2022 – 2023 Güz Dönemi



Yavuz ÇİÇEK

211110024

E-Mail Adresi: [kuguvega@linuxmail.org](mailto:kuguvega@linuxmail.org)

Bilişim Güvenliği Teknolojisi

BGT211 SİBER GÜVENLİK

VİZE ÖDEVİ

Öğr.Gör.ANIL YILDIZ

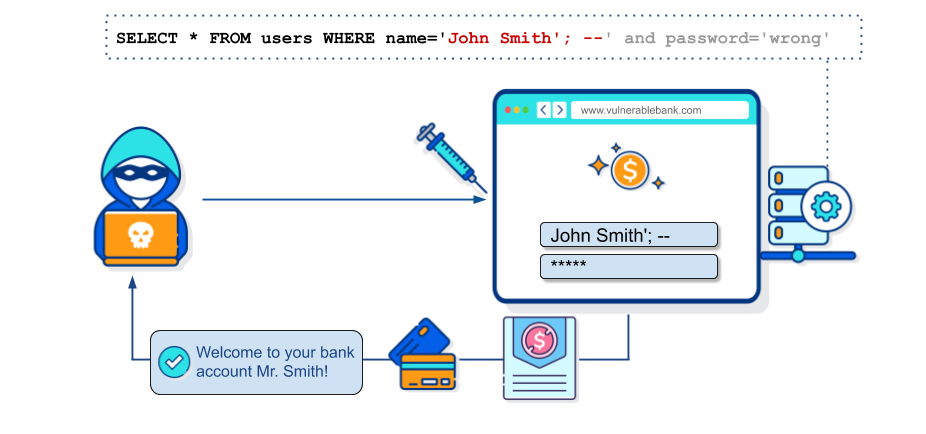
İstanbul Aydın Üniversitesi

Mühendislik Fakültesi

8 Kasım 2022

1. SQL Enjeksiyonu Tanımı

Uygulamalara, güvenilir olmayan kaynaklardan gelen girdilerin kontrol edilmeden SQL sorgusunu oluşturan cümlecik içerisinde kullanılmasından kaynaklanmaktadır .Modern web uygulamalarının hemen hemen hepsi ilişkisel veritabanlarını, dolayısıyla SQL yapısal sorgulama dilini kullanmaktadır.Fakat gerek kullanılan yöntemler (prepared statement/parametrized query/bind variables ) ,gerekse de kullanılan uygulama çatıları nedeni ile geçmişteki kadar (2010 öncesi) her uygulamada görülebilecek bir zafiyet olmaktan çıkmıştır.Fakat etkisi sebebiyle ciddiye alınması gereken problemlerden biridir.Aynı zamanda her uygulamada olmasa bile hala azımsanmayacak sayıda uygulamada SQL Enjeksiyon zafiyeti rastlamak mümkündür.

SQL dili, aslında uygulamaların arka uçta bulunan veritabanı sunucuları ile konuşmasına olanak sağlayan dildir.Dolayısıyla bu dil yardımı ile uygulama geliştiriciler, kullanıcıların uygulama üzerinden göndermiş oldukları verileri işleyerek veritabanına kaydeder ve daha sonra bu verilerin bir kısmınıda gösterim için kullanırlar .

**Şekil 1 : SQL Injection (Kaynak 1)**

SQL Enjeksiyonu saldırılarının sızma testi gerçekleştirenler için sevindirici yanı; uygulama geliştiricilerin SQL sorgularına uygulamanın birçok yerinde ihtiyaç duymalarıdır.Elbette bir uygulamanın çok sayıda SQL sorgusu içermesi onu zayıf kılmaz.Fakat her ne kadar güvenli kod geliştirme prensiplerine uygun uygulama geliştiriyor olsak bile çok sayıda girdinin kontrolünün uygulanabilir olmasının zorlukları sebebiyle en az bir SQL Enjeksiyonu zafiyetine sahip olan uygulamanın güvensiz olduğunu söylemek yanlış olmaz.

SQL Enjeksiyonunun oluşmasına sebep olan temel problemin uygulama geliştiricilerin SQL cümlecikleri içerisinde dinamik parametreler kullanması ve bunların hiçbir kontrolden geçirmemesi veya yetersiz kontroller kullanmasından kaynaklandığı bilinmektedir.

