|  |
| --- |
| ***SQL Injection*** |
| Kaan Efe Öğüt  *ADLİ BİLİŞİM MÜHENDİSLİĞİ* |
| bWAPP & Web for Pentester web uygulamaları üzerinde bulunan SQL İnjection zafiyetlerine birlikte sızıp kullanımlarını göreceğiz. |

***SQL***

**28.07.2021**

-Herhangi bir veritabanı üzerinde kullanılan dildir.SQL ile yalnızca veri tabanı üzerinde işlem yapılabilir.

-Veri tabanlarında bulunan sistemlere bilgi ekleme,bilgi değiştirme,bilgi çıkarma ve bilgi sorgulama işlemleri için kullanılabilir.

-SQL Veritabanları üzerinde “Select-From-Where-Union-Order By-Group By” gibi komutlar üzerinden işlem yapılmaktadır.

***SQL INJECTION***

-Manuel veya tool üzerinden işlem yapılarak veritabanına gönderilen SQL sorgularına kod enjekte etmeye dayalı bir zafiyet türüdür.

-Backend veri deposunu hedef alır.

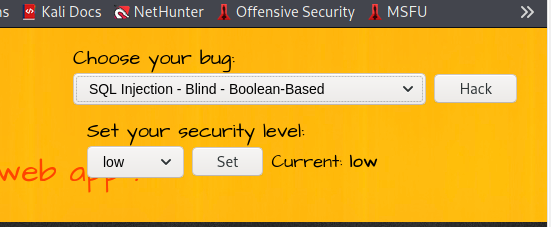
-Uygulamanın veritabanına gönderdiği sorgular değiştirilerek,sistemdeki veritabanları üzerinde ki tüm bilgiler okunabilir,değiştirilebilir veya silinebilir.

-Genellikle üye giriş panellerinde ve sayfa linklerinde karşımıza çıkar.Siteye girildiğinde veritabanından herhangi bir bilgide veya bölümde bulunabilir.

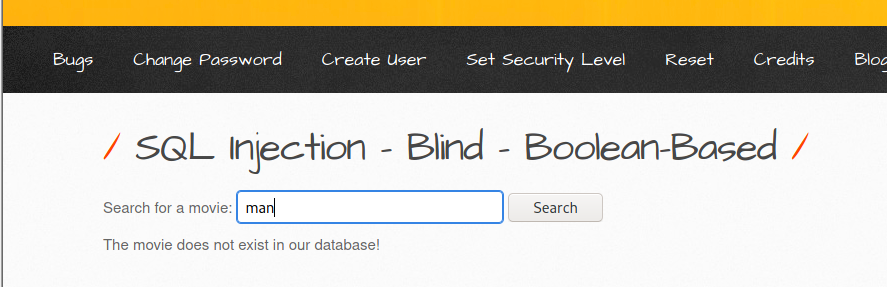
-Daha fazla teorik bilgi vermeden bWAPP ve WFP üzerinden uygulama gerçekleştirmek istiyorum.Ardından ise SQL Injection zafiyetlerinden nasıl korunabileceğimiz hakkında bilgi vereceğim.

-Öncelikle bWAPP üzerinden yapacağım 2 adet uygulama ile başlamak istiyorum.Bu yapacağımız uygulamalar SQLİ çalışma mantığını anlamak için olacaktır.

**Uygulama 1 :** İlk uygulama Blind SQL üzerine olacaktır.



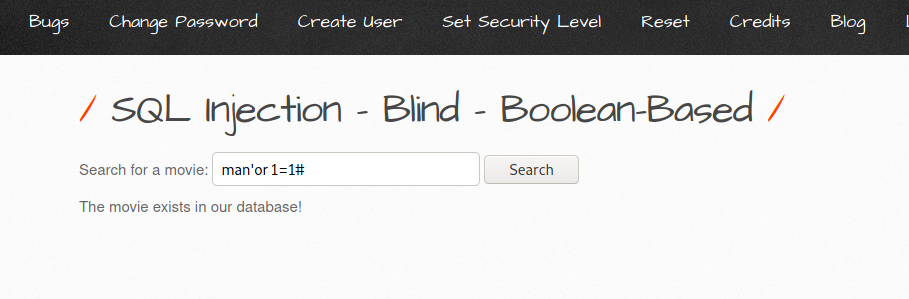
-Öncelikle bWAPP üzerinden zafiyetimi başlatıyorum.Bu zafiyet içerisinde film database’i bulunduran bir veritabanı içeriyor.



-Arama motoru üzerinde ilk önce veritabanından bir film yakalamaya çalışıyorum.

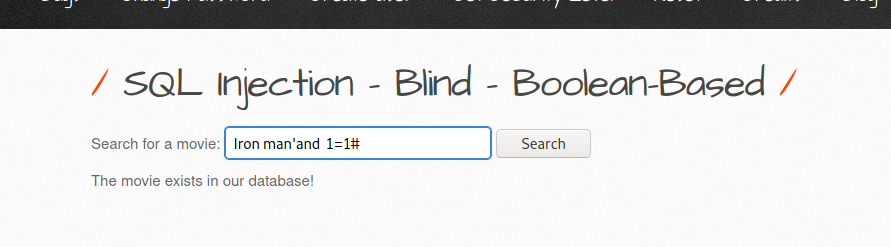


-Büyük küçük harfe duyarlı bir arama motoru olduğunu ve Bool sistemiyle çalıştığını öğreniyorum.



-Veritabanı üzerinde bir adet False bir adet ise True döndüren değeri bulduktan sonra veritabanı üzerinde payload sorgulamalarına başlıyorum.(‘)Kesme işareti ile form dışına çıkıp ikinci aramayı gerçekleştirebiliyorum.

-OR komutu ile işlem yaptığımda biri doğru olmasına rağmen veritabanında yok hatası alıyorum.



-İki adet doğru değer giriyorum ve veritabanında var diyip True döndürüyor.

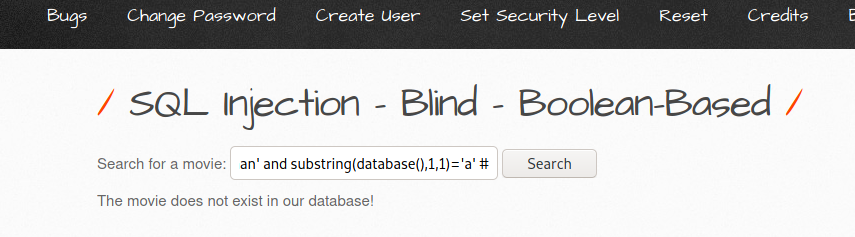
-Substring işlemlerine girmeden önce basit bir şekilde çalışma mantığını göstermek istiyorum.

substring('Blackbox',1,1) = 1. karakterden başla ve 1 değer al.- B

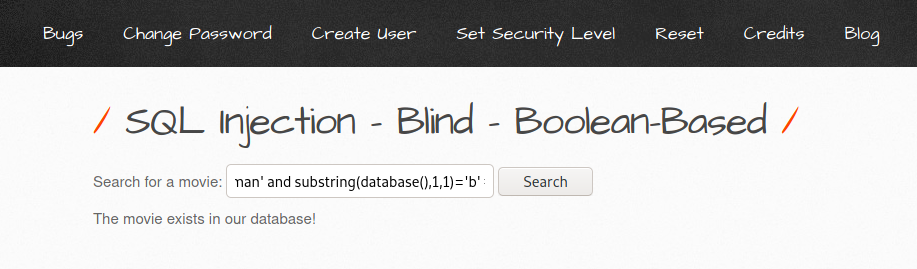
substring('Blackbox',2,1) = 2. karakterden başla ve bir değer al- L

substring('Blackbox',1,2) =1. Karakterden başla iki değer al -BL

substring('Blackbox',3,3) = 3.Karakterden başla 3 değer al -ACK



-Ardından Substring() fonksiyonu ile ifadeyi bölüyorum ve and komutunu kullanarak veritabanının ilk harfini bulmaya çalışıyorum.İlk harfi “a” mı diye kontrol ediyorum.False dönüyor.

