KOCAELİ ÜNİVERSİTESİ * FEN BİLİMLERİ ENSTİTÜSÜ

SQL VERİTABANI SİSTEMLERİNDE GÜVENLİK VE ÖRNEK UYGULAMA

YÜKSEK LİSANS

Muhsin YAZICI

Anabilim Dalı: Bilgisayar Mühendisliği

Danışman: Yrd. Doç. Dr. H.Engin DEMİRAY

KOCAELİ, 2008

KOCAELİ ÜNİVERSİTESİ * FEN BİLİMLERİ ENSTİTÜSÜ

SQL VERİTABANI SİSTEMLERİNDE GÜVENLİK VE ÖRNEK UYGULAMA

YÜKSEK LİSANS TEZİ

Muhsin YAZICI

Tezin Enstitüye Verildiği Tarih: 08 Ekim 2008

Tezin Savunulduğu Tarih: 26 Kasım 2008

Tez Danışmanı	Üye	Üye
Yrd. Doç. Dr. H.Engin DEMİRAY	Prof. Dr. Ali OKATAN	Doç. Dr. Nevcihan DURU
(.....)	(.....)	(.....)

KOCAELİ, 2008

ÖNSÖZ VE TEŞEKKÜR

Oğlum Bedri Ömer'e…

Günümüz dünyasının temel hammaddelerinden olan verinin elektronik ortamlarda güvenli bir şekilde saklanması, hızlı erişiminin yapılabilmesi, kullanılması ve paylaşılması veritabanı sistemleri aracılığıyla gerçekleştirilmektedir. Bu işlemlerin yapılması esnasında meydana gelebilecek güvenlik sorunlarına karşılık veri ve veritabanı güvenliğinin sağlanması için etkili ve yeni yaklaşımların geliştirilmesi mecburiyeti ortaya çıkmaktadır. Veritabanlarındaki verinin güvenliğinin çok iyi bir seviyede sağlanması için "Veritabanı Güvenliği" konusunun iyi bir şekilde kavranması, planlanması, uygulanması ve tüm veritabanı kullanıcılarının bilinçlendirilmesi gerekir. Bu sayede, elektronik ortamlarda işlemler daha güvenilir yapılırken, bilginin kullanımı yaygınlaşacak, işlemler daha hızlı gerçekleşecek ve hemen hemen her alanda verimlilik artışı sağlanmış olacaktır.

Bu önemli konuda çalışmamı sağlayan, bana yol gösteren, destek ve yardımlarını esirgemeyen, konu hakkındaki bilgisinden yararlandığım danışman hocam Sayın Yrd. Doç. Dr. H.Engin DEMİRAY'a sonsuz teşekkürlerimi sunmak isterim.

İÇİNDEKİLER

ÖNSÖZ VE TEŞEKKÜR i
İÇİNDEKİLER ii
ŞEKİLLER DİZİNİ v
TABLOLAR DİZİNİ vi
SEMBOLLER vii
ÖZET viii
İNGİLİZCE ÖZET ix
1. GİRİŞ 1
1.1. Literatürde Yapılan Çalışmalar 3
1.2. Tez Çalışmasının Amacı 3
1.3. Tez Çalışmasının Katkıları 4
1.4. Tez Organizasyonu 4
2. VERİ GÜVENLİĞİ 5
2.1. Veri Güvenliğine Giriş 5
2.2. Güvenlik Prensipleri 5
2.2.1. Gizlilik (Confidentiality) 5
2.2.2. Veri bütünlüğü (Data Integrity) 5
2.2.3. Erişilebilirlik (Availability) 5
2.2.4. Denetim (Accountability) 6
2.2.5. Kullanıcı doğrulama (Authentication) 6
2.2.6. Güvenilirlik (Reliability - Consistency) 6
2.2.7. İnkâr edememe (Non-repudiation) 6
3. VERİ GÜVENLİK YÖNETİMİ MODELLERİ 8
3.1. Erişim Denetimi 8
3.1.1. Tanımlama (Identification) 8
3.1.2. Kullanıcı doğrulama (Authentication) 9
3.1.3. Yetkilendirme (Authorization) 9
3.2. Şifreleme 10
3.2.1. Simetrik anahtarlı şifreleme sistemleri 11
3.2.1.1. Blok şifreleme 12
3.2.1.2. AES (Advanced Encryption Standard) 13
3.2.1.3. DES (Data Encryption Standard) 13
3.2.1.4. Triple-DES (3-DES) 14
3.2.2. Asimetrik anahtarlı şifreleme sistemleri 14
3.2.2.1. RSA şifreleme algoritması 15
3.2.3. Özetleme fonksiyonları 17
3.2.4. Sayısal imza ve açık anahtar altyapısı (AAA) 17
3.2.4.1. Sayısal sertifikalar 18
3.2.4.1.1. X.509 Sertifikalar 19
3.2.4.2. Sertifikasyon otoriteleri 20
3.3. Uygulama Güvenliği 20
3.4. İşletim Sistemi Güvenliği 21
3.5. Ağ ve İnternet Güvenliği 21