1. SQL Enjeksiyon Zafiyeti Çeşitleri

SQL Enjeksiyonun tespiti diğer zafiyetlerden olduğu gibi yapılan isteğe karşın sunucudan alınan cevabın analiziyle mümkün olmaktadır .Dolayısıyla verilecek olan SQL Enjeksiyonu zafiyeti çeşitleri de bu sonuçların farklılıkları ile birbirinden ayrılmaktadır.

1. Union Tabanlı(Union Based) SQL Enjeksiyonu

SQL dilindeki yüzlerce komuttan biride “union” komutudur.Union,”birleştirme” komutudur.Burada dikkat edilmesi gereken Union tabanlı bir SQL Enjeksiyonunun istismarı için zafiyet barındıran SQL cümleciğinin kullanıcıya muhakkak veritabanından çektiği verileri gösteriyor olmasıdır. Aksi takdirde UNION cümleciklerini SQL cümleciğini ilave etmenin anlamı olmayacaktır.

Union’un iki farklı cümleciği bulunmaktadır;

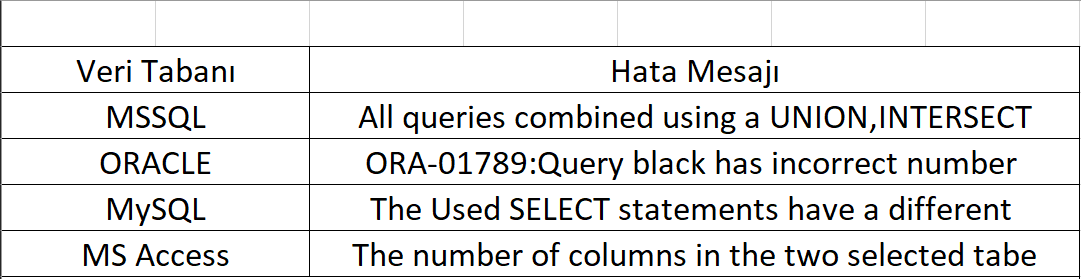
* SELECT column1 FROM table1 UNION SELECT \* FROM table1
* SELECT column1 FROM table1 UNION ALL SELECT \* FROM table1

UNION : İki tabloyu birleştirir ve tüm kayıtlar tekil olacak şekilde listeler.

UNION ALL : İki tabloyu birleştirir ve tüm kayıtlar tekil olmayacak şekilde (aynı kayıttan birden fazla görme ihtimaliniz var ) listeler.

Union sorgularında dikkat edilmesi gereken diğer bir husus ise;UNION kullanılan iki tablodan çekilen kolon ve her bir kolonun karşılıklı veri tipinin aynı olması gerektiğidir.Fakat SQL Enjeksiyonu barındıran bir uygulamada kolon veri tipleri farklı da olsa bunların atlatılması (bypass edilmesi) mümkün olabilmektedir.

* 1. Tespiti

Union tabanlı SQL Enjeksiyonu saldırılarının tespiti,genel SQL Enjeksiyonu saldırılarının tespiti ile neredeyse aynıdır.Eğer bir hata elde ediliyor ise ve bu hatanın elde edildiği noktada aslında kullanıcıya veritabanından çekilen veri gösterimi yapılıyor ise çoğunlukla Union bazlı istismar gerçekleştirilebilir demektir.Fakat yine de alınan hatalardan emin olunmak istenir ise Union sorguları sonucu çıkabilecek muhtemel hataları takip etmek gerekecektir. Örnek bir Union denemesinde, çeşitli veritabanında alabileceğiniz hata mesajları aşağıdaki gibidir.

Eğer hedef veritabanı üzerinde geçerli olan bir tablo için herhangi bir Union sorgusu deneniyor ise alınacak bir hata mesajları yukarıdaki gibi olacaktır.Bu aslında gerçekleşmesi gereken 1. Kural olan Union sorgusuna dahil edilen iki tablonun çekilen kolon sayılarının aynı olması gerekliliği koşulundan doğmaktadır.Bazı uygulamalarda hata mesajlarının kullanıcılara gösterilmediği durumlar ile karşılaşabilirsiniz.Bu durum zafiyetin olmadığı anlamına gelmeyecektir. Her ne kadar hata mesajı kullanıcıya gösterilmiyor olsa bile SQL Enjeksiyonu zafiyeti için bu tür bir uygulama testi gerçekleştiriliyor ise tespitinden çok istismarının yapılıyor olması gerekecektir.Çünkü burada da tahmin ile yürütülen bir seneryo işletmek gerekir.Yine geçerli olan bir tablo adı ve olması gerektirdiği kadar kolon sayısı ile Union sorgusu hazırlanır ve uygulamaya gönderilir.Kullanıcıya gösterilen veriler içerisinde bizim Union ile almak istediğimiz veriler bulunuyor ise , hata elde etmeden de tespit ve istismar aynı anda gerçekleştirilmiş olacaktır.