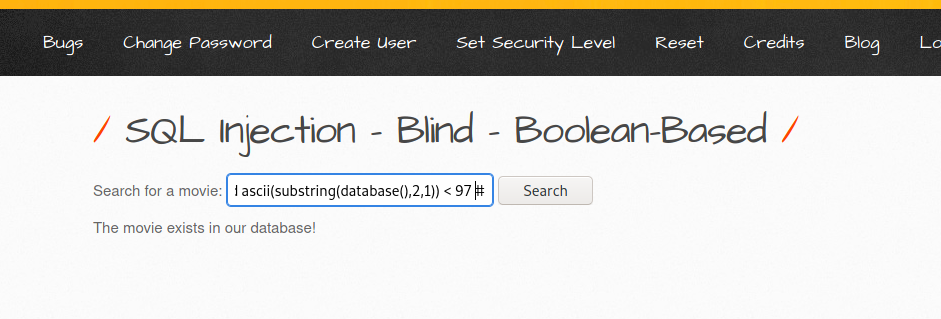


-İlk harfi “b” mi diye deniyorum ve True dönüyor.Veritabanının ilk harfinin B olduğunu öğreniyorum.

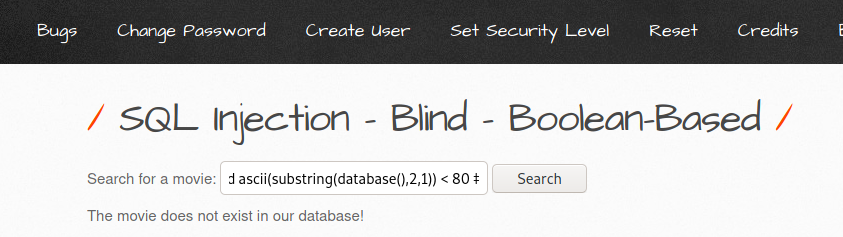
-ABC diye tek tek denemek çok uzun süreceği için farklı ASCII yöntemini kullanacağım.

-ASCII tablosunda 97 sayısı bir sınırdır.Büyük olanlar küçük harf,küçük olanlar ise büyük harftir.

-Bu sebeple ilk önce ASCII değerininin 97’den küçük mü yoksa büyük mü oludğunu kontrol edeceğim.



-True döndüğü için ikinci harfin Büyük harf olduğunu anlıyorum.



-Türkçe karakterler 65-90 arasında olduğu için bu aralıkta deneme işlemi gerçekleştireceğim.False döndüğü için 80’den büyük bir değer olduğunu anlıyorum.



-True döndüğü için 80 ile 90 arasında bir değer olduğunu anlıyorum.

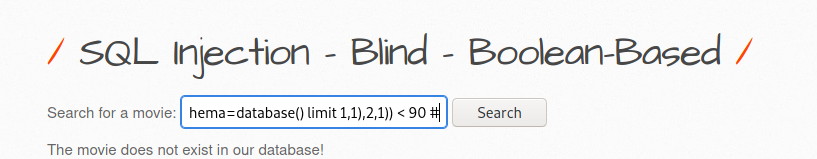


-Ardından doğru değeri bulmak için random değerler deniyorum.True döndüğü için 88’den küçük olduğunu anlıyorum.



-Yaptığım sorguyu = olarak değiştirip 87’yi deniyorum ve True döndüğünü görüntülüyorum.Ascii karakterlerinde 87 “W” harfini temsil etmektedir.İkinci harfin W olduğunu buluyorum.

-Bu şekilde önce Database adını ardından ilk tablosunun adını bulmam çok uzun sürecek bu yüzden kodumu UNION ile birleştiriyorum.



-“Iron man' and ascii(substring((select table\_name FROM information\_schema.tables WHERE table\_schema=database() limit 1,1),2,1)) < 90 #”

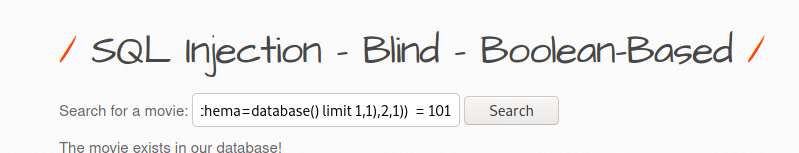
-“Information\_schema\_tables” standart olarak kullanılan bir tablodur.Bu veritabanının ilk tablo adını öğreneceğim.False değeri döndüğü için 90’dan büyük olduğunu öğreniyorum.



-True döndüğü için 105 değerinden küçük olduğunu görüntülüyorum.



-Ardından 101 değerini denediğim zaman;False döndüğünü görüntülüyorum.101’den 105’e kadar değer deneyeceğim.

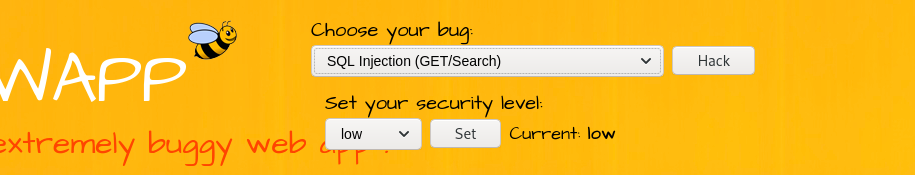


-101’e eşit mi diye sorguladığımda True döndüğünü görüntülüyorum.Ascii karakterlerinde 101 değeri “e” harfini temsil etmektedir.

-Database’in ilk tablosunun ikinci karakteri “e” çıktı db\_name = \*e\*\*\*

-Bu şekilde genellikle işlem yapmıyoruz.SQLMap üzerinden veya BurpSuite üzerinden Payload denemesi yapıp sonuca otomatik araçlar sayesinde erişmekteyiz.Direkt olarak payloadlar üzerinden işlem yapıp geçseydik arka planda yapılan işlemi anlayamayacaktık.

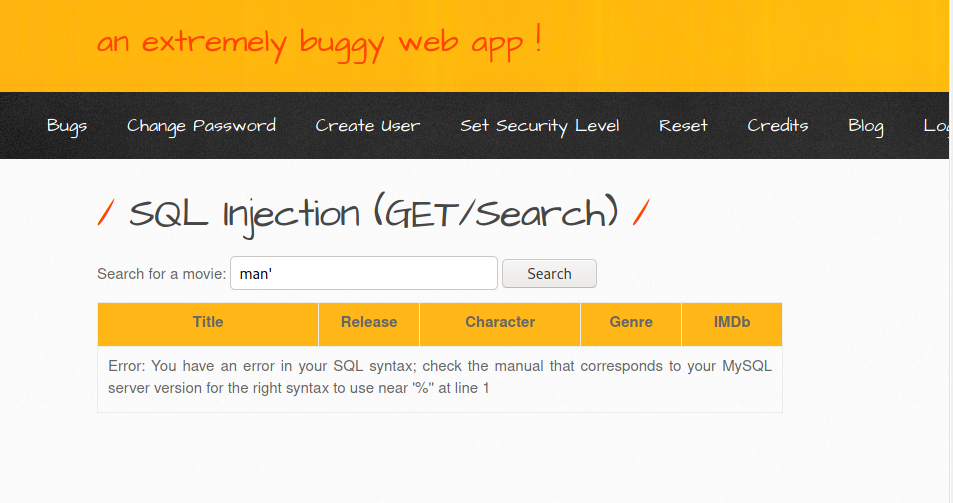
**Uygulama2 :** GET SQL üzerine manuel olarak çalışacağımız zafiyet uygulamasıdır.



-Öncelikle zafiyetimizi başlatıyoruz.

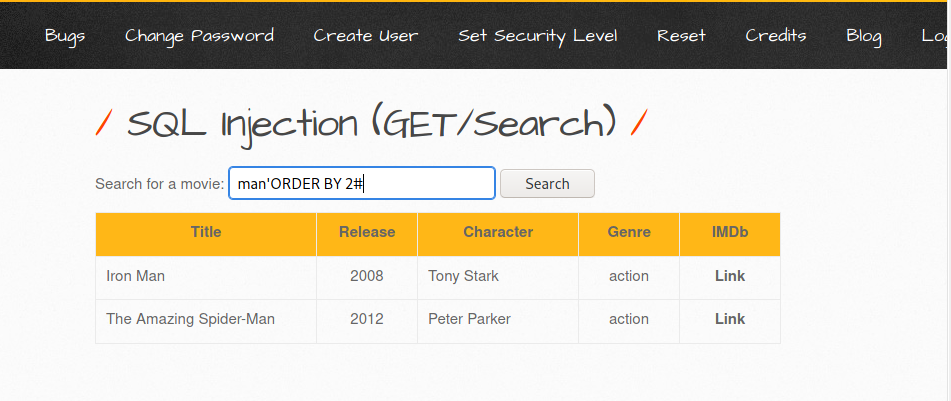


-Blind’den farklı olarak tüm içerisinde arama değerini içeren tüm veritabanını bana döndürüyor.