3.5.1. Güvenlik duvarları (Firewalls)....22
3.5.2. Saldırı tespit sistemleri....23
3.5.2.1. Ağ tabanlı saldırı tespit sistemleri....24
3.5.2.2. Sunucu tabanlı saldırı tespit sistemleri....25
3.5.3. Sanal özel ağ (VPN)....25
3.5.4. Kerberos....26
3.5.5. SSL (Secure Sockets Layer)....29
3.6. Fiziki Güvenlik....29
3.7. Yedekleme ve Kurtarma....30
4. VERİTABANLARINDA GÜVENLİK....31
4.1. Microsoft SQL Server Güvenliği....31
4.1.1. Kullanıcı doğrulama (Authentication)....31
4.1.1.1. Şifre politikası (Password Policy) sistemi....31
4.1.2. Yetkilendirme (Authorization)....32
4.1.2.1 Kullanıcılar (Principals)....33
4.1.2.2. Emniyete alınabilir varlıklar (Securables)....33
4.1.2.3. İzinler (Permissions)....33
4.1.2.4. Roller ve şemalar....34
4.1.2.5. Kullanıcı - şema ayrımı....36
4.1.2.6. Çalıştırma içeriği (Execution Context)....37
4.1.2.7. Katalog güvenliği....39
4.1.2.8. Ağ üzerinde şifreleme (Network encryption)....40
4.1.3. Veri şifreleme (Data Encryption)....40
4.1.3.1. Veri şifreleme fonksiyonları....46
4.1.3.2. Saydam Veri Şifreleme (TDE, Transparent Data Encryption)....50
4.1.4. Hesap Denetimi....52
4.2. Oracle Veritabanı Güvenliği....52
4.2.1. Doğrulama (Authentication)....52
4.2.1.1. İşletim sistemiyle kullanıcı doğrulama....53
4.2.1.2. Ağ (Network) yoluyla kullanıcı doğrulama....54
4.2.1.2.1. Üçüncü parti tabanlı kullanıcı doğrulama....54
4.2.1.2.2 Açık anahtar altyapısı tabanlı kullanıcı doğrulama (PKI)....54
4.2.1.2.3. Kerberos kullanarak kullanıcı doğrulama....54
4.2.1.2.4. Uzaktan kullanıcı doğrulama....54
4.2.1.2.5. Dizin tabanlı servisleri kullanarak kullanıcı doğrulama....55
4.2.1.3. Oracle veritabanı yoluyla kullanıcı doğrulama....55
4.2.1.4. Çok katmanlı kullanıcı doğrulama....57
4.2.1.5. Veritabanı yöneticilerini doğrulama....58
4.2.2. Yetkilendirme (Authorization)....58
4.2.2.1. Kullanıcı kaynak sınırlandırmaları ve profiller....59
4.2.2.2. Sistem kaynakları ve sınırlamalar....59
4.2.2.3. Profiller....60
4.2.2.4. Haklar....61
4.2.2.4.1. Sistem hakları....62
4.2.2.4.2. Nesne hakları....62
4.2.2.5. Roller....65
4.2.3. Sanal kişisel veri tabanı (Virtual Private Database)....66
4.2.4. Veri şifreleme....67
4.2.4.1. Saydam Veri Şifreleme (Transparent Data Encryption)....68

4.2.4.2. Veri şifreleme API'lerini kullanarak şifreleme....73
4.2.4.2.1. DBMS_CRYPTO paketini kullanarak veri şifreleme....73
4.2.4.2.2. Veri bütünlüğü için DBMS_SQLHASH paketinin kullanılması....75
4.2.5. Hesap Denetimi....76
4.2.5.1. Veritabanı bazlı hesap denetimi....76
4.2.5.2. Uygulama bazlı hesap denetimi....78
4.3. IBM DB2 Güvenliği....78
4.3.1. Doğrulama (Authentication)....78
4.3.1.1. Uzak istemciler için doğrulama....81
4.3.2. Yetkilendirme....82
4.3.2.1. Yetkiler, haklar ve yetkilendirme....84
4.3.2.1.1. Sistem yönetim yetkisi (SYSADM)....84
4.3.2.1.2. Sistem kontrol yetkisi (SYSCTRL)....85
4.3.2.1.3. Sistem bakım yetkisi (SYSMAINT)....85
4.3.2.1.4. Güvenlik yönetim yetkisi (SECADM)....86
4.3.2.1.5. Veritabanı yönetim yetkisi (DBADM)....86
4.3.2.1.6. Sistem izleme yetkisi (SYSMON)....87
4.3.2.1.7. LOAD yetkisi....88
4.3.2.1.8. Veritabanı yetkileri....88
4.3.2.1.9. Şema yetkisi (IMPLICIT_SCHEMA)....89
4.3.2.1.10. Şema hakları....89
4.3.2.1.11. Tablespace hakları....90
4.3.2.1.12. Tablo ve görüntü hakları....90
4.3.2.1.13. Paket hakları....90
4.3.2.1.14. Index hakları....91
4.3.2.1.15. Sequence hakları....91
4.3.2.1.16. Routine hakları....91
4.3.3. Veri Şifreleme....91
4.3.4. Etiket tabanlı erişim kontrolü (LBAC)....93
4.3.5. Hesap denetimi....95
5. VERİTABANLARINDAKİ RİSKLER....98
5.1. Güvenlik Boşlukları....98
5.2. Temel Saldırı Metotları....99
5.2.1. Ara bellek taşması (Buffer overflow)....100
5.2.2. SQL enjeksiyonu....101
5.2.2.1. SQL enjeksiyonundan korunma yolları....106
6. DAHA İYİ BİR VERİTABANI GÜVENLİĞİ İÇİN NELER YAPILMALI....108
6.1. SQL Server Veritabanı İçin Yapılması Gerekenler....109
6.2. Oracle Veritabanı İçin Yapılması Gerekenler....110
6.3. DB2 Veritabanı İçin Yapılması Gerekenler....110
6.4. Veritabanları İçin Diğer Yapılması Gerekenler....111
6.4.1. Veritabanı yamalarını yükleme....111
6.4.2. Güvenlik bildirilerini izlemek....111
7. TEZ KAPSAMINDA YAPILAN YAZILIM UYGULAMALARI....113
7.1. RSA Şifreleme Uygulaması....113
7.2. SQL Server Veritabanı Şifreleme Uygulaması....119
8. SONUÇ....129
KAYNAKLAR....136
ÖZGEÇMİŞ....140

