Bunyamin Demir, <bunyamindemir at gmail dot com>,
webguvenligi.org, 24/01/2011

ORACLE VERİTABANINDAKİ HASSAS VERİLERİN GÜVENLİĞİ (TRANSPARAN VERİ ŞİFRELEME)

Veritabanı güvenliği denilince, ilk başlarda akla hak yükseltme, oturum bilgisi çalma, servis dışı bırakma v.b. saldırılar gelse bile, veritabanının kullanmış olduğu disk ve dosyaların elde edilmesi sonucu ortaya çıkabilecek zafiyetler de söz konusudur. Daha önce yazmış olduğum bir makalede [1] Oracle veritabanı trafiğinin güvenliğinden bahsetmiştim. Dolayısıyla belirtilen makale yardımıyla hassas verinin bilgisayar ağları üzerinde hareketini güvenli hale getirmiş olduk. Oracle veritabanı barındırdığı verileri işletim sistemi üzerinde bir dosya sisteminde tutmaktadır. Dolayısıyla bu dosyalara yapılacak olan yetkisiz erişimler verilerin çalınmasına sebep olabilir. Bu yüzden özellikle hassas verilerin (müşteri bilgileri, kredi kartı bilgileri v.b.) veritabanı içerisinde şifrelenerek tutulması gerekmektedir. Bu sayede bir önceki makalede [1] göz önünde bulundurularak, hassas verinin hem saklandığı yerde hem de son kullanıcının önüne sunulana kadar olan yaşam döngüsü içerisinde güvenliğini sağlamış olacağız.

Hassas verilerin şifrelenmesi için Oracle'da birden fazla yol izlenebilir. Özellikle Oracle ASO ile birlikte gelen "transparan veri şifreleme" (Transparent Data Encryption – TDE) metodu yaygın olarak kullanılmaktadır. TDE verinin disk üzerinde şifreli olarak tutulmasını sağlamaktadır. Bu yüzden veri veritabanından SELECT, INSERT, UPDATE gibi isteklere maruz kaldığı anda, düz metine çevirme işlemine tabi tutulur. Dolayısıyla hassas veri, yine veritabanı yöneticileri tarafından görüntülenebilir, değiştirilebilir, silinebilir. Fakat özellikle disk çalınması, veritabanı dosyalarının ele geçirilmesine sebep olan saldırılar sonucu, hassas bilgi yetkisiz kullanıcılar tarafından elde edilebilir olmayacaktır.

Hassas verinin şifreleme işleminin de bir CPU maliyeti vardır. Fakat Oracle 11G ile yapılan iyileştirmeler sonucu, bu maliyet kabul edilebilir seviyelere indirilmiştir. Bu sebepten dolayı dosya sistemi üzerinde tutulan hassas verilerin şifrelenmesi için TDE diğer yöntemlerden bir adım önde olmaktadır.

Örnek bir TDE işlemi yapılacak olursa; öncelikle şifreleme esnasında kullanılacak anahtarın saklanacağı dizinin oluşturulması gerekmektedir. TDE bu işi Oracle Wallet yardımı ile yapmaktadır. Bir dizin oluşturulup, bu dizinde şifreleme sırasında kullanılacak anahtarlar saklanır.

```
mdkir /opt/oracle/admin/oratest/wallet
```

Dizin oluşturduktan sonra şifreleme için kullanılacak anahtar oluşturulur.

```
SQL> ALTER SYSTEM SET ENCRYPTION KEY IDENTIFIED BY "1Bny456";
System altered.
```

Anahtarın oluşturulduğuna dair kontrol yapılır.

```
[oracle@oracle dbhome_1]$ ls -l /opt/oracle/admin/oratest/wallet/
total 4
-rw-r--r-- 1 oracle oinstall 1573 May  2 13:31 ewallet.p12
```

Şifreleme işlemi için anahtarı oluşturduktan sonra şifreleme yapılacak öğeyi belirtmek gerekmektedir. TDE ile şifreleme işlemleri tablo bazında yapılacağı gibi, tablonun bir öğesine kadar da indirgenebilir. Yapılacak transparan veri şifreleme örneği için "KREDI_KARTI" adında bir tablo oluşturup, kredi kartı numarasının tutulduğu "KARTNO" alanı şifreli olarak saklayalım.

```
CREATE TABLE KREDI_KARTI (
"RECORDID" NUMBER NOT NULL ENABLE,
"AD" VARCHAR2(50 BYTE),
"SOYAD" VARCHAR2(50 BYTE),
"KARTNO" VARCHAR2(50 BYTE) ENCRYPT,
CONSTRAINT "KREDI_KARTI_PK" PRIMARY KEY ("RECORDID"));
```

Tablo oluşturulduktan sonra, tablonun özelliklerine bakıldığında KARTNO alanı için şifreleme (ENCRYPT) özelliğinin varlığı görülecektir.

SQL> desc bunyamin.KREDI_KARTI;

Name	Null?	Type
RECORDID		NOT NULL NUMBER
AD		VARCHAR2(50)
SOYAD		VARCHAR2(50)
KARTNO		VARCHAR2(50) ENCRYPT

Tabloya veri ekleme işlemi için:

INSERT INTO KREDI_KARTI (RECORDID, AD, SOYAD, KARTNO) VALUES ('1', 'bunyamin', 'demir', '1111111111111111');

INSERT INTO KREDI_KARTI (RECORDID, AD, SOYAD, KARTNO) VALUES ('2', 'ahmet', 'ordu', '222222222222222');

KARTNO alanı ENCRYPT değerine sahip olduğu için. Bu alana yazılmış olan “1111111111111111”, “222222222222222” değerleri şifreli olarak tutulacaktır.

Ardından tablonun herhangi bir kolonunu şifrelemek için:

SQL> ALTER TABLE KREDI_KARTI MODIFY KARTNO ENCRYPT;
Table altered.

Tablonun herhangi bir kolonunun şifresini çözmek (decryption) için:

SQL> ALTER TABLE KREDI_KARTI MODIFY KARTNO DECRYPT;
Table altered.

Şifreleme algoritmasını değiştirmek için:

SQL> ALTER TABLE KREDI_KARTI REKEY USING 'AES256';
Table altered.

komutları uygulanabilir. Bu sayede veriler disk üzerinde şifreli olarak saklanacaktır.

Eğer hem trafik hem de dosya sistemi üzerinde veri şifrelenmiş ise, yapılan istek karşısında, veri önce düz metin haline getirilip daha sonra trafiğin şifreleme yöntemiyle tekrar şifrelenip, istemciye iletilecektir.

Kaynaklar:

1) http://www.webguvenligi.org/docs/Oracle_Veritabani_Trafiginin_Guvenligi.pdf