Command Injection

Task 1 Introduction (What is Command Injection?) (Görev 1 Giriş (Komut Enjeksiyonu Nedir?))

Bu odada, komut enjeksiyonu olan web güvenlik açığını ele alacağız. Bu güvenlik açığının ne olduğunu anladıktan sonra, etkisini ve bir uygulama üzerinde yarattığı riski göstereceğiz.

Daha sonra, bu bilgiyi uygulamaya koyabileceksiniz, yani:

- Komut enjeksiyonu güvenlik açığı nasıl keşfedilir
- Farklı işletim sistemleri için tasarlanmış yükler kullanılarak bu güvenlik açığının nasıl test edileceği ve istismar edileceği
- Bir uygulamada bu güvenlik açığı nasıl önlenir
- Son olarak, odanın sonundaki bir uygulamada teoriyi pratik öğrenmeye uygulayacaksınız.

Başlangıç olarak, öncelikle komut enjeksiyonunun ne olduğunu anlayalım. Komut enjeksiyonu, bir cihazdaki uygulamanın çalıştığı ayrıcalıkların aynısını kullanarak işletim sistemi üzerinde komutlar çalıştırmak için bir uygulamanın davranışının kötüye kullanılmasıdır. Örneğin, joe adlı bir kullanıcı olarak çalışan bir web sunucusunda komut enjeksiyonu gerçekleştirmek, bu joe kullanıcısı altında komutları yürütecek ve dolayısıyla joe'nun sahip olduğu tüm izinleri alacaktır.

Komut enjeksiyonu, bir uygulama içinde uzaktan kod yürütme yeteneği nedeniyle genellikle "Uzaktan Kod Yürütme" (RCE) olarak da bilinir. Bu güvenlik açıkları genellikle bir saldırgan için en kazançlı olanlardır çünkü saldırganın savunmasız sistemle doğrudan etkileşime girebileceği anlamına gelir. Örneğin, bir saldırgan sistem veya kullanıcı dosyalarını, verileri ve bu türden şeyleri okuyabilir.

Örneğin, uygulamanın hangi kullanıcı hesabını çalıştırdığını listelemek için whoami komutunu gerçekleştirmek üzere bir uygulamayı kötüye kullanabilmek bir komut enjeksiyonu örneği olacaktır.

Komut enjeksiyonu, Contrast Security'nin AppSec istihbarat raporu tarafından 2019'da bildirilen ilk on güvenlik açığından biriydi. (Contrast Security AppSec., 2019). Dahası, OWASP çerçevesi sürekli olarak bu nitelikteki güvenlik açıklarını bir web uygulamasının ilk on güvenlik açığından biri olarak önermektedir (OWASP çerçevesi).

Soru ⇒ Burayı okuyun.

Cevap ⇒ Cevap Gerekmemektedir.

Task 2 Discovering Command Injection (Görev 2 Komut Enjeksiyonunu Keşfetme)

Bu güvenlik açığı, uygulamaların genellikle PHP, Python ve NodeJS gibi programlama dillerindeki işlevleri kullanarak makinenin işletim sistemine veri aktarması ve sistem çağrıları yapması nedeniyle ortaya çıkmaktadır. Örneğin, bir alandan girdi almak ve bir dosyaya giriş aramak gibi. Örnek olarak aşağıdaki kod parçasını ele alalım:

Bu kod parçasında uygulama, kullanıcının $title adlı bir giriş alanına girdiği verileri alarak bir dizinde şarkı adı arıyor. Bunu birkaç basit adıma ayıralım.

```php
<?php

$songs = "/var/www/html/songs"

if (isset $_GET["title"])) {
    $title = $_GET["title"];

    $command = "grep $title /var/www/html/songtitle.txt";

    $search = exec($command);
    if ($search == "") {
        $return = "<p>The requested song</p><p> $title does </p><b>not</b><p> exist!</p>";
    } else {
        $return = "<p>The requested song</p><p> $title does </p><b>exist!</b>";
      }

     echo $return;
}

?>
```