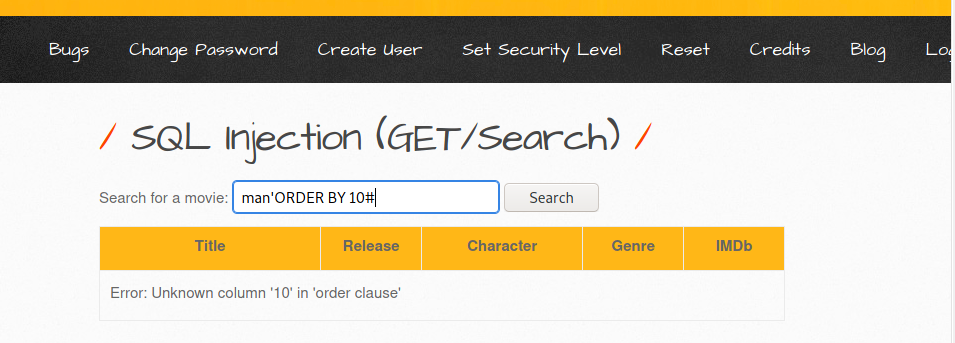


-Kesme İşareti(‘)’ni kullandığımda SQL syntax hatası vermektedir.

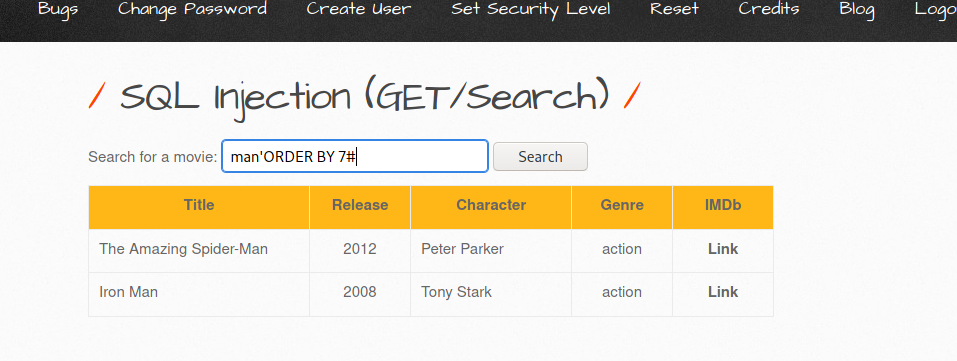
-Bu sebeple Union veritabanı araması gerçekleştiriyoruz.UNION eşit sayıda kolona sahip veritabanlarında kullanılır.



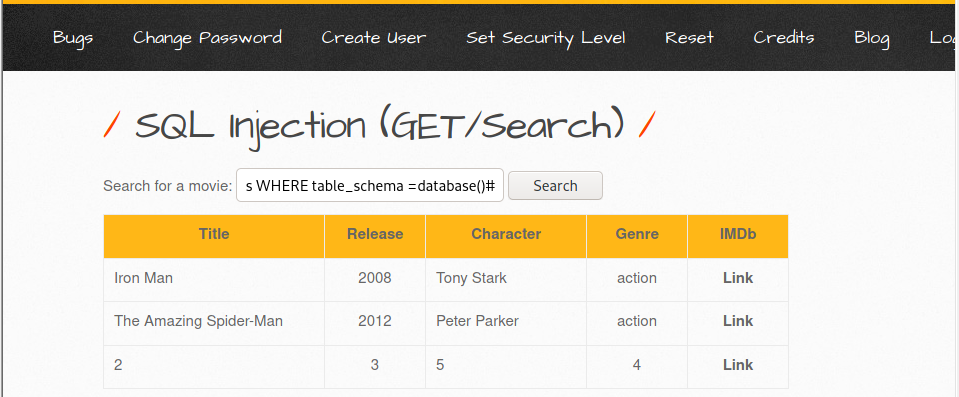
-“Man^ORDERBY=2#” komutu ile ne kadar kolonu olduğunu bulmaya çalışıyorum.Eğer yanlış bir değer girmiş olsaydım bana değer döndürmeyecekti.



-Hata aldığım için 10 adet kolonu olmadığını öğreniyorum.Ardından tam değeri bulasıya kadar değer deniyorum.

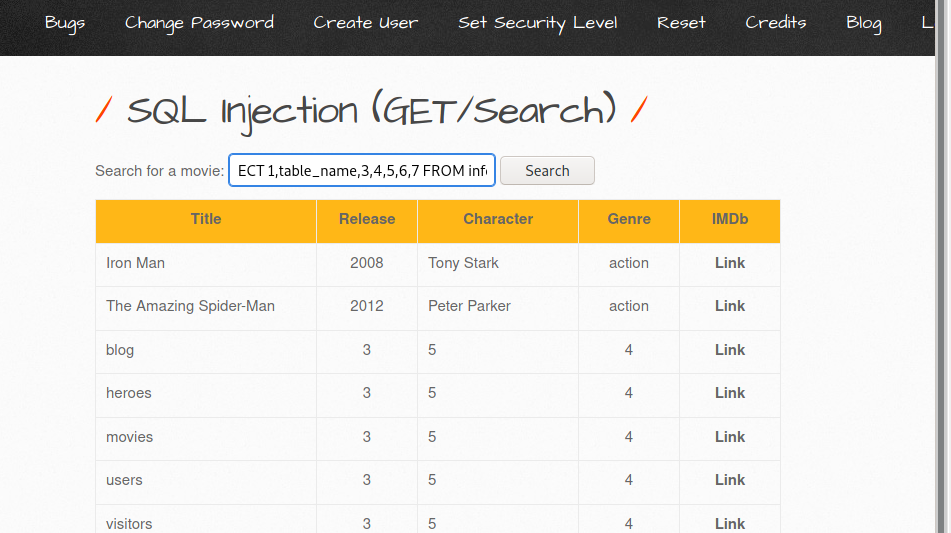


-Tam değerinin 7 olduğunu tek tek deneyip öğreniyorum.



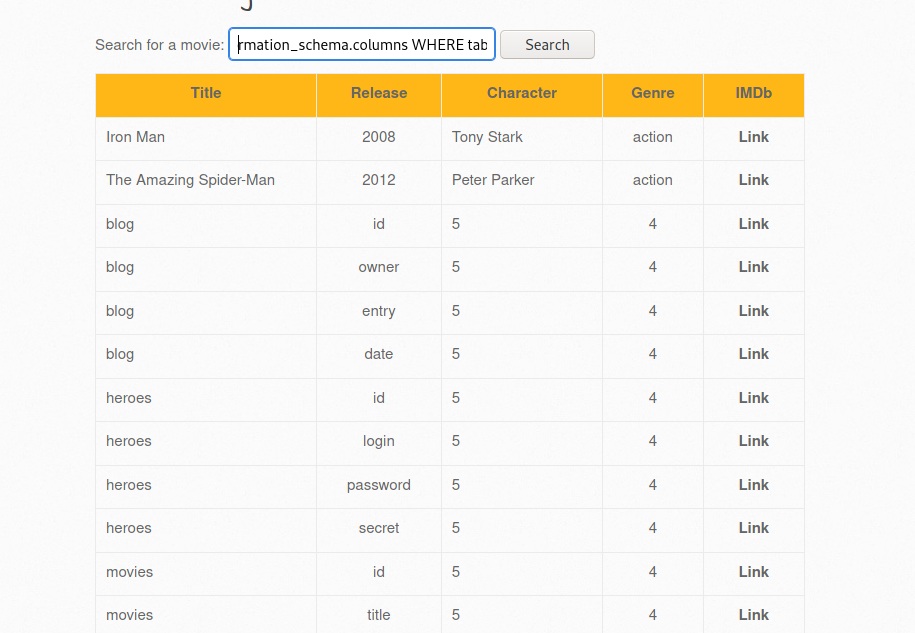
-“man'UNION SELECT 1,2,3,4,5,6,7 FROM information\_schema.tables WHERE table\_schema =database()#”

-Union kodu ile her bir kolona int değer atadım.From’dan sonra tablo ismi girmem gerekiyor fakat bilmediğim için her veritabanı tablosunda olan “information” tablosundan yararlanıyorum.