ŞEKİLLER DİZİNİ

Şekil 3.1: Kerberos doğrulamanın çalışma şekli. 26
Şekil 4.1: SQL Server 2005 Veritabanı sahiplik zinciri.....37
Şekil 4.2: SQL Server 2005 sahiplik mimarisi örneği. 38
Şekil 4.3: SQL Server 2005 çalıştırma içeriği (execution context) örneği 39
Şekil 4.4: SQL Server veri şifreleme mimarisi. 40
Şekil 4.5: SQL Server veri şifreleme örneği (şifrelenmiş).....49
Şekil 4.6: SQL Server veri şifreleme örneği. 49
Şekil 4.7 : SQL Server Veritabanında Saydam Veri Şifreleme (TDE) 50
Şekil 4.8: Oracle veritabanında GRANT deyiminin kullanılışı. 63
Şekil 4.9: Oracle veritabanında GRANT deyiminin sistem hakları için kullanılışı. ..63
Şekil 4.10: Oracle veritabanında GRANT deyiminin nesne hakları için kullanılışı. .63
Şekil 4.11: Oracle veritabanında GRANT deyiminin nesne üzerinde kullanılışı. 64
Şekil 4.12: Oracle veritabanında GRANT deyiminin kullanıcı veya rol için kullanılışı. 64
Şekil 4.13: Oracle veritabanında örnek rol dağılımı. 66
Şekil 4.14: Saydam veri şifreleme. 69
Şekil 7.1: RSA programının çalışma şekli 114
Şekil 7.2: RSA programında şifreleme. 115
Şekil 7.3: RSA programında şifrelenmiş mesajın iletilmesi. 116
Şekil 7.4: RSA şifrelemede mesajın sahibi tarafından deşifre edilmesi. 117
Şekil 7.5: RSA şifrelemede mesajın farklı kullanıcılarca ele geçirilmesi.....118
Şekil 7.6: RSA şifrelemede mesajın farklı kullanıcılarca deşifre edilmesi.....118
Şekil 7.7: SQL Server Veritabanı Şifreleme Programı. 119
Şekil 7.8: SQL Server Veritabanı doğrulama yöntemi. 120
Şekil 7.9: SQL Server veritabanında yeni bir veritabanı oluşturma. 120
Şekil 7.10: SQL Server da veritabanı içerisinde yeni bir tablo oluşturma.....121
Şekil 7.11: Veritabanında seçili (aktif) tabloya ait sütunlar.....121
Şekil 7.12: SQL Server da veritabanı master anahtarı oluşturma.....122
Şekil 7.13: SQL Server da şifrelemede kullanılmak üzere sertifika oluşturma. 122
Şekil 7.14: SQL Server simetrik anahtar oluşturma. 123
Şekil 7.15: SQL Server da asimetrik anahtar oluşturma. 124
Şekil 7.16: SQL Server da veritabanı master anahtarının açılması. 125
Şekil 7.17: SQL Server veritabanında sertifika kullanılarak şifreleme. 125
Şekil 7.18: SQL Server veritabanında sertifikayla şifreleme çıktısı.....126
Şekil 7.19: SQL Server veritabanında simetrik anahtar kullanılarak şifreleme. 126
Şekil 7.20: SQL Server veritabanında simetrik anahtar şifreleme çıktısı. 127
Şekil 7.21: SQL Server veritabanında asimetrik anahtar kullanılarak şifreleme. 127
Şekil 7.22: SQL Server veritabanında asimetrik anahtar şifreleme çıktısı. 127
Şekil 7.23: SQL Server veritabanında deşifrelenmiş sütun.....128

TABLOLAR DİZİNİ

Tablo 3.1: Simetrik ve asimetrik şifreleme algoritmalarının karşılaştırması.....11
Tablo 4.1: SQL Server veritabanında izinler.....34
Tablo 4.2. Oracle veritabanında nesne hakları. 62
Tablo 4.3: DBMS_CRYPTO ve DBMS_OBFUSCATION_TOOLKIT
Karşılaştırması. 74
Tablo 8.1: Veritabanları güvenlik özellikleri karşılaştırması.....130