* 1. İstismarı

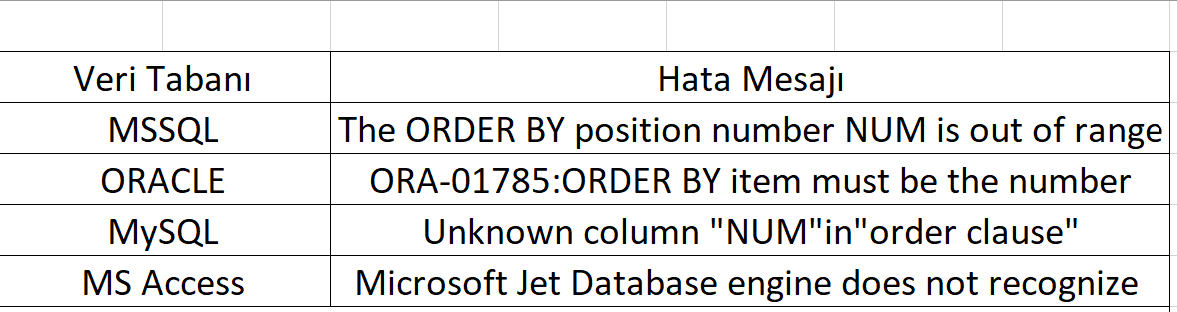
Union bazlı SQL Enjeksiyonunda kullanılan iki tablodaki kolon sayılarının aynı olması gerektiğini belirtmiştik.İstismar için kullanılacak kolon sayısının bulunması için basit bir yöntem bulunmaktadır.SQL sorgularında sıralama için kullandığımız ORDER BY anahtar kelimesi , bize kolon sayısının bulunmasında yardımcı olacaktır.

**Örneğin;** Ad, Soyad ,Telefon bilgileri içeren üç kolonlu kullanıcı adında bir tablomuz olsun.

Sorgusu yardımı ile Ad kolonuna göre sıralama yapabilmekteyiz

Sorgusu ile telefon kolonuna göre sıralama yapabilmekteyiz .Fakat eğer aşağıdaki gibi bir sorgu çalıştırırsak.

O zaman hedef veritabanı üzerinde mevcut örnek için bir hata elde ediyor olacağız. Çeşitli veritabanları için alınabilecek hata mesajları aşağıdaki gibidir.



İlgili örneğimizde de ORDER BY 1, ORDER BY 2, ORDER BY 3 ile hata almazken ORDER BY 4 ile yukarıdaki hata mesajlarından birini alıyor oluruz.Bu da kullanılan tablonun üç kolondan oluştuğunu bize bildirir.Dolayısıyla Union için kullanacağımız ikici tabloda maksimum kaç adet kolon kullanmamız gerektiğini öğrenmiş oluruz.

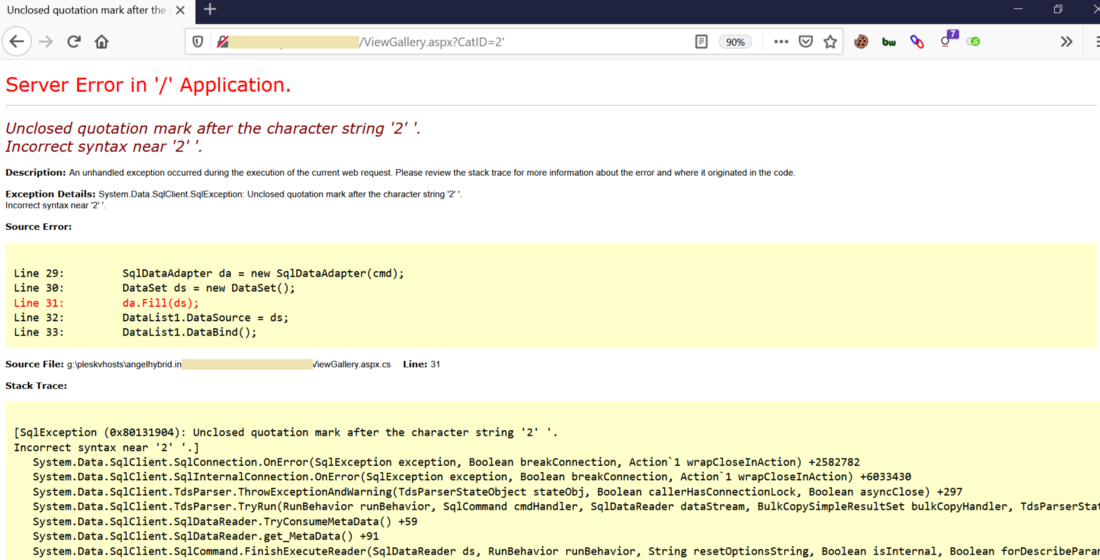
1. Hata Tabanlı (Error Based) SQL Enjeksiyonu