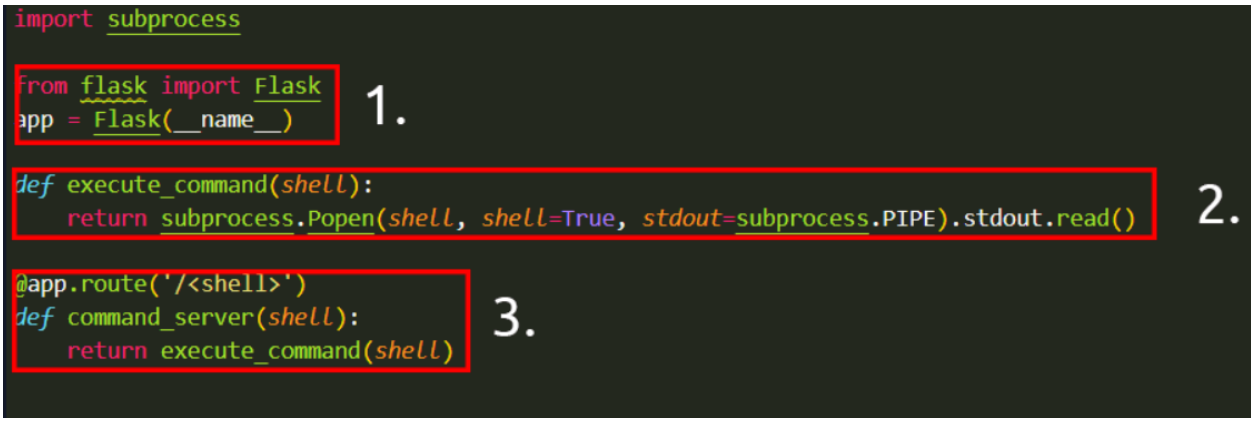
1. Uygulama MP3 dosyalarını işletim sisteminde bulunan bir dizinde saklar.

2. Kullanıcı aramak istediği şarkı adını girer. Uygulama bu girdiyi $title değişkenine kaydeder.
3. Bu $title değişkeni içindeki veriler grep komutuna aktarılarak songtitle.txt adlı bir metin dosyasında kullanıcının aramak istediği şeyin girdisi aranır.
4. Songtitle.txt dosyasında yapılan bu aramanın çıktısı, uygulamanın kullanıcıya şarkının var olup olmadığını bildirip bildirmeyeceğini belirleyecektir.

Şimdi, bu tür bilgiler tipik olarak bir veritabanında saklanır; ancak bu, bir uygulamanın uygulamanın işletim sistemiyle etkileşim kurmak için kullanıcıdan girdi aldığı durumlara sadece bir örnektir.

Bir saldırgan, uygulamanın yürütmesi için kendi komutlarını enjekte ederek bu uygulamayı kötüye kullanabilir. Songtitle.txt dosyasında bir girdi aramak için grep kullanmak yerine, uygulamadan daha hassas bir dosyadan veri okumasını isteyebilirler.

Uygulamaların bu şekilde kötüye kullanılması, uygulamanın kullandığı programlama dili ne olursa olsun mümkün olabilir. Uygulama işlediği ve yürüttüğü sürece komut enjeksiyonuna neden olabilir. Örneğin, aşağıdaki kod parçacığı Python ile yazılmış bir uygulamadır.



Unutmayın, bu uygulamaların arkasındaki sözdizimini anlamanız beklenmiyor. Ancak, mantıklı olması açısından, bu Python uygulamasının nasıl çalıştığına dair adımları da özetledim.

1. "flask" paketi bir web sunucusu kurmak için kullanılır

2. Cihaz üzerinde bir komut çalıştırmak için "subprocess" paketini kullanan bir fonksiyon
3. Web sunucusunda, sağlanan her şeyi çalıştıracak bir rota kullanırız. Örneğin, whoami'yi çalıştırmak için http://flaskapp.thm/whoami adresini ziyaret etmemiz gerekir

Sorular

Soru ⇒ Bu görevdeki PHP kod parçacığında kullanıcının girdisini hangi değişken depolar?

Cevap ⇒ $title

Soru ⇒ PHP kod parçacığında bir kullanıcı tarafından gönderilen verileri almak için hangi HTTP yöntemi kullanılır?

Cevap ⇒ GET

Soru ⇒ Python kod parçacığındaki id komutunu çalıştırmak isteseydim, hangi rotayı ziyaret etmem gerekirdi?

Cevap ⇒ /id

Task 3 Exploiting Command Injection (Görev 3 Komut Enjeksiyonundan Yararlanma)

Bu odanın uygulamalı oturumunda göreceğiniz gibi, komut enjeksiyonunun gerçekleşip gerçekleşmeyeceğini genellikle bir uygulamanın davranışlarından belirleyebilirsiniz.