SEMBOLLER

Kısaltmalar

AAA	: Açık Anahtar Altyapısı
AS	: Authentication Server
ASP	: Active Server Page
DAS	: DB2 Administration Server
DCL	: Data Control Language
DML	: Data Manipulation Language
GPO	: Group Policy Object
KDC	: Key Distribution Center
LBAC	: Label-based access control
LDAP	: Lightweight Directory Access Protocol
LOB	: Large Object.
NTLM:	: NT LAN Manager
SQLDA	: SQL Descriptor Area.
SSO	: Single Sign-On
PKI	: Public Key Infrastructure
RADIUS	: Remote Authentication Dial-In User Service
TDE	: Transparent Data Encryption
TGS	: Ticket Granting Server
VPD	: Virtual Private Database
VPNs	: Virtual Private Networks

SQL VERİTABANI SİSTEMLERİNDE GÜVENLİK

VE

ÖRNEK UYGULAMA

Muhsin YAZICI

Anahtar Kelimeler: Veritabanı Güvenliği, Kullanıcı doğrulama, Yetkilendirme, Hesap Denetimi, Şifreleme, SQL Server Veritabanı Güvenliği, Oracle Veritabanı Güvenliği, DB2 Veritabanı Güvenliği.

Özet: Bu tezde, genel olarak veri güvenlik yöntemleri, büyük veritabanı sistemlerinin güvenlik özellikleri, veritabanlarındaki riskler ve veritabanlarını sağlamlaştırma konuları detaylı olarak incelenmiştir. Veritabanları güvenlik yapılarını, kullanıcı doğrulama, kullanıcı yetkilendirme, veritabanındaki verinin şifrelenmesi ve kullanıcı denetiminin yapılması gibi ilkeler üzerinde oturtmuşlardır.

Kullanıcı doğrulama; işletim sistemi yoluyla, ağ üzerinden, çok katmanlı olarak, üçüncü parti araçlar/protokoller (SSL, Kerberos, dizin tabanlı servisler vb.) ve veritabanının kendi doğrulama mekanizması kullanılarak yapılabilmektedir.

Kullanıcının yetkilendirilmesi yetkiler, roller, şemalar, izinler, haklar, kaynak sınırlandırmaları, profiller aracılığıyla veritabanı sunucusu ve nesne seviyesinde yapılabilmektedir.

Kullanıcıların hesap denetimi ise, önceden tanımlanmış veritabanı olayları veya tüm işlemler için, bilgiler üretilerek, veritabanı ve uygulama bazlı olarak yapılabilmektedir. Her bir veritabanı ortamı kendine has denetim mekanizması ve araçlarına sahiptir.

Veritabanındaki verilerin şifrelenmesi için Oracle saydam veri şifreleme, SQL Server sertifikalar ve anahtarlar yardımıyla şifreleme, DB2 ise kendi şifreleme fonksiyonlarıyla şifreleme gibi yöntemler kullanmaktadır.

Ayrıca veritabanlarının güvenliğini sağlamak için işletim sistemi güvenliğinin, ağ ve internet güvenliğinin, uygulama güvenliğinin, fiziki güvenliğin sağlanması, veritabanı güvenlik duyuru ve yamalarının takip edilip uygulanması gerekmektedir.

SECURITY IN SQL DATABASE SYSTEMS

AND

EXAMPLE APPLICATION

Muhsin YAZICI

Keywords: Database Security, Authentication, Authorization, Auditing, Encryption, SQL Server Database Security, Oracle Database Security, DB2 Database Security.

Abstract: In this thesis, the data security methods, the security characters of huge database systems, risks in databases and topic of hardening databases are generally analysed in detail. Databases puts their security structures on the principles such as user confiarmation authorization, authentication, auditing, encryption**.**

Authentication can be done through operating system on network as multi-layer using third party appliances (SSL, Kerberos, directory based systems etc.) and the confirmation mechanism of database itself.

User authorization can be done through authorities, roles, schemas, permissions, rights, resource restrictions, profiles on the level of object and database server.

Auditing on the other hand can be done database and application-based producing information for pre-defined database events and all operations. Each database environment has its own auditing mechanism and appliances.

For the encryption of data in database methods such as Oracle Transparent Data Encryption, SQL Server Certificates, Encryption with Key, and Self-functioned Encryption in DB2 are used.

Also, for the security of databases it is necessary to enable the security of operating system, the security of network and internet, application security, physical security and to follow and practice the database security notices and mends.

1. GİRİŞ

E-uygulamalar günümüz dünyasının ayrılmaz bir parçası olmuş olup her geçen gün kullanım, çeşitlilik ve fonksiyonel bakımdan artış göstermektedirler. Başta e-ticaret, e-iş ve e-devlet olmak üzere artık her alanda internet üzerinden yapılan işlemler yaşamımızı değiştirmektedir.

Gerek e-uygulamalarda gerekse diğer uygulamalarda, veri veya bilginin saklanması gereken her yerde kullanılması zorunlu hale gelen veritabanları, çok büyük miktarlardaki verilerin güvenli bir şekilde tutulabildiği, hızlı erişimin yapılabildiği, veri bütünlüğünün sağlanabildiği ve birçok kullanıcıya aynı anda hizmet verebilen ortamlar olarak karşımıza çıkmaktadırlar.