Uygulama geliştiricilerin veritabanı hata mesajlarını aleni şekilde kullanıcılarına göstermesi sonucu SQL Enjeksiyonu zafiyetinin tespit ve istismarı daha da kolaylaşmaktadır.Çünkü elde edilen hata mesajları sayesinde kullanılan veritabanı adı ve türü , tablo adı, veri tipi ve uzunluğu , metotlar vb. gibi parametreler hakkında bilgi elde edilmekte ve bu bilgiler doğrultusunda hedefe daha az adet deneme ile mümkün olabilmektedir.

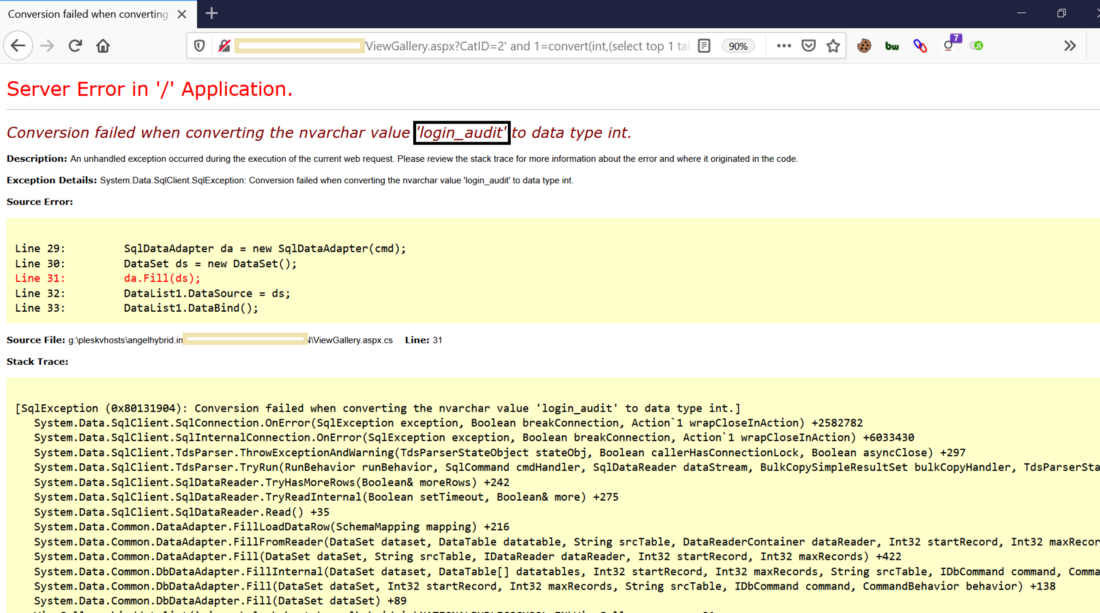
* 1. Tespiti

Yukarıda da bahsettiğimiz gibi tespit etmek için veritabanı hata mesajının ekrana basılıyor yeterli olmaktadır.Fakat dikkat edilmelidir ki, kimi uygulamalarda hata mesajları kullanıcıların göreceği şekilde olmayıp HTML yorum satırı (comment out) içerisinde saklanıyor olabilir.Dolayısıyla yapılan isteğin cevabı tarayıcıda hata mesajı olmaksızın gözüksede hata mesajının HTML kaynak kodlarının incelenmesi gerektiği unutulmamalıdır.