Sistem komutlarını verilerle doldurmak için kullanıcı girdisini kullanan uygulamalar genellikle istenmeyen davranışlarla birleştirilebilir. Örneğin, kabuk operatörleri ;, & ve && iki (veya daha fazla) sistem komutunu birleştirecek ve her ikisini de yürütecektir. Bu kavrama aşina değilseniz, bu konuda daha fazla bilgi edinmek için Linux temelleri modülüne göz atmanız faydalı olacaktır.

Komut Enjeksiyonu çoğunlukla iki yoldan biriyle tespit edilebilir:

1. Blind (Kör) komut enjeksiyonu

2. Verbose (Ayrıntılı) komut enjeksiyonu

Bu iki yöntemi aşağıdaki tabloda tanımladım, alttaki iki bölüm bunları daha ayrıntılı olarak açıklayacaktır.

Method	Açıklama
Blind	Bu tür enjeksiyon, yükleri test ederken uygulamadan doğrudan çıktı alınamadığı durumlardır. Yükünüzün başarılı olup olmadığını belirlemek için uygulamanın davranışlarını araştırmanız gerekecektir.
Verbose	Bu tür enjeksiyon, bir yükü test ettikten sonra uygulamadan doğrudan geri bildirim alınmasıdır. Örneğin, uygulamanın hangi kullanıcı altında çalıştığını görmek için whoami komutunu çalıştırmak. Web uygulaması kullanıcı adını doğrudan sayfaya çıktılayacaktır.

Kör Komut Enjeksiyonunu Algılama

Kör komut enjeksiyonu, komut enjeksiyonunun gerçekleştiği, ancak görünür bir çıktı olmadığı, dolayısıyla hemen fark edilmediği durumdur. Örneğin, bir komut çalıştırılır ancak web uygulaması hiçbir mesaj çıktısı vermez.

Bu tür bir komut enjeksiyonu için, biraz zaman gecikmesine neden olacak yükler kullanmamız gerekecektir. Örneğin, ping ve sleep komutları test etmek için önemli yüklerdir. Örnek olarak ping kullanıldığında, uygulama belirttiğiniz ping sayısına bağlı olarak x saniye boyunca askıda kalacaktır.

Kör komut enjeksiyonunu tespit etmenin bir başka yöntemi de bazı çıktıları zorlamaktır. Bu, > gibi yönlendirme operatörleri kullanılarak yapılabilir. Eğer bu konuda bilginiz yoksa Linux temelleri modülüne göz atmanızı tavsiye ederim. Örneğin, web uygulamasına whoami gibi komutları çalıştırmasını ve bunu bir dosyaya yönlendirmesini söyleyebiliriz. Daha sonra yeni oluşturulan bu dosyanın içeriğini okumak için cat gibi bir komut kullanabiliriz.

Komut enjeksiyonunu bu şekilde test etmek genellikle karmaşıktır ve komutların sözdizimi Linux ve Windows arasında farklılık gösterdiğinden oldukça fazla deneme gerektirir.

curl komutu, komut enjeksiyonunu test etmek için harika bir yoldur. Bunun nedeni, yükünüzde bir uygulamaya ve uygulamadan veri iletmek için curl kullanabilmenizdir. Örnek olarak aşağıdaki kod parçasını ele alalım, komut enjeksiyonu için bir uygulamaya basit bir curl yükü göndermek mümkündür.

```
curl http://vulnerable.app/process.php%3Fsearch%3DThe%20Beatles%3B%20whoami
```

Verbose Komut Enjeksiyonunu Algılama

Komut enjeksiyonunu bu şekilde tespit etmek muhtemelen ikisi arasında en kolay yöntemdir. Verbose komut enjeksiyonu, uygulamanın size neler olduğuna veya yürütüldüğüne dair geri bildirim veya çıktı vermesidir.

Örneğin, ping veya whoami gibi komutların çıktısı doğrudan web uygulamasında görüntülenir.

Useful payloads (Faydalı yükler)

Hem Linux hem de Windows için bazı değerli yükleri aşağıdaki tablolarda derledim.

Linux

Payload	**Açıklama**
whoami	Uygulamanın hangi kullanıcı altında çalıştığını görün.
ls	Geçerli dizinin içeriğini listeler. Yapılandırma dosyaları, ortam dosyaları (belirteçler ve uygulama anahtarları) ve daha birçok değerli şey gibi dosyaları bulabilirsiniz.
ping	Bu komut uygulamayı askıda kalmaya çağıracaktır. Bu, kör komut enjeksiyonu için bir uygulamayı test ederken faydalı olacaktır.
sleep	Bu, makinede ping yüklü olmadığı durumlarda kör komut enjeksiyonu için bir uygulamayı test etmede başka bir yararlı yüktür.
nc	Netcat, savunmasız uygulama üzerinde bir ters kabuk oluşturmak için kullanılabilir. Bu dayanak noktasını kullanarak hedef makinede diğer hizmetler, dosyalar veya ayrıcalıkları artırmanın olası yolları için gezinebilirsiniz.