Önceleri kurum veya şirkete ait veritabanlarına uygulamalar aracılığıyla sadece kendi çalışanları erişebilmekteydi. Bugün ise, dünyanın her tarafından kişiler, veritabanıyla alakaları olmamalarına rağmen, veritabanını kullanan kişiler haline gelmişlerdir. Bu da artık veritabanlarının kullanıcılarla daha içli dışlı olmasına yol açmaktadır.

Hacker diye adlandırılan, bilgisayar sistemlerini ele geçirmeyi veya zarar vermeyi amaçlayan kişiler, artık network ve işletim sistemi saldırılarından sonra veritabanları dolayısıyla da veritabanlarının içindeki önemli verileri hedef almaktadırlar. Hacker forumları ve bazı güvenlik firmalarının bültenleri, veritabanlarının içerdikleri hassas veriler nedeniyle yeni bir hedef olarak, uğradıkları saldırıların sayısında önemli ve sürekli bir artış olduğunu göstermektedir.

Veritabanı sistemlerine yapılmış bazı saldırılar ve verdikleri zararlar şunlardır: [35]

- 2000 yılında, CD Universe online müzik satıcısı, "Maxus" adında bir hacker tarafından saldırıya uğramıştır. Hacker, satıcının veritabanındaki kredi kartı numaralarını çalmış olup karşılığında para istemiştir. Talepleri reddedilince de, binlerce müşterinin kredi kartı numarasını internette yayımlamışlardır.

- 2000 Aralıkta, Egghead.com online satış sitesi, müşteri veritabanının yetkisiz kişilerin eline geçtiğini açıklamış ve 3.5 milyondan fazla kredi kartı numarasının çalınmış olabileceğini açıklamıştır. Egghead.com, daha sonra yaptırdığı araştırma ve inceleme sonucunda, kredi kartı numaralarının çalınmadığını açıklamış fakat yapılan bu araştırma soruşturma maliyetinin milyonlar aldığını belirtmiştir. Bu olaydan kısa süre sonra da şirket iş dünyasından çekilmiştir.
- 2001'de, Amazon.com'un bir parçası olarak faaliyet gösteren, az bulunur ve baskısı tükenmiş kitapların yer aldığı Bibliofind, saldırıya uğramıştır ve 100.000 adet kredi kartı numarası çalınmıştır. Saldırganlar fark edilene kadar, 4 ay boyunca veritabanına erişim sağlamışlardır. Daha sonra, Bibliofind, tüm satış servislerini durdurmuştur.
- Mart 2001'de, FBI, yaklaşık 50 banka ve satıcı web sitesinin Rus ve Ukraynalı hackerlar tarafından saldırıya uğradığını açıklamıştır.
- 2001'de, India Üniversitesi 2 kez saldırıya uğramış, öğrenci ve personelin sosyal güvenlik numaraları ve adresleri gibi kişisel bilgileri çalınmıştır.
- 2004 Ekimde, bir hacker, Kaliforniyada yaşayan 1.4 milyondan fazla kişinin kişisel bilgisini içeren veritabanını ele geçirmiştir. Bu olay, 2005 Ağustosuna kadar fark edilememiştir. Veritabanında, kişilerin isimleri, adresleri, sosyal güvenlik numaraları ve doğum tarihleri gibi hassas bilgileri vardı.

Kurumları her yıl milyarlarca dolarlık zarara uğratan, veritabanı güvenliğini tehlikeye sokan tehditlere karşı korkup, internet ve ağ teknolojilerinden, e-uygulamalardan uzak kalmanın getireceği alternatif maliyet, işgücü ve zaman kaybının zararı çok daha yüksek olacaktır. Önemli olan ise, tehditleri minimuma indirecek veritabanı güvenlik altyapısını tesis edebilmektir.

Veritabanı güvenliği nedir? Veritabanı güvenliği, veritabanını kasıtlı ve rastlantısal tehlikelere karşı koruyan mekanizmadır. [56] Bir başka deyişle, veritabanını, internet kaynaklı virüslere, güvenlik sistemlerine aykırı davranan çalışanlara ve bilinçli olarak yapılan saldırılara karşı korumaktır.

Bir veritabanının aşağıdakilere karşı korunması gerekir: [56]

- Erişim yetkisi olmayan kişilerin erişiminden,
- Veri, program ve donanım hırsızlarından,
- Virüs ve solucanların girişlerinden,
- Veri bozulması ve verinin bütünlüğünü kaybetmesinden,
- Yetersiz eğitime sahip personelden,
- Fiziki zararlardan.

Saldırının düzeyi, şekli ve bilgilerin türüne bağlı olarak, saldırılar önemli kayıplara sebep olabilir ve mevcut sistemi tamamen çalışmaz hale getirebilir. Kurumun büyük miktarlarda maddi kaybı olabileceği gibi telafisi mümkün olmayan veri, iş gücü ve zaman kayıplarına da yol açabilir. Yetkisiz veya kötü niyetli kişilerin, bilgi sistemlerinin zayıf noktalarından yararlanıp veritabanlarındaki önemli bilgiyi çalmaları, gizli bilgilere ulaşmaları, ulaştıkları bilgileri kötü amaçlı kullanmaları, değiştirmeleri, bozmaları, silmeleri, bilgiye yetkili olan kişilerin erişim ve kullanımını engellemeleri ve benzeri zararlı eylemler, veritabanı güvenliğine karşı değişik tür saldırı ve tehditleri oluşturur. Bunların yanında meydana gelebilecek arıza, kaza ve doğal afetler de veritabanı güvenliğini tehdit etmektedir.