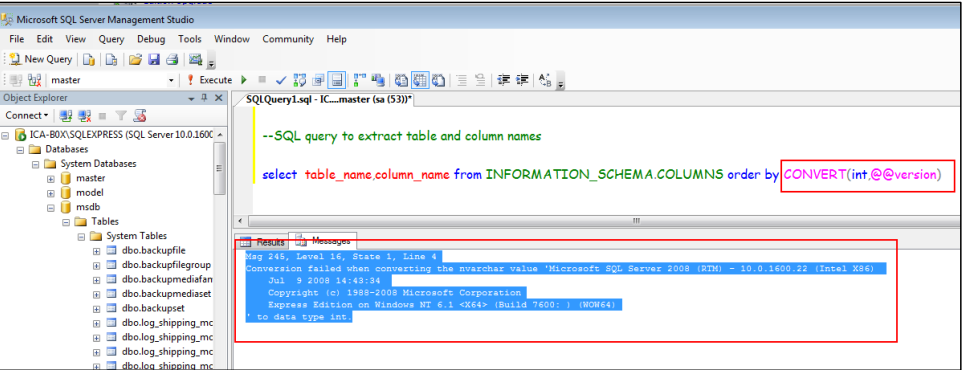
* 1. İstismarı

Yine Klasik test parametremiz olan tek tırnak karakterini hedef uygulamanın zafiyet barındırdığını düşündüğümüz girdi noktasına gönderelim ve aşağıdaki gibi bir hata mesajı aldığmızı varsayalım

**Şekil 2 : Error Based Injection (Kaynak 2 )**

CatID=’ (tek tırnak yerine, CatId= ‘ and 1=convert(int,(select top 1 table\_name from information\_schema.tables where table\_name not in (‘Download\_Dcoument’,'login\_audit')))-- gibi bir ifade girildiğinde ise ;

Error converting data type nvchar to int gibi bir hata mesajı alıyoruz.Bu da bu alanda tam sayı beklendiğini ama bizim onun yerine bir karakter kümesi gönderdiğimiz anlamına gelmektedir.Bu aşamadn sonra hedef uygulamanın veritabanı sunucusu hakkında bilgi almak adına bir deneme yapılabilir.Bunun için ise test parametresi ve isteği yapıyor olmak çoğunlukla yeterli olacaktır.



**Şekil 3 : Error Based Injection Check Version (Kaynak 3 )**

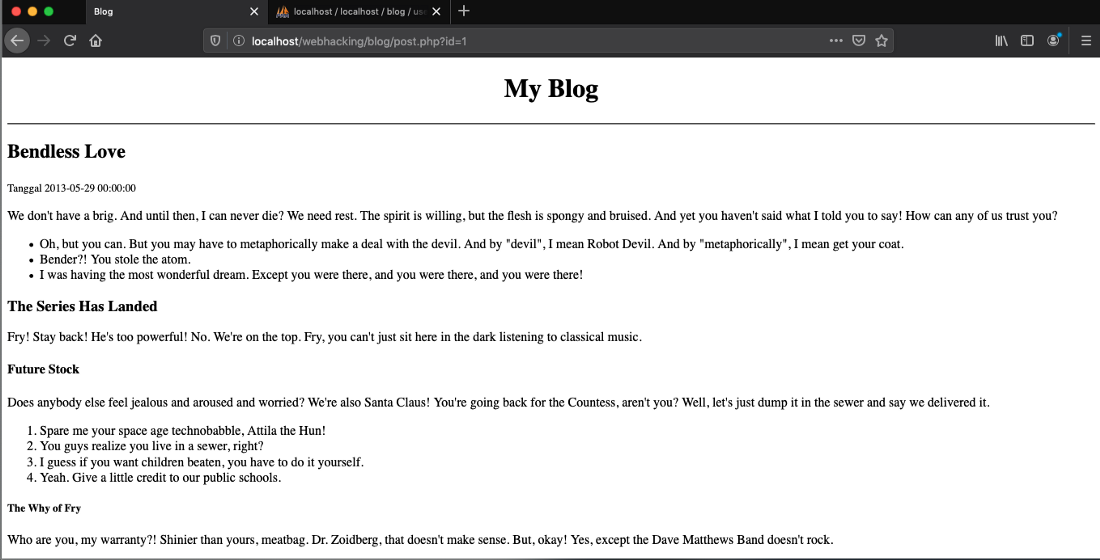
Dolayısıyla hata mesajları yardımı ile veritabanının sürüm bilgisinin okunması gerçekleşti.Buna benzer şekilde test parametresinin @@version olan alanından sunucu üzerinde istediğiniz verileri okuma imkanı elde edersiniz.SELECT convert(int,user)vb. gibi.İstismarın genişletilmesi size kalmış.

