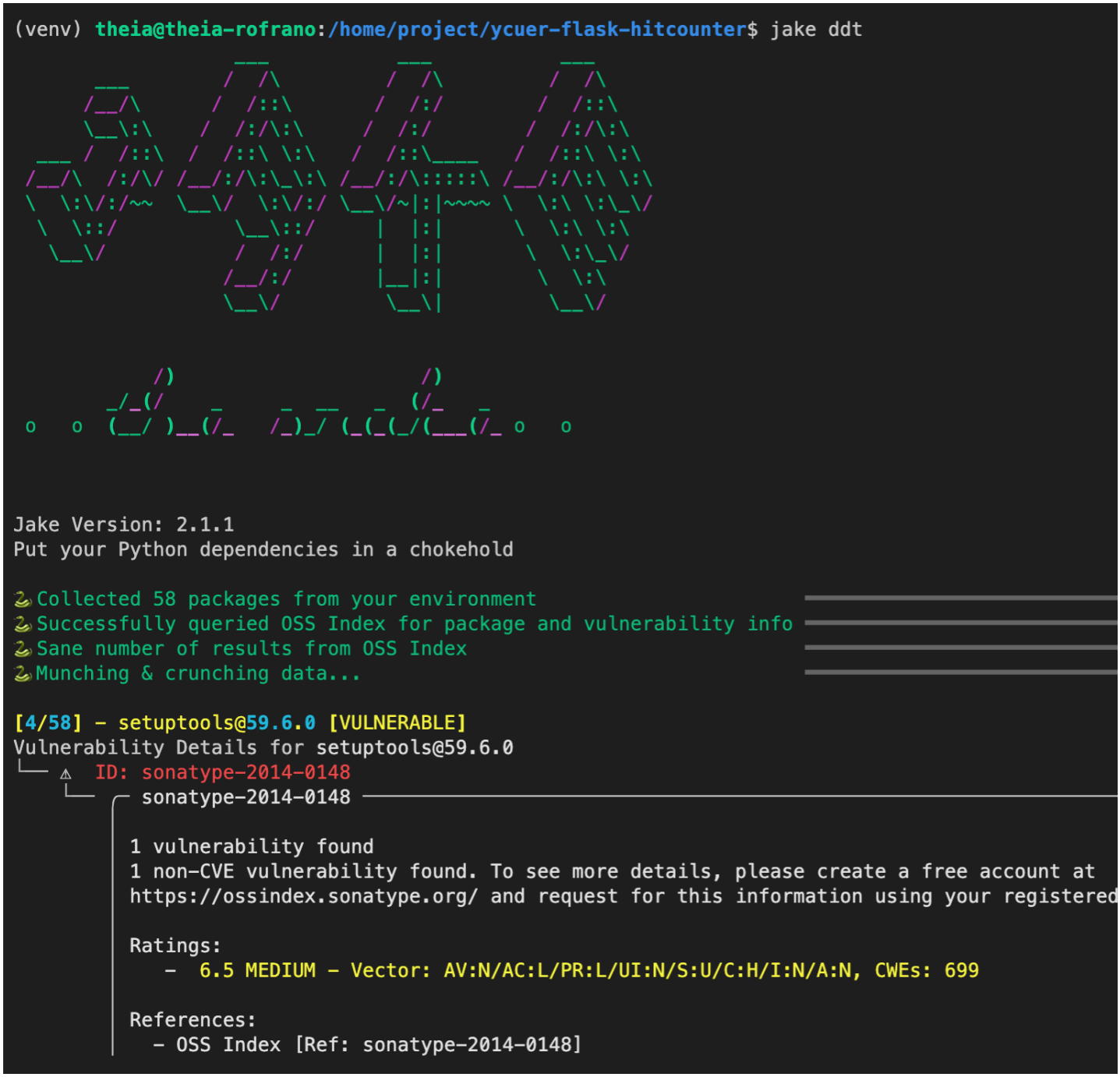
Göreviniz

1. `ycuer-flask-hitcounter` klasöründen `jake` komutunu çalıştırın:

```
jake ddt
```