Ayrıca veritabanı güvenliği, hem dışarıdan hem de içeriden yapılabilecek güvenlik saldırılarına karşı duyarlı olmak zorundadır. Genellikle mevcut kullanılan önlemler dış saldırılara karşı geliştirilmiştir. Güvenlik duvarları, virüs tarayıcılar, şifreli iletişim gibi teknikler dışarıdan gelen saldırılara karşı kullanıldığından iç kaynaklı saldırılara karşı koruyucu olmayabilir. Kurum içerisinden yapılan saldırılar, daha zor fark edilmeleri ve daha zor önlenebilmeleri nedeniyle daha zarar verici olabilirler.

Veritabanı güvenliği, kurumların her geçen gün önemi artan temel yönetim yapıtaşlarından biri durumuna gelmekte olup, başta veritabanı güvenlik yöneticileri olmak üzere tüm bilişim teknolojisi yöneticilerinin yeni stratejiler, önlemler ve yaklaşımlar oluşturmalarını gerekli kılmaktadır.

1.1. Literatürde Yapılan Çalışmalar

Veritabanı güvenliği ile ilgili çok sayıda makale, kitap vb. çalışma yapılmış olup bu çalışmalar genelde veritabanı güvenliğinin bir yönü üzerine odaklanmışlardır. Bu tez çalışması ise, her bir veritabanı yönetim sisteminin tüm güvenlik özelliklerini inceleyerek, geniş kapsamlı bir araştırma haline gelmesini hedeflemiştir. Veritabanı güvenliği ile ilgili yapılan çalışmalardan bazıları şunlardır: [35]

- 2007 yılında, Dr. Larry Ponemon tarafından yapılan çalışmada veritabanlarının yapılarındaki tehlike ve veritabanlarının öncelikleri incelenmiştir. Şirketler üzerinde yaptığı araştırma ve anketlerle kurumların önceliklerini, veritabanlarını kullanma nedenlerini, verilerini korumak için neler yaptıklarını incelemiştir.
- 2006 yılında, David Litchfield, Oracle ve SQL Server veritabanlarında ortaya çıkan güvenlik açıklarını karşılaştırmıştır.
- 2006 yılında Symantec tarafından yapılan çalışmada, veritabanı güvenliğini arttırmak için hesap denetiminin, rol bazlı erişimin ve ağ ortamında verinin şifrelenmesinin önemine dikkat çekilmiştir.
- 2005 yılında, Hassan A. Afyouni tarafından, veritabanlarındaki veri bütünlüğünün korunması ve erişimin denetlenmesi konuları üzerinde bir çalışma yapılmıştır.
- 2005 yılında, Joaquin A. Trinanes, veritabanlarının riskli ortamlarda güvenliklerinin nasıl artırılabileceği konusunda bir çalışma yapmıştır. İnternet ortamı, veri kayıpları, uygulama güvenliği ve saldırganların oluşturduğu tehlikeler üzerinde durulmuştur.

1.2. Tez Çalışmasının Amacı

Bu tez çalışmasının amacı, temel güvenlik gereksinimleri konusunda bilgi edinerek, büyük veritabanlarında kullanılan güvenlik yetenek ve özelliklerini kavramak, analizini yapmak, veritabanlarındaki riskleri ve tehlikeleri belirlemek, veritabanı güvenliğinin nasıl arttırılabileceğini ortaya koymak ve ülkemizde bu zamana kadar yapılmayan veritabanı güvenliğiyle ilgili çalışmalara örnek ve temel teşkil edecek bir çalışma olmaktır.

1.3. Tez Çalışmasının Katkıları

Bu tez çalışması, veri güvenlik metotları ve büyük veritabanı sistemlerinin güvenlik özellikleri hakkında bilgiler edinilmesini sağlamıştır. Veritabanlarındaki riskler, veritabanlarının kendi savunma mekanizmaları, şifreleme yöntemleri ve veritabanı güvenliğini sağlamlaştırma konularında bilgi verilmiştir. Veritabanları güvenlik yapılarının karşılaştırmalı analizi yapılmıştır.

1.4. Tez Organizasyonu

Tez çalışması sekiz ana başlıktan oluşmaktadır ve şu şekilde organize edilmiştir.

Bölüm 2'de, veri güvenliği ve veri güvenlik prensipleri hakkında bilgiler verilmiştir.

Bölüm 3'te, veri güvenlik yönetim modelleri anlatılmıştır. Veritabanlarının temel güvenlik elemanları olan kullanıcı tanımlama, doğrulama ve yetkilendirme yöntemleri açıklamıştır. Ayrıca veri şifreleme yöntemleri, sertifikalar, uygulama güvenliği, işletim sistemi güvenliği, ağ ve internet güvenliği, güvenlik duvarları, Kerberos ve SSL protokolleri incelenmiştir.

Bölüm 4'te, SQL Server, Oracle ve DB2 veritabanlarının güvenlik yapıları detaylı incelenmiştir. Veritabanı sistemlerinin veri şifreleme teknolojilerine değinilmiştir.