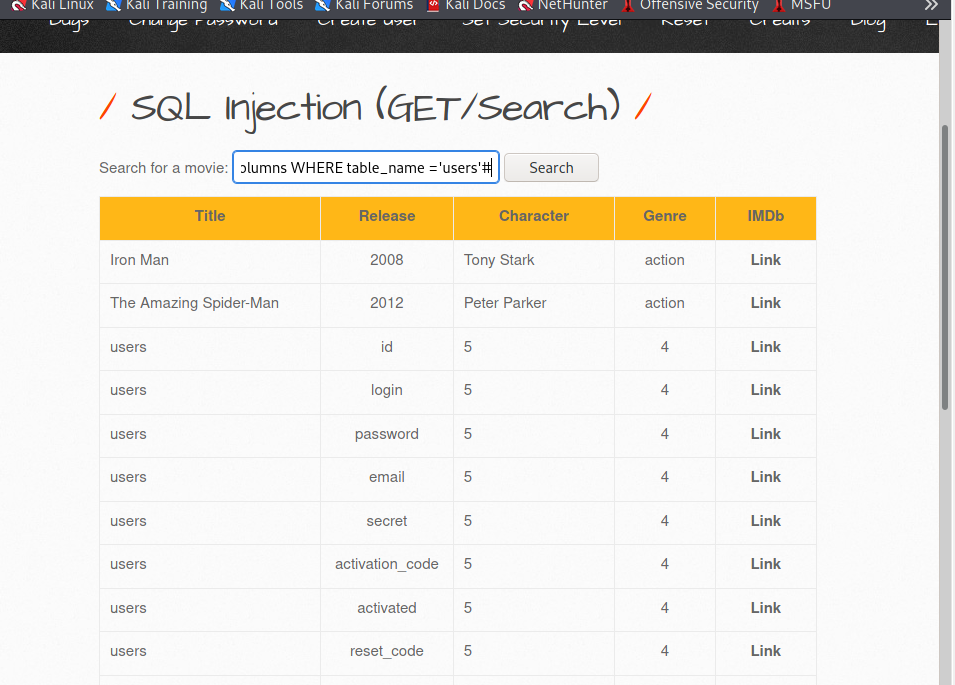
-“man'UNION SELECT 1,table\_name,3,4,5,6,7 FROM information\_schema.tables WHERE table\_schema =database()#”

-Bu adımda ise kullanılmayan şema int. Değerlerini siliyorum ve bulunan değerlere elde etmek istediğim değerleri yazıyorum.Bu işlem sonucunda tablo isimlerini döndürdüm.



-“man'UNION SELECT 1,table\_name,column\_name,4,5,6,7 FROM information\_schema.columns WHERE table\_schema =database()#”

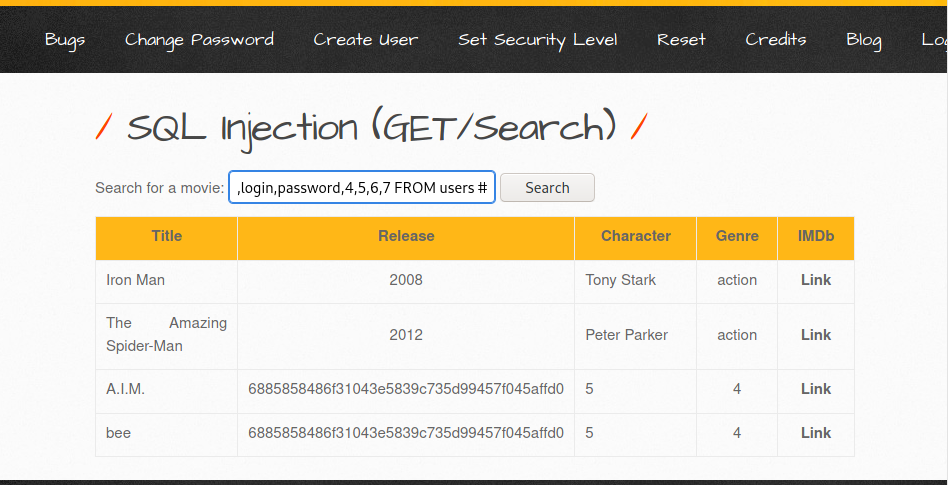
-Bu işlem sonucunda şema isimlerini öğrendim.



-“man'UNION SELECT 1,table\_name,column\_name,4,5,6,7 FROM information\_schema.columns WHERE table\_name ='users'#”

-Users üzerinde ki kolonları çekiyorum.

-Yapmış olduğum işlemler sonucunda login ve password kısmını görüntüledim.Bu kolonlar üzerinden bilgi çekmeye çalışacağım.



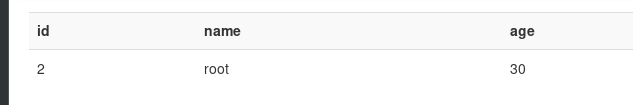
-“man' UNION SELECT 1,login,password,4,5,6,7 FROM users #”

-Yazılımcının yazmış olduğu sql sorgu ile kendi sql sorgumu birleştirdim ve ID ve Pass. Kısmıyla ilgili bilgileri görüntüledim.

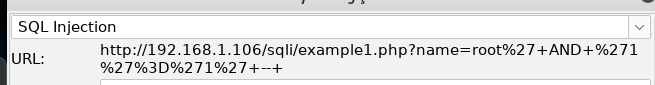
-Web for Pentester uygulamama geçiş yapıyorum ve burada işlemlere başlıyorum.WFP üzerinde 7 adet zafiyet uygulamam vardır.Bu uygulamalar basitden başlayıp zor’a kadar devam edecektir.

**Uygulama3 :** SQL Çalışma mantığını anlamak.

-WFP üzerinde SQL zafiyetlerinden Example1’i başlatıyorum.

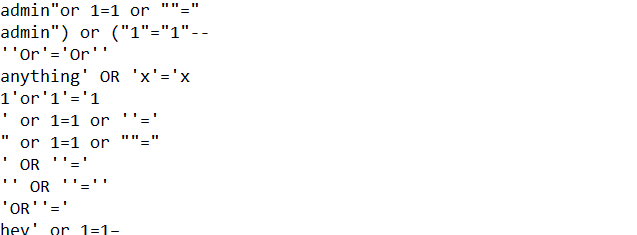


-WFP üzerinden Example1’i açtığımda kullanıcı adı girilen bir textbox var.Dışarıdan girilen bilgileri ID-Name-Age parametrelerini veritabanından getirdiği için açık aranabileceğini düşünüyorum.

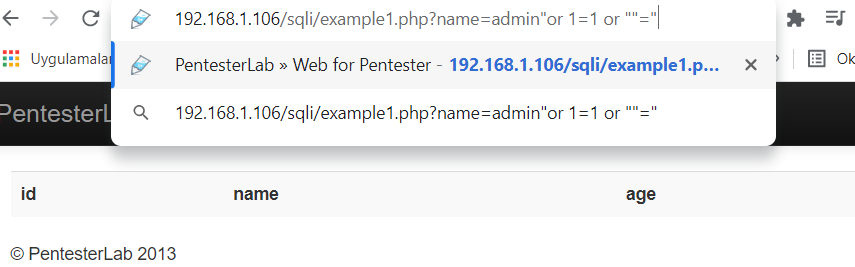


-Zaproxy üzerinden WFP URL’imi zafiyet testinde çalıştırdığımda bulmuş olduğu açığı burada görüntüleyebiliyorum.

-İlk olarak manuel yol ile istismar etmeye başlayacağız.Diğer zafiyetlerde de kullanacağımız gibi burada da payloadlar kullanacağız.

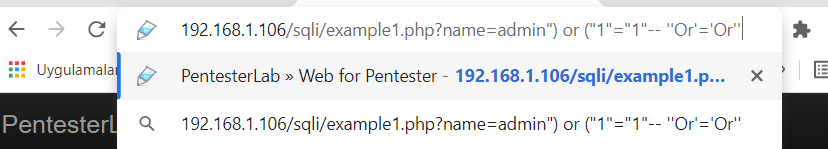


-SQL Payload şeklinde tarayıcımıza yazdığımızda bu ve bunun gibi bir sürü payload elde edebiliriz.