Windows

Payload	**Açıklama**
whoami	Uygulamanın hangi kullanıcı altında çalıştığını görün.
dir	Geçerli dizinin içeriğini listeler. Yapılandırma dosyaları, ortam dosyaları (belirteçler ve uygulama anahtarları) ve daha birçok değerli şey gibi dosyaları bulabilirsiniz.

ping	Bu komut uygulamayı askıda kalmaya çağıracaktır. Bu, kör komut enjeksiyonu için bir uygulamayı test ederken faydalı olacaktır.
timeout	Bu komut aynı zamanda uygulamayı askıda kalmaya çağıracaktır. Ping komutu yüklü değilse, kör komut enjeksiyonu için bir uygulamayı test etmek için de kullanışlıdır.

Sorular

Soru ⇒ Uygulamanın hangi kullanıcı olarak çalıştığını belirlemek isteseydim hangi yükü kullanırdım?

Cevap ⇒ whoami

Soru ⇒ Bir Linux makinesinde kör komut enjeksiyonunu test etmek için hangi popüler ağ aracını kullanabilirim?

Cevap ⇒ ping

Soru ⇒ Bir Windows makinesini kör komut enjeksiyonuna karşı test etmek için hangi yükü kullanabilirim?

Cevap ⇒ timeout

Task 4 Remediating Command Injection (Görev 4 Komut Enjeksiyonunu Düzeltme)

Komut enjeksiyonu çeşitli yollarla önlenebilir. Bir programlama dilindeki potansiyel olarak tehlikeli işlevlerin veya kütüphanelerin minimum kullanımından, kullanıcının girdisine güvenmeden girdiyi filtrelemeye kadar her şey. Bunları aşağıda biraz daha detaylandırdım. Aşağıdaki örnekler PHP programlama diline aittir; ancak, aynı ilkeler diğer birçok dile genişletilebilir.

Vulnerable Functions (Hassas Fonksiyonlar)

PHP'de birçok işlev kabuk aracılığıyla komutları çalıştırmak için işletim sistemiyle etkileşime girer; bunlar şunları içerir:

- Exec
- Passthru

- System

Örnek olarak aşağıdaki snippet'i ele alalım. Burada, uygulama yalnızca forma girilen numaraları kabul edecek ve işleyecektir. Bu, whoami gibi herhangi bir komutun işlenmeyeceği anlamına gelir.

```
<input type="text" id="ping" name="ping" pattern="[0-9]+"></input> 1.
<?php
echo passthru("/bin/ping -c 4 "$_GET["ping"].'); 2.

?>
```

1. Uygulama yalnızca belirli bir karakter kalıbını kabul edecektir (0-9 rakamları)
2. Uygulama daha sonra yalnızca tamamı sayısal olan bu verileri yürütmeye devam edecektir.

Bu işlevler bir dize veya kullanıcı verisi gibi girdileri alır ve sistemde sağlanan her şeyi çalıştırır. Bu işlevleri uygun kontroller olmadan kullanan herhangi bir uygulama komut enjeksiyonuna karşı savunmasız olacaktır.

Input sanitisation (Girdi sterilizasyonu)

Bir uygulamanın kullandığı kullanıcı girdilerini sanitize etmek, komut enjeksiyonunu önlemenin harika bir yoludur. Bu, bir kullanıcının gönderebileceği veri biçimlerini veya türlerini belirleme işlemidir. Örneğin, yalnızca sayısal verileri kabul eden veya > , & ve / gibi özel karakterleri kaldıran bir giriş alanı.

Aşağıdaki kod parçasında, filter_input PHP işlevi, bir girdi formu aracılığıyla gönderilen herhangi bir verinin sayı olup olmadığını kontrol etmek için kullanılır. Eğer sayı değilse, geçersiz bir girdi olmalıdır.

```
<?php

if (!filter_input(INPUT_GET, "number", FILTER_VALIDATE_NUMBER)) {

}
```

Bypassing Filters (Baypass filtreleri)

Uygulamalar, bir kullanıcının girdisinden alınan verileri filtrelemek ve sterilize etmek için çok sayıda teknik kullanacaktır. Bu filtreler sizi belirli yüklerle kısıtlayacaktır; ancak, bu filtreleri atlamak için bir uygulamanın arkasındaki mantığı kötüye kullanabiliriz. Örneğin, bir uygulama tırnak işaretlerini çıkarabilir; bunun yerine aynı sonucu elde etmek için bunun onaltılık değerini kullanabiliriz.