Bölüm 5'te, veritabanlarındaki riskler, güvenlik boşlukları anlatılmış ve temel saldırı yöntemlerinden SQL enjeksiyonu incelenmiştir.

Bölüm 6'da, daha iyi bir veritabanı güvenliğinin sağlanması için SQL Server, Oracle ve DB2 veritabanlarında yapılması gerekenler belirtilmiştir.

Bölüm 7'de ise, tez kapsamında yapılan yazılım uygulamaları anlatılmıştır.

Bölüm 8 ise, sonuç kısmı olup veritabanı güvenliği ile ilgili analiz ve görüşler aktarılmıştır.

2. VERİ GÜVENLİĞİ

2.1. Veri Güvenliğine Giriş

Veri, uygulamalar için en önemli kaynak durumunda olup daha çok veri bir araya geldiğinde veya toplandığında değeri artar. Bu yüzden de veri hırsızlığı, değiştirilmesi veya yok edilme tehlikesi ortaya çıkmaktadır. [37] Veri güvenliği için kısaca verinin bir varlık olarak hasarlardan korunması durumu denebilir. Daha geniş anlamda ele alınırsa, verinin bozulmadan saklanabilmesi, iletim hattı üzerinden güvenli bir şekilde değişmeden iletilmesi, yetkisiz kişilerce erişiminin engellenmesi, erişim yetkisi olanların kullanabilmesi, kimlik doğrulama gibi işlemlerin elektronik ortamda güvenle yapılabilmesidir.

2.2. Güvenlik Prensipleri

2.2.1. Gizlilik (Confidentiality)

Verinin yetkisiz kişilerin eline geçmesinin engellenmesi yani bilginin sadece erişim hakkı olan yetkili kişilerce erişilebilir olmasının temini anlamına gelir.

2.2.2. Veri bütünlüğü (Data Integrity)

Veriyi göndericiden çıktığı haliyle alıcısına ulaştırmaktır. Bu şekilde veri, iletim esnasında izlediği yollarda değiştirilmemiş, araya yeni veriler eklenmemiş, belli bir kısmı ya da tamamı tekrar edilmemiş ve sırası değiştirilmemiş olarak alıcısına ulaşır.

2.2.3. Erişilebilirlik (Availability)

Erişilebilirlik, mevcut yapıyı, kurum içinden ve dışından gelebilecek tehditlere karşı korumayı amaçlar.

2.2.4. Denetim (Accountability)

Bu hizmetin hedefi sistemde gerçekleşen olayları, daha sonra analiz edilmek üzere kayıt altına almaktır. Burada olay dendiğinde, bilgisayar sistemi ya da ağı üzerinde olan herhangi bir faaliyet anlaşılabilir. Bir sistemde gerçekleşebilecek olaylara, kullanıcının parolasını yazarak sisteme girmesi, bir web sayfasına bağlanmak, e-posta almak göndermek gibi örnekler verilebilir.

2.2.5. Kullanıcı doğrulama (Authentication)

Kullanıcı doğrulama, alıcının, göndericinin iddia ettiği kişi olduğundan emin olmasıdır. Yani geçerli kullanıcı ve işlemlerin tanınması ve doğrulanması ile bir kullanıcının hangi kaynaklara erişme hakkının olduğunun belirlenmesidir.

2.2.6. Güvenilirlik (Reliability - Consistency)

Bir bilgisayar veya sistemin beklenen davranışı ile elde edilen sonuçlar arasındaki tutarlılık durumudur. Başka bir deyişle güvenilirlik, sistemden ne yapmasını bekliyorsak, sistemin de buna sürekli ve kesin bir şekilde uyarak çalışması ve bunu çok güvenli bir şekilde yapabilmesidir.

2.2.7. İnkâr edememe (Non-repudiation)

İnkâr edememe, bir bilgiyi alan veya gönderen tarafların, o bilgiyi aldığını veya gönderdiğini inkâr edememesini sağlama işlemidir. Bu durumda gönderenin gönderdiği bir mesajı daha sonra inkâr edememesi, etse bile alıcının gönderenin mesajı gönderdiğini üçüncü kişilere ispat edebilmesi mümkün olur.

Yukarda değinilen güvenlik gereksinimlerini günlük hayatta karşılamak için aşağıdaki yöntemler kullanılmaktadır:

- Gizlilik sağlamak için mühürlü zarf,
- Bütünlük sağlamak için imza, barkod, damgalama,

- Kimlik doğrulaması için noter, kimlik kartı, trafik ehliyeti, kişinin şahsen başvuru yapması,
- İnkâr edememezlik için imza, alındı, onay,
- Haberleşmenin sürekliliğini sağlamak için farklı, birbirine alternatif iletişim yolları.

Elektronik ortamda veri iletişiminde tehditlere karşı yine elektronik tedbirler alınarak korunma sağlanabilir. Bu tehditlere karşı alınabilecek tedbirler ve kullanılabilecek yöntem ve araçlar şunlardır:

- Gizlilik sağlamak için veri şifreleme yöntemleri kullanılır.
- Bütünlük sağlamak için özetleme algoritmaları, mesaj özetleri, sayısal (elektronik) imzalar kullanılır.
- Kimlik doğrulaması için özetleme algoritmaları, mesaj özetleri, sayısal (elektronik) imzalar, sertifikalar kullanılır.
- İnkâr edememezlik için sayısal (elektronik) imzalar, işlem kayıtları kullanılır.
- Süreklilik için yedek sistemler, bakım, yedekleme, alternatif haberleşme kanalları kullanılır.