-bWAPP üzerinde yapmış olduğumuz uygulamalarda arama yerinden SQLİ denemesi yapmıştık fakat bu sefer URL üzerinden değiştirmeler yaparak işlem gerçekleştiriyorum.

-İlk denediğim paylaod başarısız oldu bir şey döndürmedi.



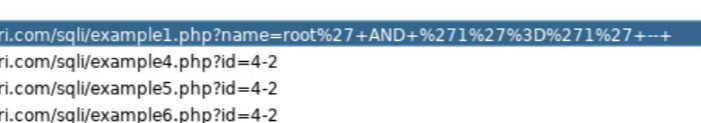
-Kullanmış olduğum bu payload bana bir veritabanı döndürdü.Fakat ilk uygulamamızda amacımız şuan Veritabanı değil çalışma mantığını anlamak.

-Burada yapmış olduğumuz işlem bWAPP üzerinde

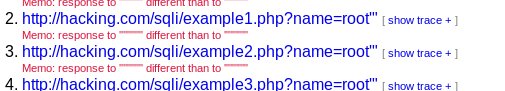
-Select \* from tablo where name="admin" 'OR'='OR" Admin gibi düşünebiliriz.Burada da bWAPP üzerinde ki gibi şart sağlandığı için bilgiler döndürüldü.

**Uygulama 4 :** Manuel olarak işlemlerin nasıl gerçekleştiğini ve mantığını anladıktan sonra Burpsuite üzerinden payload kullanarak işlem yapacağım.

-Bu ve bundan sonra ki uygulamalarımda zafiyet testi olarak zaproxy ve skipfish aracını kullanacağım.

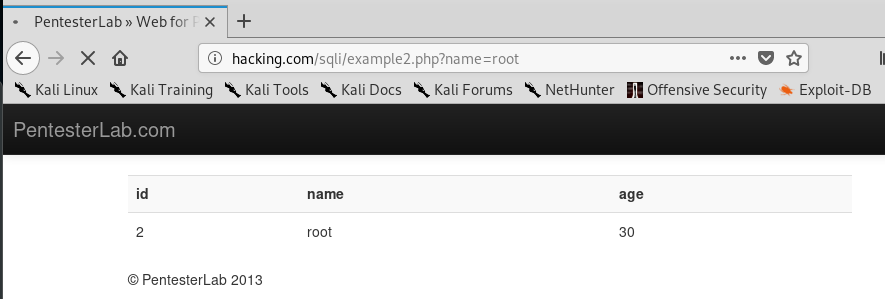


-İlk olarak zaproxy üzerinden gerçekleştirdiğim taramaya bakıyorum fakat herhangi bir zafiyet bulamadı.



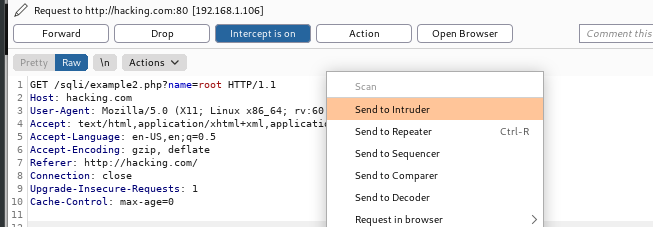
-İkinci olarak ise skipfish aracıyla zafiyet testi yapıyorum ve zafiyeti bulduğunu görüntülüyorum.

-WFP üzerinden Example2 ‘yi açtığımda görüntü bu şekildedir.



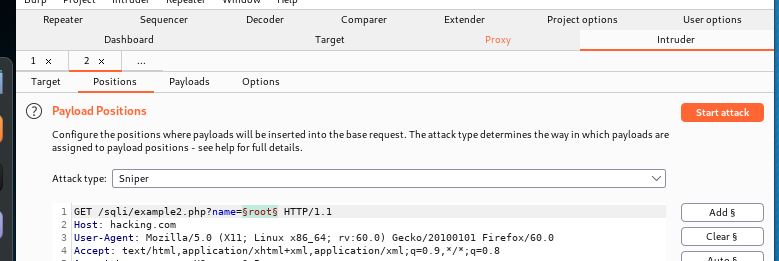
-Arka planda Burpsuite ile Tarayıcım arasında gerekli proxy ayarlarını yapıyorum.Bu ayarlara oluşturduğum Burpsuite dökümanlarında detaylı olarak değindiğim için atlıyorum.

-Burpsuite üzerinden bağlantı kurulduktan sonra sayfayı yeniliyorum ve bilgileri Burpsuite üzerine çekiyorum.



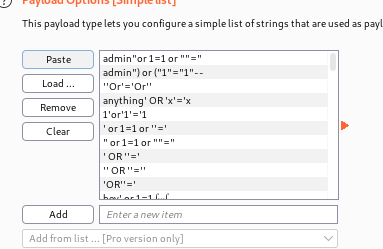
-Burpsuite üzerine geçiş yaptıktan sonra sağ tıklayıp “Send to Intruder” ile devam ediyorum.Bu sekme bana payload denemesi yapabilmem için yardımcı olacak.

-Intruder sekmesine geçiş yapıyorum ve burada kontrol sağlayacağım.

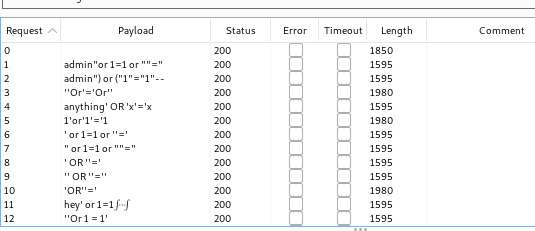


-Position kısmına geçiyorum ve root kısmı seçili olarak geldiğini görüntülüyorum.Root kısmı seçili olarak gelmeseydi.Clear komutu ile silip daha sonrasında değişecek olan bölümü “Add” ile ekleyip işleme devam edecektim.

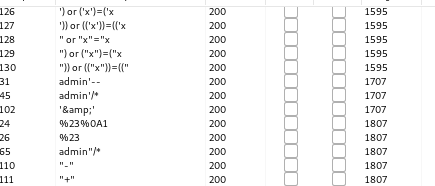
-Yapmış olduğum işlem sonrasında Intruder>Payload sekmesine geçiş yapıyorum.



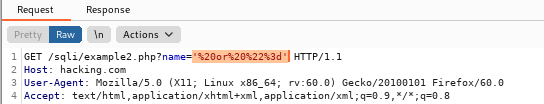
-Buraya daha öncesinden manuel olarak deneme yaptığım payloadlarımı ekliyorum ve paylaod deneme işlemini başlatıyorum.



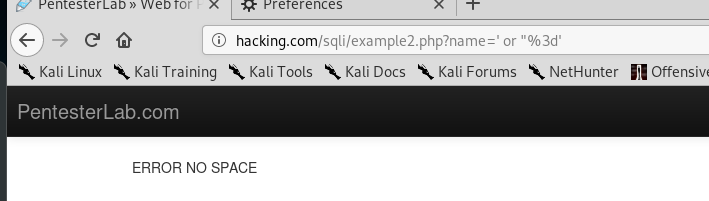
-Tarama sekmesi üzerinde deneme yaptığı payloadları görüntüleyebiliyorum.Bu payloadları “Length” değerlerine göre ayırt edeceğim.



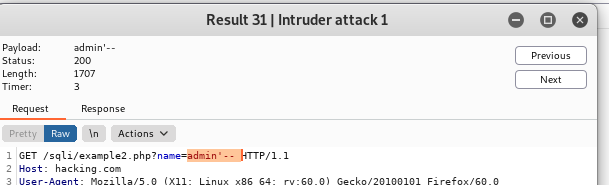
-Sonuçlara daha detaylı baktığımda 4-5 adet farklı değer çıktığını görüntülüyorum.Bu farklı değerler içerisinden 1 adet deneme yapmam o zafiyet denemesinde başarılı olup olmadığını anlamam için yeterli olacaktır.