Çalıştırıldığında, verilen veriler beklenenden farklı bir formatta olsa da, yine de yorumlanabilir ve aynı sonucu verir.

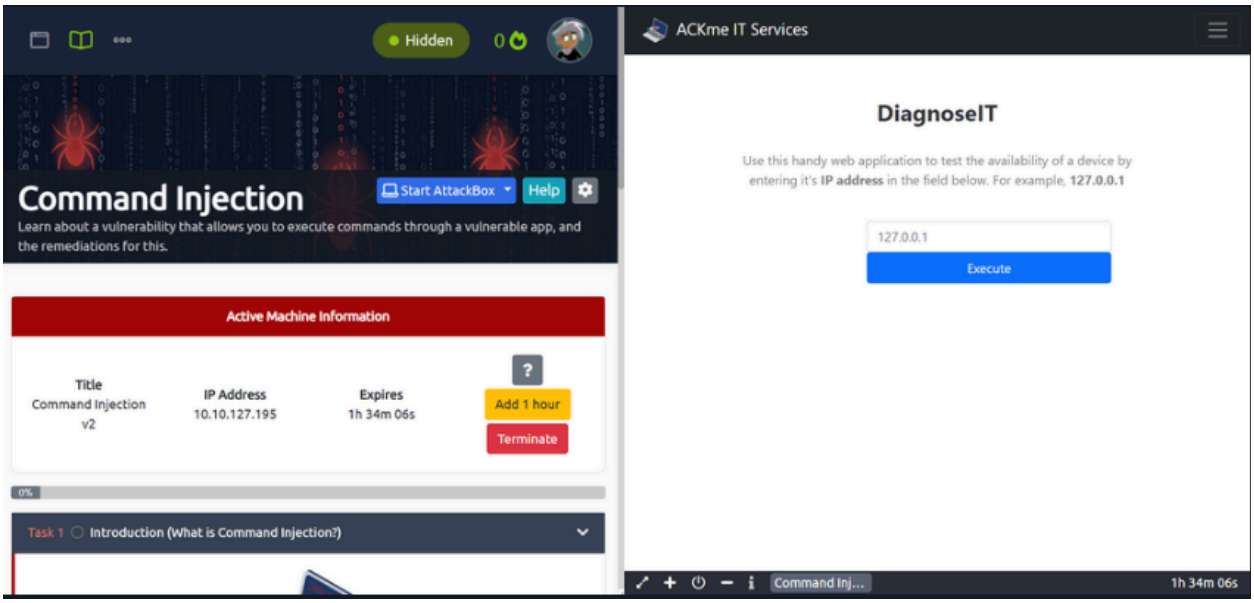
```
$payload = "\x2f\x65\x74\x63\x2f\x70\x61\x73\x73\x77\x64"
```

Soru ⇒ Bir uygulamaya sağlanan kullanıcı girdisini "temizleme" işlemi için kullanılan terim nedir?

Cevap ⇒ sanitisation

Task 5 Practical: Command Injection (Deploy) (Görev 5 Pratik: Komut Enjeksiyonu (Deploy))

Bu göreve bağlı makineyi dağıtın; hazır olduğunda bölünmüş ekran görünümünde görünür olacaktır.