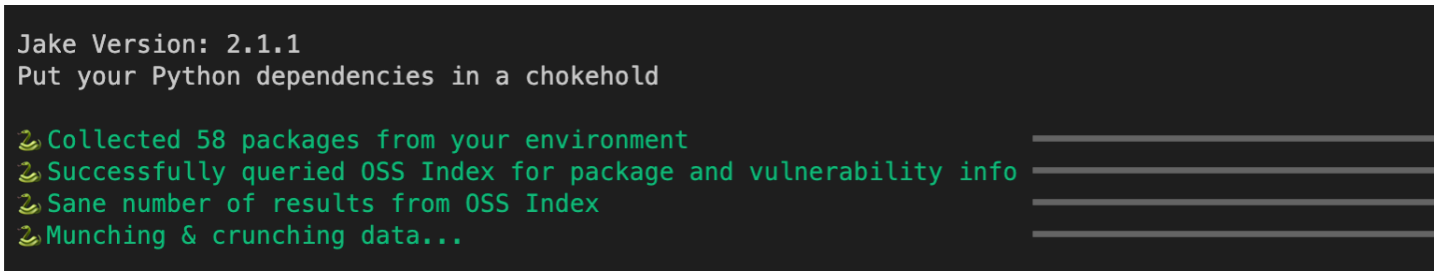
Sonuçlar

Çıktı, aşağıdaki görüntüye benzer olmalıdır. Benzer diyoruz çünkü zamanla daha fazla güvenlik açığı bulabilir. Bu laboratuvarı ne zaman çalıştırdığınıza bağlı olarak farklı sonuçlar alabilirsiniz. Bu, çıktının sadece en üst kısmıdır:

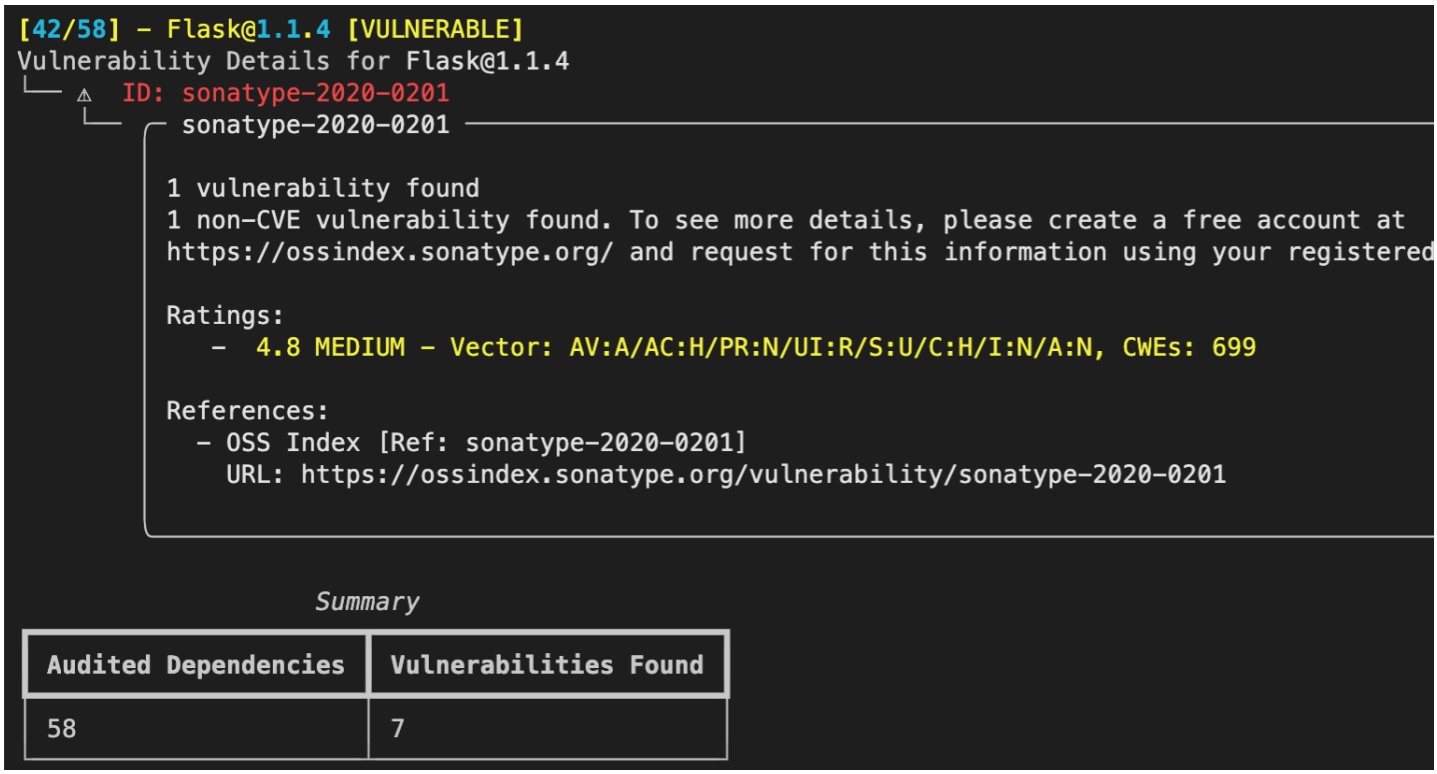


Adım 4: Sonuçların Yorumlanması

Tarama, bu uygulamanın doğrudan veya dolaylı olarak kullandığı 58 paket hakkında bilgi topladı ve birkaç güvenlik açığı ortaya çıkardı.



Daha sonra, güvenlik açığı olan paketlerin adını, güvenlik açığı hakkında detayları, güvenlik açığının ne olduğunu daha fazla araştırmak için URL'leri ve nasıl düzeltilebileceğine dair bilgileri içeren 7 güvenlik açığı hakkında detaylar sağladı. Çıktının sonuna bir örnek:



Gördüğünüz gibi, Flask 1.1.4 orta seviyede bir güvenlik açığına sahip, bu nedenle muhtemelen bu güvenlik açığına sahip olmayan daha yeni bir sürüme yükseltilmelidir. Bunu `requirements.txt` dosyasındaki sürümü güncelleyerek yapabilirsiniz. Daha fazla bilgi, detaylardaki URL'de bulunabilir. Ayrıca, listenin sonunda 58 paketten 7'sinin güvenlik açığı olduğunu gösteren bir özet bulunmaktadır.

Sonuç

Tebrikler! Bir web uygulamasında güvenlik açığı taraması yapma konusundaki ilk denemenizi tamamladınız. Güvenlik açıklarını taramak, güvenli uygulamalar geliştirmek için sahip olunması gereken önemli bir beceridir.

Bu laboratuvar çalışmasında, güvenlik tarama sürecinin kritik bir bileşeni olan güvenlik açığı analizi yapmayı öğrendiniz. Bu, güvenlik açığını tanımlamayı, analiz etmeyi, riskini değerlendirmeyi ve düzeltmeyi içeren yinelemeli bir süreçtir; ardından bu prosedürü tekrarlarsınız.

Sonraki Adımlar

Şimdi bu yeni bilgiyi uygulamaya koyma zamanı. Python uygulamalarınızda `jake` çalıştırarak bağımlılıklarınızdaki güvenlik açıklarını kontrol edin ve ortaya çıkardığınız güvenlik açıklarını düzeltmek için ek araştırmalar yapın. Çoğu zaman, sadece `requirements.txt` dosyanızı güvenli olmayan daha yeni sürümlerle güncellemeniz yeterlidir.

Author

[John J. Rofrano]
[David Pasternak]

Other Contributor(s)

Sam Propochuk

Michelle R. Sanchez, 25 yılı aşkın kurumsal düzeyde teknik eğitim ve destek deneyimi olan Skill-up Technologies'de Eğitim Tasarımcısı.