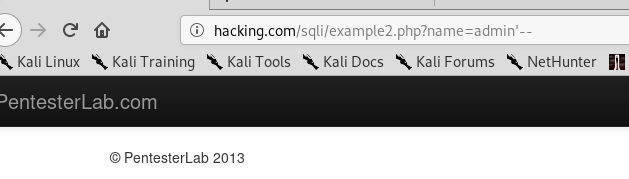
-İçlerinden bir tanesini seçiyorum ve URL kısmını kopyalıyorum.



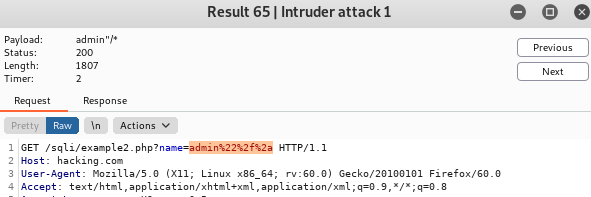
-Ardından tarayıcı üzerinde görüntüleme yaptıktan sonra denemiş olduğum payload üzerinde boşluk olduğu için onun engellendiğini görüyorum ve diğer payloadlar üzerinden aynı işleme devam ediyorum.



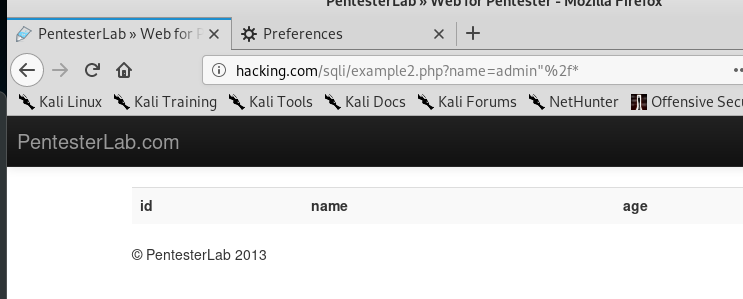
-1707 Length değerinden bir tanesini seçiyorum ve URL’ini kopyalıyorum.



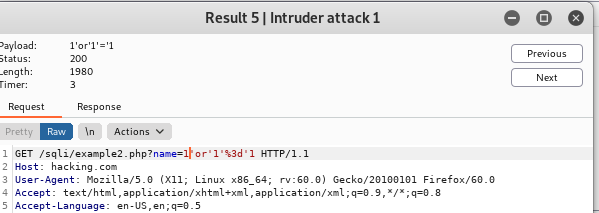
-Tarayıcı üzerinden çalıştırdığımda hiçbir dönüt alamadığım için diğer değerlere bakıyorum.



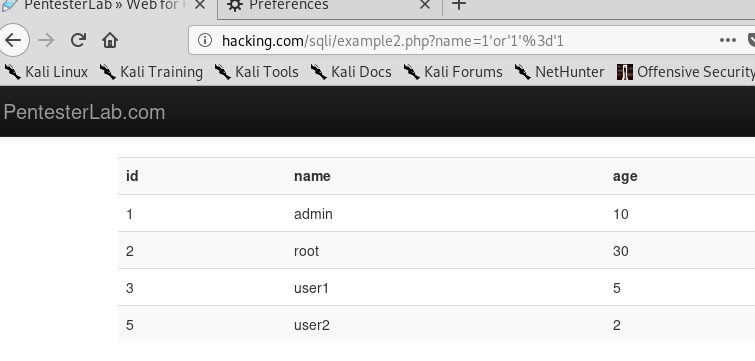
-1807 Length değeri ile devam ediyorum ve URL’ini kopyalıyorum.



-Tarayıcı üzerinden çalıştırdığımda hiçbir dönüt alamadığım için diğer değerlere bakıyorum.



-1980 “Length” değerleri ile devam ediyorum.Bu URL’i kopyalayıp tarayıcı üzerinde çalıştırıyorum

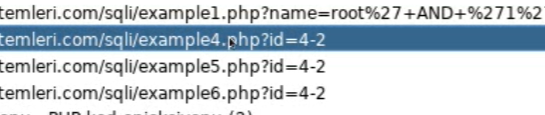


-Tarayıcı üzerinde çalıştırdığımda zafiyetin başarılı bir şekilde çalıştığını ve bana tabloları geri döndürdüğünü görüntülüyorum.

**Uygulama 5 :** SQL Map yardımı ile Unit SQL Veri tabanından bilgi çekme.

-İki veya daha fazla select ifadesinin sonucunu tek bir noktadan gösterip bilgileri ele geçirmeye çalışacağım.

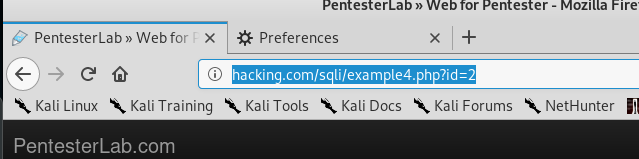
-Her zaman olduğu gib zaproxy ve skipfish üzerinden zafiyet tespit edilmiş mi diye bakıyorum.



-Zaproxy aracının zafiyeti başarılı bir şekilde bulduğunu görüntülüyorum.

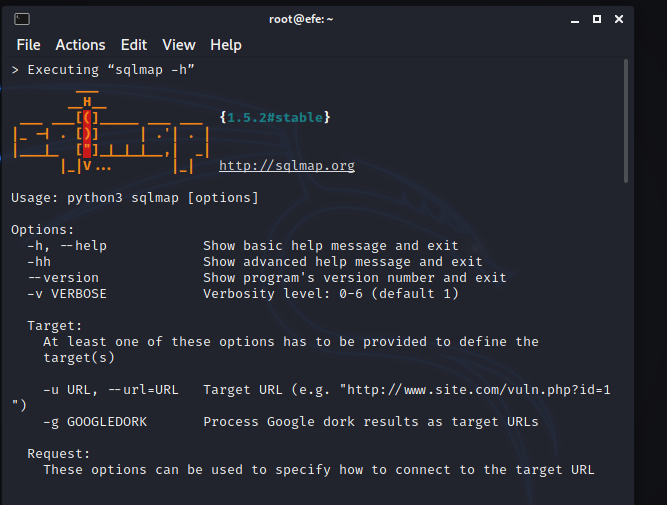


-Skipfish aracının da zafiyeti başarılı bir şekilde bulduğunu görüntülüyorum.

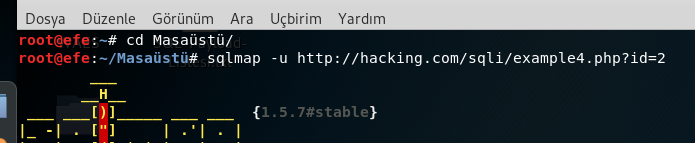


-WFP üzerinden Example4’ü açıyorum ve URL üzerinden incelediğimde diğerlerine benzer bir yapıda olduğunu görüntülüyorum.

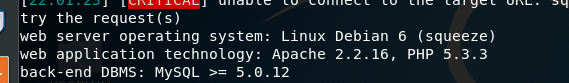
-Sqlmap üzerinden işlem gerçekleştireceğim için URL’i kopyalıyorum.



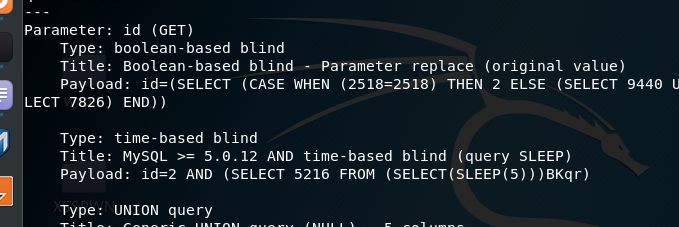
-“Sqlmap –h” komutu ile kullanabileceğim komutlar ve yazış biçimiyle ilgili bilgilere erişebiliyorum.



-Sqlmap ile zafiyetin bulunduğunu bildiğim URL üzerinden sqlmap tarama işlemini başlatıyorum.



-Burada kullandığı SQL’i görüntüleyebiliyorum.

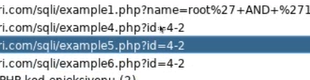


-SQL’in yapısına göz attığımda boolean olarak görüntülüyorum.Bu tür üzerinde bWapp üzerinde manuel olarak işlem yapmıştık.Eğer farklı bir tür olsa idi tek tek deneme yapmamıza gerek kalmazdı.