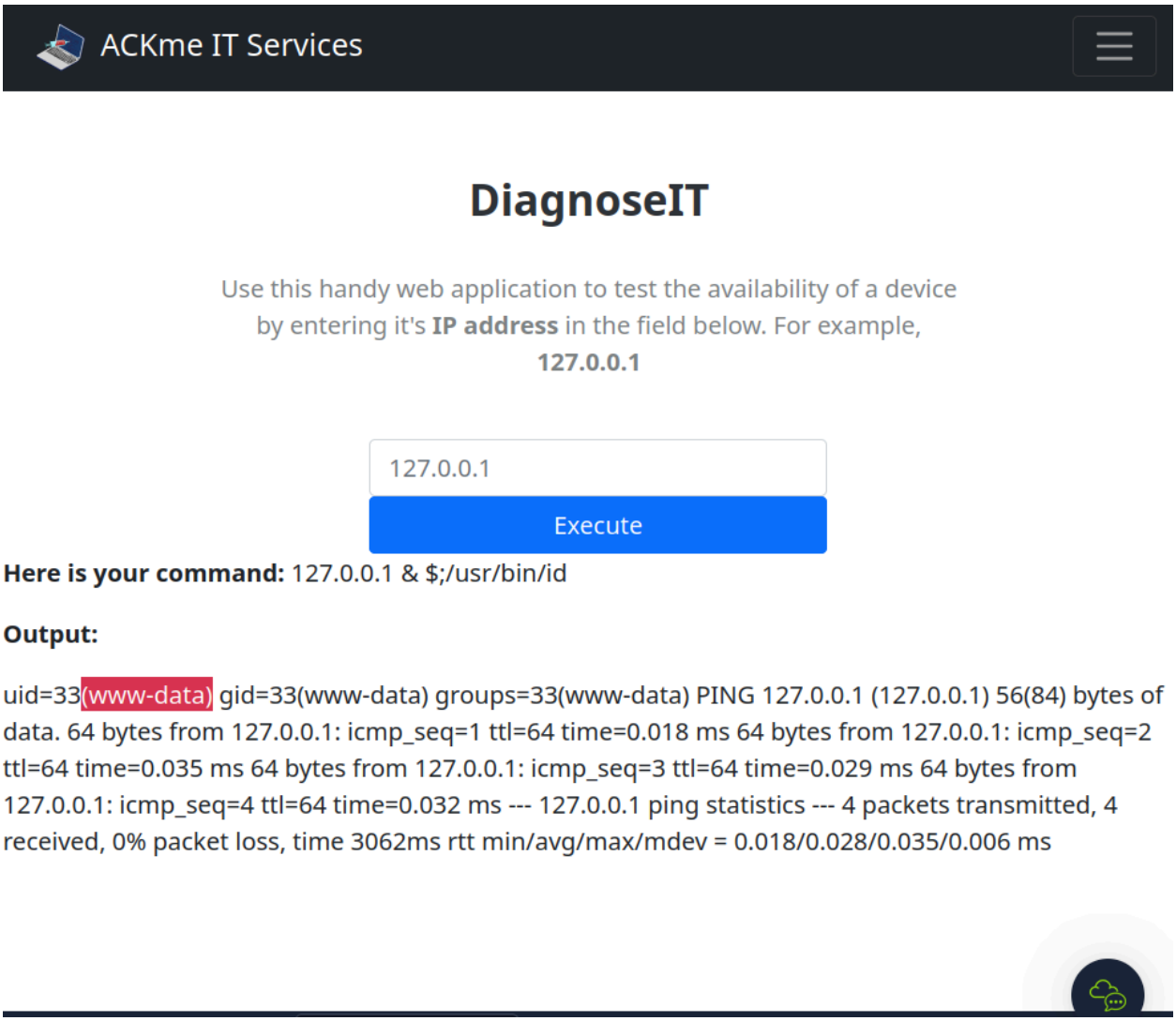
Komut enjeksiyonunu test etmek için bölünmüş ekran görünümünde görünen web sitesinde barındırılan uygulamada bazı yükleri test edin. Takıldığınız veya daha karmaşık yükleri keşfetmek istediğiniz durumlarda bu kopya sayfasına başvurun.

/home/tryhackme/flag.txt içinde bulunan bayrağın içeriğini bulun. Bunu başarmak için çeşitli yükler kullanabilirsiniz - birden fazla denemenizi tavsiye ederim.