1. Kör (Blind) SQL Enjeksiyonu

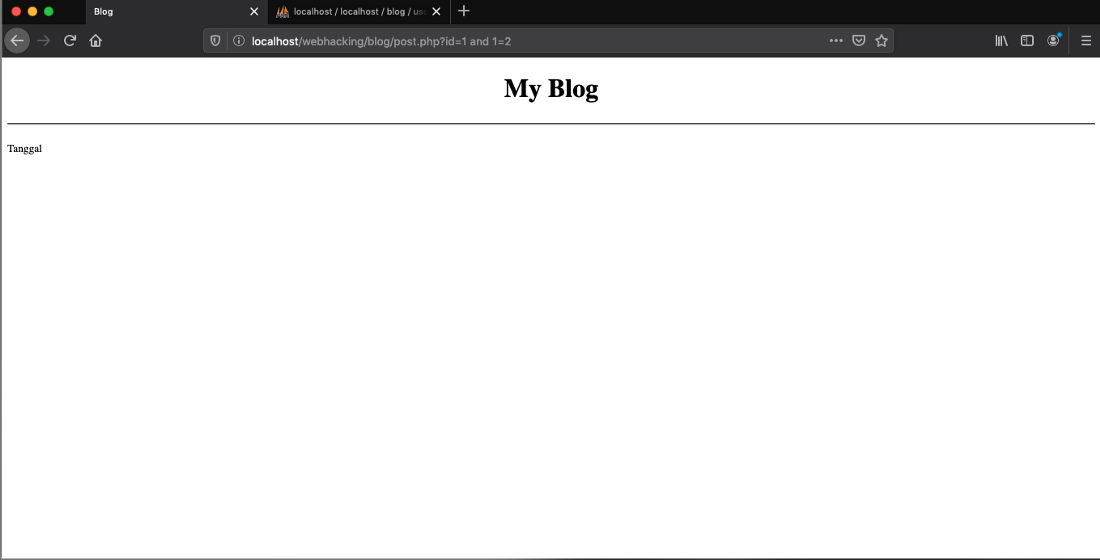
Günümüz uygulamaları yaygın olarak hata mesajlarını ayrıntılı olarak kullanıcıya göstermek yerine özelleştrilmiş bir hata mesajı veya sayfası dönmeyi tercih etmektedirler.Hatta bu özellik çoğu güncel uygulama/web sunucusunda ön tanımlı olarak gelmektedirler.Bu durumda hata mesajlarından yola çıkılarak SQL Enjeksiyonu zafiyetinin tespiti ve istismarı yöntemleri yetersiz kalmaktadır.İşte tam da bu esnada kör SQL Enjeksiyonu yöntemi imdada yetişir

* 1. Tespiti

Kör SQL Enjeksiyonu,uygulamanın hata mesajlarının bakmak yerime zafiyet barındıran ilgili servisin/sayfanın manipülasyonlara verdiği cevapların doğru/yanlış(true/false) olması ile ilgilenir.Bir Örnek ile durumu izah edecek olursak ; Bir blog sitemiz olsun aşağıdaki gibi

**Şekil 5 : Blind Injection (Kaynak 4 )**

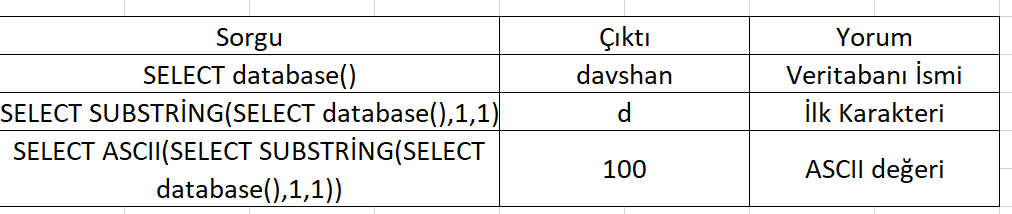
Eğer id parametresinde kör SQL Enjeksiyonu zafiyeti konusunda bir tespitimiz olacak ise (herhangi bir hata mesajı alınmadığını varsayarsak) aşağıdakine benzer şekilde doğru/yanlış(true/false) bulmamız gerekecektir.