-bWAPP anlatımında tek tek deneme yaptığım için sizleri sıkmamak için bu adımı atlıyorum.

**Uygulama 6 :** JSQL ile elde edilen veritabanını görüntüleme.

-Diğer uygulamalarda olduğu gibi önce zaproxy ve skipfish araçları ile zafiyet testi gerçekleştiriyorum.



-Gerçekleştirmiş olduğum zafiyet testleri sonucunda bilgi elde edebildim.

-JSQL elde ettiğimiz veritabanlarını görsel olarak incelememize yardımcı olan bir tooldur.

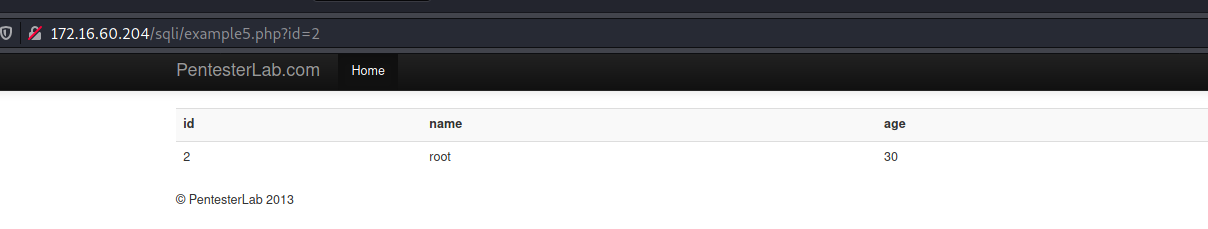
-JSQL’in kurulumu ile devam edelim.



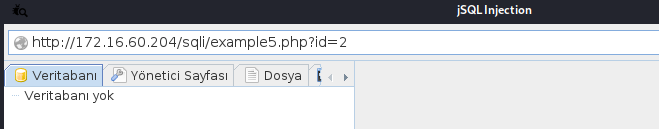
-Linux üzerinden git clone komutu sayesinde ilk önce kurulum dosyalarını indiriyorum.



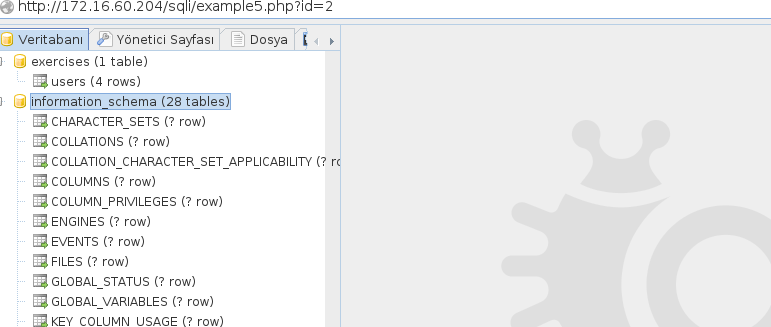
-İnen .jar uzantılı dosyayı java komutu ile çalıştırıyorum ve kurulumunu sağlıyorum.



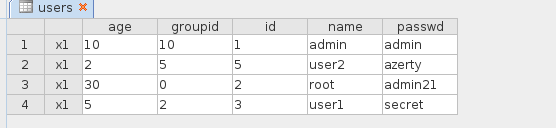
-Kurulum sonrasında WFP ‘a geçiş yapıyorum ve Example5 üzerinde ki URL’i kopyalıyorum.



-Kopyalama işlemi sonrasında JSQL üzerine geçiş yapıyorum ve URL’i buraya yapıştırıp çalıştırıyorum.



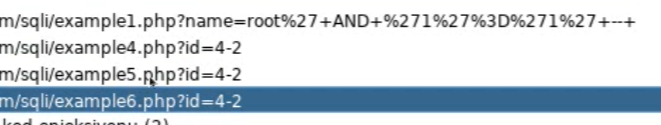
-İçerisinde bulunan tabloları görüntüleyebiliyorum.



-Son olarak ise görüntülemek istediğim tablodaki kolonları seçip sağ tıklıyorum ve “Thread yükü” seçeneği ile tabloları görsel bir şekilde görüntüleyebiliyorum.

**Uygulama 7 :** SQLMap ile Time Based Blind.

-Öncelikle Zaproxy ve skipfish araçlarıyla yapmış olduğum testleri görüntülüyorum.

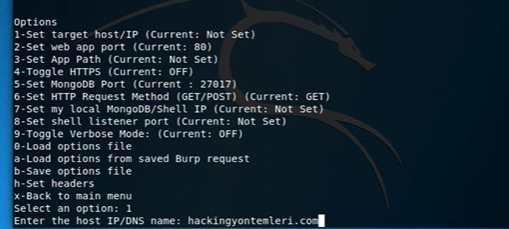


-Zaproxy aracı zafiyeti bulmakta başarısız oldu.



-Skipfish aracına baktığımda onunda zafiyeti bulamadığını görüntülüyorum.

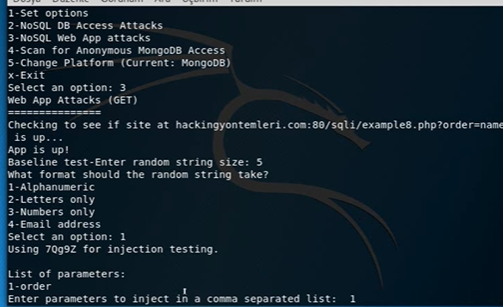
-Zafiyetleri görüntüleyemediğim için Nosqlmap uygulaması zafiyet taraması gerçekleştireceğim.



-Nosqlmap’i açtıktan sonra 1 seçeneği ile devam ediyorum ve buraya WFP linkimi bırakıyorum.



-Ardından port numarasını 80 olarak veriyorum.HTTP 80 portu üzerinden işlem yapmaya olanak sağlar.



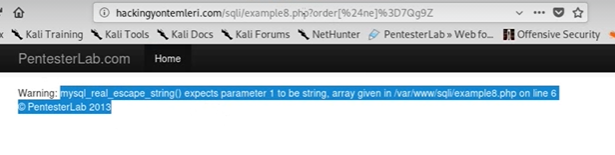
-Açılan sekme üzerinde 3 seçeneği ile devam ediyorum ve buraya zafiyetimin bulunduğu Example8’in URL’ini yapıştırıyorum.

-Ardından “Alpha” seçeneği ile devam edip işlemi başlatıyorum.



-İşlem sonucunda tarafıma zafiyet bulduğu 2 adet URL döndürdü.

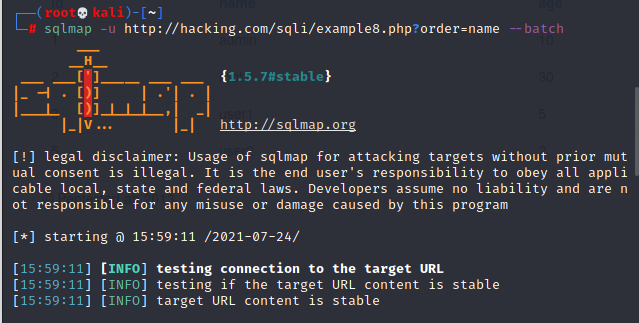
-Linkleri ilk önce manuel olarak çalıştırıyorum.



-Manuel olarak çalıştırdığımda SQL hatası ile karşılaşıyorum.Bu sebeple bu linkler üzerinde Sqlmap ile bir çalışma gerçekleştireceğim.

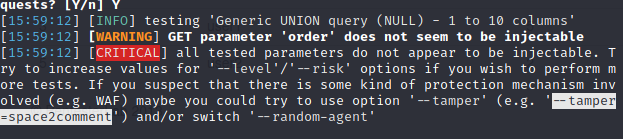


-URL’ e yönelik bir işlem gerçekleştireceğim için Example8’in URL’ini kopyalıyorum.

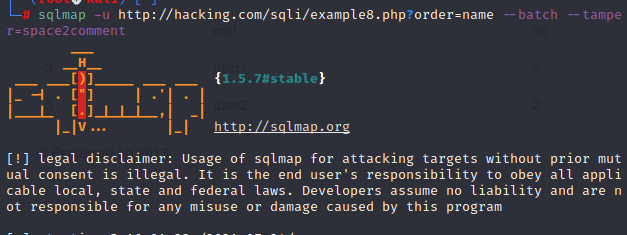


-Ardından uçbirimi açıp bu link üzerinde Sqlmap ile çalışmalara başlıyorum.