Sorular

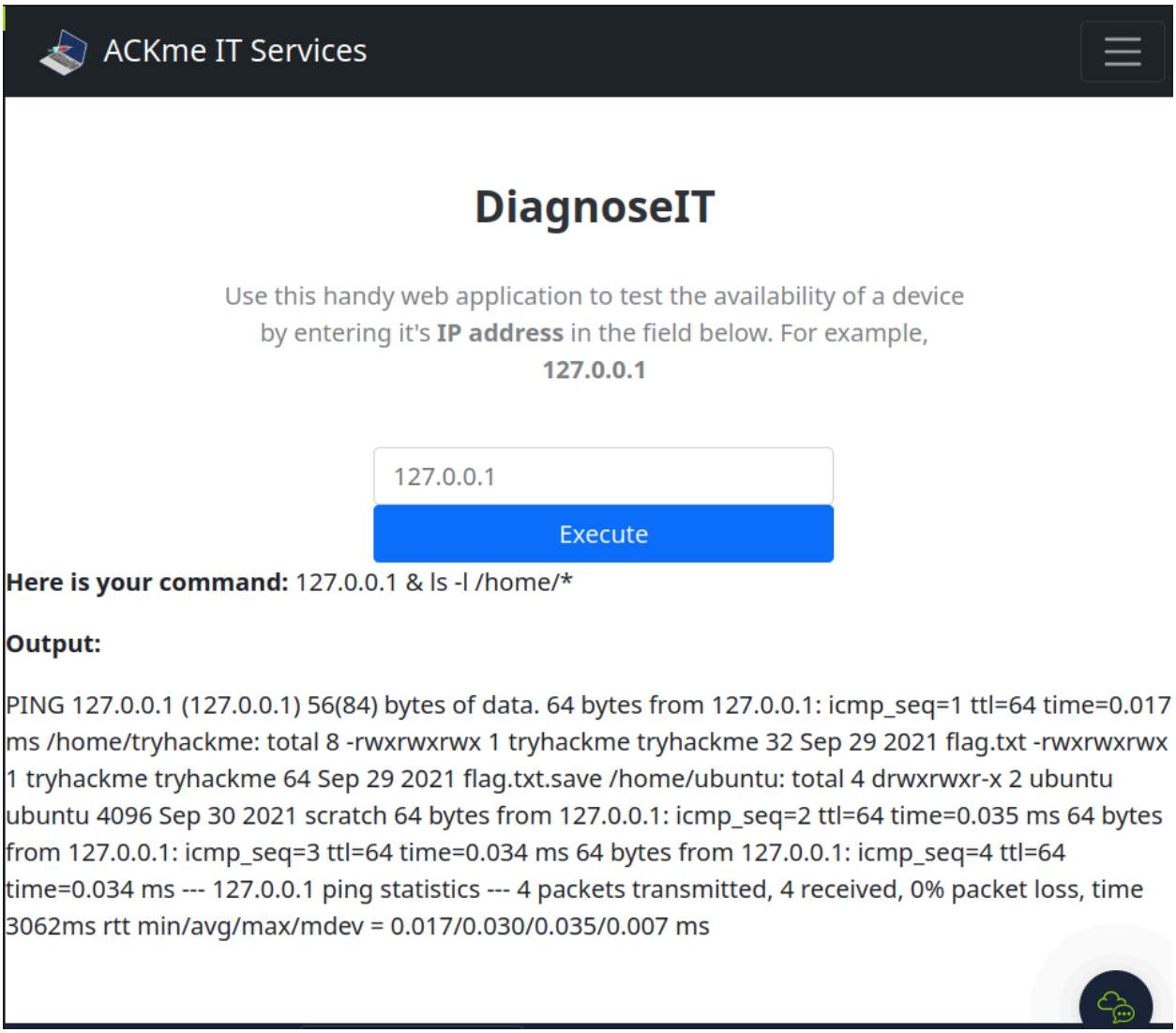
Soru ⇒ Bu uygulama hangi kullanıcı olarak çalışıyor?