**Şekil 4 : Blind Injection 2 (Kaynak 4 )**

Yukardaki istek için dönen cevap bize false olduğu için artık burada bir Kör SQL Enjeksiyonu var diyebiliriz.

* 1. İstismarı

Kör SQL Enjeksiyonu zafiyetinin istismarı diğer SQL Enjeksiyon zafiyetleri gibi hızlıca meyvelerini vermez.Fakat teorisi iyi anlaşıldıktan sonra istismarı otomatize edecek adımlardan bahsediyor olacağız.Varsayalım ki hedef uygulamamız bir MySQL veritabanı sunucusu ile çalışıyor olsun.MySQL sunucularında veritabanı adını çekmek için SELECT database() SQL sorgusu kullanılmaktadır.İlgili sorguyu aşağıdaki adımları ile birlikte MySQL sunucusu üzerinde çalıştırıp sonuçlarını inceleyelim.



Dolayısıyla ilgili sorguları kullanarak veritabanından çekeceğimiz herhangi bir verinin karakterlerinin ASCII değerlerine erişme gibi bir imkanımız olmaktadır. 5.1‘de de aslında nasıl istismar ediceğimiz hakkında bahsetmiştik.

1. Zaman Tabanlı (Time Based) SQL Enjeksiyonu

Şu ana kadar ele aldığımız SQL Enjeksiyonu zafiyetlerinde alınan hata mesajları, kullanıcıya gösterilen verilerin değişimi veya uygulamanın yanlışı doğru gibi bir izlenim bırakması sonucu istismarlarımız gerçekleşti.Fakat kimi zaman uygulamalar bu bahsettiğimiz davranışları sergilemezken yine de SQL Enjeksiyonu zafiyetine sahip olabilirler.Zaman Tabanlı SQL Enjeksiyonu zafiyetinde,yapılan isteğe karşın sunucudan dönen cevabın içeriği yerine cevabın dönüş süresi (response time ) göz önünde bulundurularak zafiyetin varlığı veya yokluğundan söz edilmektedir. Bu testin yapılmasına olanak sağlayan ise veritabanlarının zaman içerikli fonksiyonları/komutları veya bazı ağır çalışan sorgularıdır(heavy quaries).

* 1. Tespiti

Tespiti için hedef uygulamanın kullanmış olduğu veritabanının zaman içerikli fonksiyonları veya komutlarının bilinmesi veya tahmin edilmesi gerekmektedir.Zaten bu durum için karşılaşılabilecek çok da seçenek yoktur .Örneğin MSSQL veritabanları için WAITFOR DELAY komutu yardımıyla veritabanına belirtilen süre kadar bekleme yaptırmak mümkün olabilmektedir.

Dolayısıyla test edilen girdi noktası diğer zafiyet çeşitlerinde olduğu gibi içerik farklılığı veya hata üretmesi yerine sadece dönüş süresinin ölçümüyle saptanmaktadır.Tabi bu dönüş süresi test parametrelerinde kullanılan zaman ile neredeyse aynı değerde (+1,2 saniye sapma kabul edilebilir)olmalıdır.

* 1. İstismarı

İstismar ağırlıklı olarak Kör SQL Enjeksiyonuna benzemektedir.Çünkü Kör SQL Enjeksiyonunda yapılan isteğin sonucunda kullanıcının doğru/yanlış içerik görmesiyle ilgilenirken,Zaman Tabanlı SQL Enjeksiyonu yapılan isteğe karşın cevabın dönüş süresinin belirtilen saniyeden fazla veya az olması ile ilgilenmetedir.

KAYNAKÇA

1. <https://knowledge-base.secureflag.com/vulnerabilities/inadequate_input_validation/file_inclusion_vulnerability.html>
2. <https://infosecwriteups.com/exploiting-error-based-sql-injections-bypassing-restrictions-ed099623cd94>
3. <https://www.exploit-db.com/docs/english/44348-error-based-sql-injection-in-order-by-clause-(mssql).pdf>
4. <https://medium.com/@nyomanpradipta120/blind-sql-injection-ac36d2c4daab>
5. Ethical Hacking : Offensive & Defensive | Ömer Çıtak
6. Yazılım Güvenliği : Saldırı Savunma | Bünyamin Demir