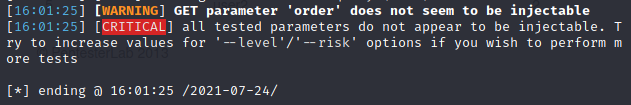
-“—Batch” komutu ile bana soracağı sorulara default olarak cevap döndürmesini sağlıyorum.



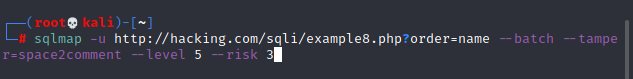
-Sqlmap bu işlem sırasında bana “–tamper” ile devam etmem gerektiğini belirtiyor.



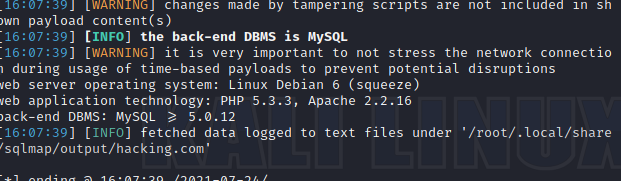
-Tamper ile işleme devam ediyorum.



-İşleme devam ederken Sqlmap --level ve --risk komutunu eklememi istiyor.



-Bu komutları da ekleyip devam ediyorum.

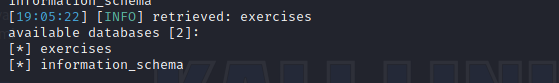


-MySQL sonucunu aldık bundan sonra yapacağımız işlem manuel olarak yapılırsa çok çok uzun süreceği için Sqlmap üzerinden devam edeceğim.



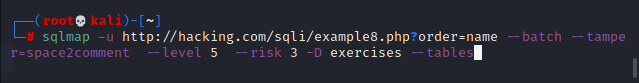
-Yazmış olduğumuz komut sonrasında veritabanıyla ilgili bilgi vericek.Şuanda sql map arka planda sürekli istekler göndererek bize bilgi vermeye çalışıyor.

-Veritabanı eğer Union Based olsa idi bu işlemin gerçekleşmesi saniyeler sürecekti.

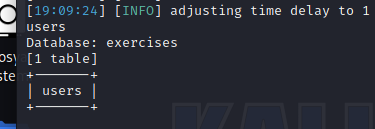


-İki adet veritabanı olduğunu bildiriyor.İnformation schema basic olarak olan bir tablo içerisinde ayarlar bulunuyor.

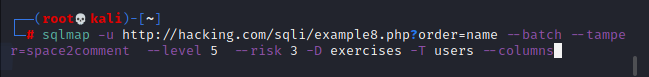
-Komutu tekrar değiştiriyorum bu sefer veritabanı adıyla işlem yapıp tabloları getiricem.



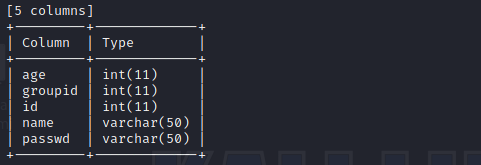
-Eklemiş olduğum komut ile exercises veritabanı üzerinden bilgi çekmeye çalışıyorum.



-Tablo ismini bulduktan sonra bu sefer yazdığım komut ile kolonları getirmeyi hedefliyorum.



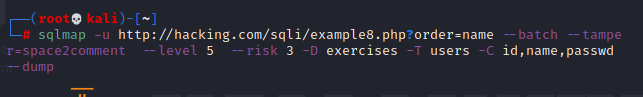
-Kolon isimlerini bulmak için işlemi başlatıyorum.



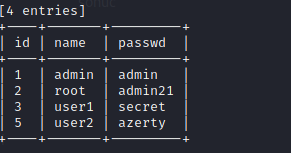
-Sqlmap’in döndürdüğü veri sonrasında Kolon isimlerini başarılı bir şekilde bulabildim.

-Bu işlem yaklaşık 10 dakika sürdü.

-Kolon isimlerinde işime yarayabilecek id,name ve passwd değerleri dikkatimi çekiyor.



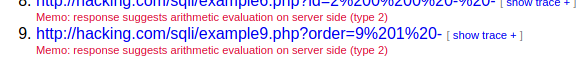
-Kolon isimleri üzerinden içerisinde ki bilgileri çekmeye çalışıyorum.



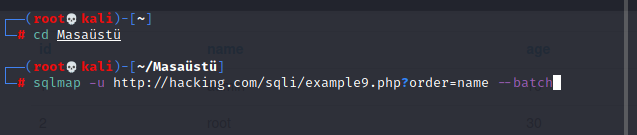
-Bilgi çekme işlemi başarılı oldu veritabanı üzerinde ki tüm bilgilere erişim sağlayabildim.

-Bu işlem veritabanının büyüklüğüne bağlı olarak 1 gün bile sürebilir.

**Uygulama 8 :** SQLMap ile Time Based Blind2.

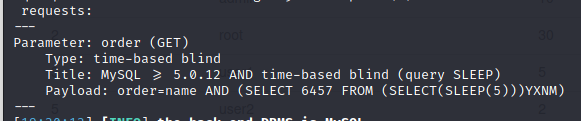


-Skipfish aracı zafiyet testi sonucunda açığı tespit etti.

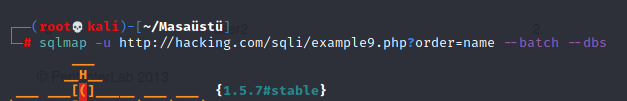


-Sqlmap ile kullanabileceğimiz saldırı türünü öğrenmeye çalışacağım.Webforpentester üzerinde ki Example9 un linkini kopyalıyorum ve sql üzerinde komutumu uyguluyorum.

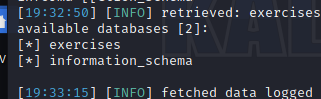
-Batch komutu default olarak bize soru sorulmadan kendi ilerlemesini sağladık.



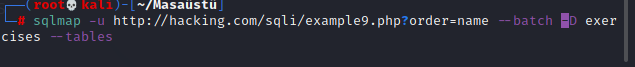
-Keşif işlemi bitti saldırı türü olarak time-based blind'i buldu.Zaman ayarlı paket gönderirken zaman bilgisi de ekleniyor ve belirli bir zaman diliminde döner ise çalıştığı biliniyor.



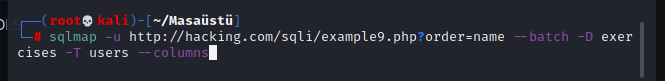
-Daha sonrası -dbs komutunu ekleyerek veritabanı isimlerini öğrenmeye çalışıyorum.



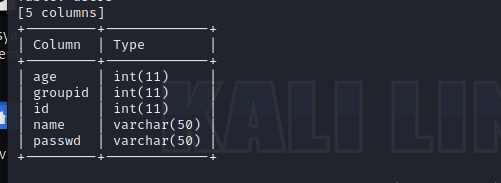
-Veritabanı isimlerini başarılı bir şekilde öğrendim.Exercises tablosu üzerinden kullanıcı bilgilerine erişim sağlamaya çalışacağım.



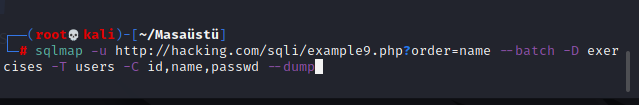
-Şuan Exercises tablosu üzerinden bilgi edinmeye çalışıyorum.İçerisinde users tablosu olduğunu görüntülüyorum.



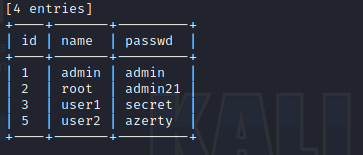
-Users tablosu içerisindeki kolonlara erişim sağlamaya çalışıyorum.



-Users tablosu içerisinde bulunan kolonları başarılı bir şekilde görüntüledim.



-İçerisinde işime yarayacak olan id,name,passwd değerlerini çekmeye çalışacağım.



-İçerisinde bulunan bilgilere başarılı bir şekilde erişim sağladım.