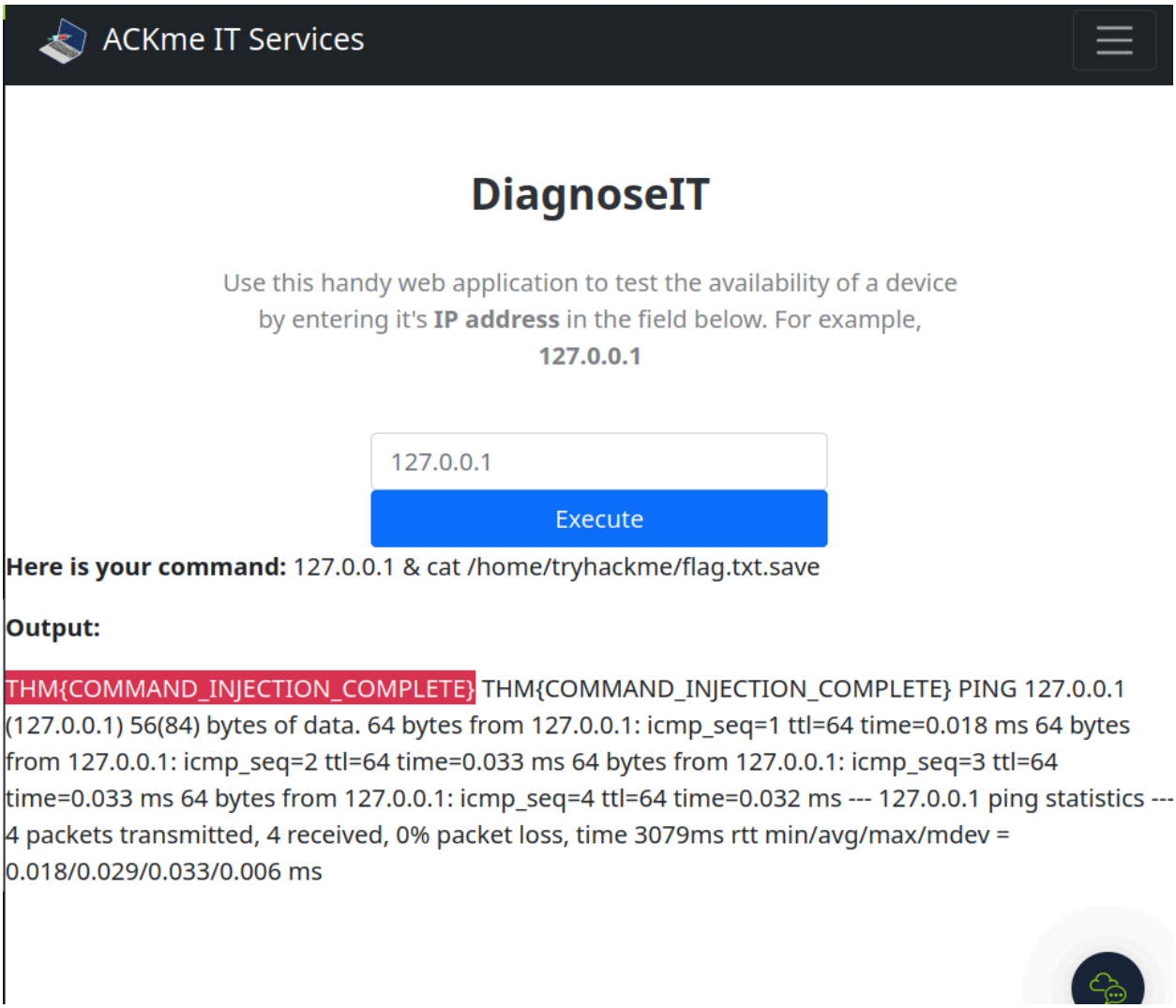
Cevap ⇒ www-data



Soru ⇒ /home/tryhackme/flag.txt içinde bulunan bayrağın içeriği nedir?

Cevap ⇒ THM{COMMAND_INJECTION_COMPLETE}


ACKme IT Services
DiagnoseIT
Use this handy web application to test the availability of a device by entering it's IP address in the field below. For example, 127.0.0.1
127.0.0.1
Execute
Here is your command: 127.0.0.1 & ls -l /home/*
Output:
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data. 64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.017 ms /home/tryhackme: total 8 -rwxrwxrwx 1 tryhackme tryhackme 32 Sep 29 2021 flag.txt -rwxrwxrwx 1 tryhackme tryhackme 64 Sep 29 2021 flag.txt.save /home/ubuntu: total 4 drwxrwxr-x 2 ubuntu ubuntu 4096 Sep 30 2021 scratch 64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.035 ms 64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.034 ms 64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.034 ms --- 127.0.0.1 ping statistics --- 4 packets transmitted, 4 received, 0% packet loss, time 3062ms rtt min/avg/max/mdev = 0.017/0.030/0.035/0.007 ms



Task 6 Conclusion (Görev 6 Sonuç)

Bu odanın sonuna kadar gelebildiğiniz için tebrikler. Özetlemek gerekirse, komut enjeksiyonunun aşağıdaki unsurlarını öğrendik:

- Komut enjeksiyonu güvenlik açığı nasıl keşfedilir
- Farklı işletim sistemleri için tasarlanmış yükler kullanılarak bu güvenlik açığının nasıl test edileceği ve istismar edileceği
- Bir uygulamada bu güvenlik açığı nasıl önlenir

- Pratik bir uygulamada komut enjeksiyonu gerçekleştirerek öğrendiklerinizi uygulamak

Muhtemelen keşfetmiş olacağınız gibi, aynı hedefe ulaşmak için kullanılabilecek birden fazla yük vardır. Bu görevin pratik unsuruna geri dönmenizi ve bayrağı almak için bazı alternatif yöntemler denemenizi şiddetle tavsiye ederim.

soru ⇒ Savunmasız makineyi görev 5'ten sonlandırın

Cevap ⇒ Cevap Gerekmemektedir.