3. VERİ GÜVENLİK YÖNETİMİ MODELLERİ

3.1. Erişim Denetimi

Veri güvenliğinde dikkat edilmesi gereken bir konu da, kaynaklara kimin, nasıl, ne şekilde eriştiğini kontrol etmek, bu sayede veri üzerinde yetkisiz değiştirme ve verinin yetkisiz kişilerin eline geçmesi durumlarını engellemektir. Bunu sağlamak üzere kullanılan yönteme erişim denetimi (access control) denir. Bir kullanıcı veriye ulaşmak istediğinde kendisine kullanıcı adı ve parola sorulması, erişim denetimine örnek olarak gösterilebilir. Erişim denetimi, yazılım ve donanımsal olarak gerçekleştirilebilir. [20]

Erişim denetimi yöntemleriyle, veri kaynaklarına yetkisiz erişimler engellenir ve yetkili kullanıcıların da sahip olduğu yetki derecesine göre erişimleri sınırlandırılır.

Erişim denetimi üç aşamalı olarak gerçekleştirilebilir:

- Tanımlama (Identification)
- Kimlik doğrulama (Authentication)
- Yetkilendirme (Authorization)

3.1.1. Tanımlama (Identification)

Veri kaynağı için, yalnızca erişmesi gerekenlere erişim yetkisi verilmelidir. Bu erişim yetkisinin denetiminde, kullanıcılardan iki adet bilgi girmesi istenir. Birinci bilgi kullanıcıya ait sosyal güvenlik numarası, sicil numarası, kullanıcı adı, kullanıcı kimliği gibi kullanıcı adı (username) olarak kullanılacak, sistemde tek (uniqe) olarak tanımlanmış bir sözcüktür. İkinci bilgi ise, yalnızca kullanıcının bildiği şifredir.

3.1.2. Kullanıcı doğrulama (Authentication)

Kullanıcı doğrulama işlemi kullanıcı kimliğini kanıtlar. Bu bilgi ağ üzerinde kişisel/tek olmayı da doğrular. Yani kullanıcı doğrulama, eğer kullanıcının ağ veya kaynağa girişine izin verilmişse kullanıcının geçerli bir kullanıcı olup olmadığının kontrol edilmesi işlemidir. Kullanıcı doğrulama, temelde bir kullanıcı adı ve şifre girişiyle gerçekleştirilir. Ayrıca bir akıllı kartın kullanımı, tek seferlik bir parola, bir sayısal imza bilgisi, biyometrik bir özelliğin belirlenmesi yoluyla da olabilir. Giriş parolası, en yaygın kullanılan kullanıcı doğrulama biçimidir. Kullanım kolaylığı olmasına rağmen, başkalarının eline geçmesi kolay olduğundan güvenlik boşluğu oluşmasına da neden olabilecek bir yöntemdir.

Biyometrik tanıma, bir kişinin vücudunda bulunan ve yalnızca ona has özellikler taşıyan bir özelliğin sisteme tanıtılmasıdır. Bu özelliklere el ayası, el geometrisi, retina taraması, iris taraması, parmak izi, ses, yüz şekli, tuşlara basma hızı gibi örnekler verilebilir. Sisteme verilecek özel bilgi olarak sahip olunan bir akıllı kart ya da jeton (hardware token) da kullanılabilir. Son yıllardaki hızlı gelişimleri ile akıllı kartlar, hem fiziksel hem de mantıksal erişim denetim sistemlerinde yaygın kullanım alanı bulmaktadırlar. Bir akıllı kart, kullanıcısına özel bir takım veriler taşır. Tek bir akıllı kart, çoklu uygulama desteği sayesinde birden çok uygulamada kullanılabilir. Yukarıda belirtilen kullanıcı doğrulama yöntemlerinden birden fazlasının birlikte kullanılması, erişim denetim güvenliğini arttıracaktır.

3.1.3. Yetkilendirme (Authorization)

Yetkilendirme, kullanıcının kimliği kontrol edilip geçerli bulunduktan sonra sistem tarafından, kullanıcının hangi kaynakları kullanabilmesine yetkisi olduğunun belirlenmesi işlemidir. Yetkilendirme, doğrulaması yapılmış kimliğin hangi kaynaklara erişim hakkı olduğunu tanımlar. Yani kullanıcı, sisteme giriş yaptıktan sonra, ağ üzerinde bir kaynaktaki bir dosyaya erişmek istediğinde, öncelikle bu kullanıcının bu kaynağa erişim yetkisi olup olmadığı kontrol edilir. Eğer yetkisi varsa, kaynağa erişmesine izin verilir. Bu izinler ağ veya uygulama yöneticileri tarafından ağ üzerinde hassas kaynaklar ve izin verilen kaynaklar gibi paylaşıma

açılabilir fakat ilgili kaynaklara tam olarak özel olarak kullanma hakkına sahip kişiler erişim sağlayabilecektir. Yetkilendirme, kaynaklara erişim kontrolü işleminin etkili ve basit olarak yapılması anlamına gelmekte olup fakat bir ağ üzerinde dinlemeyi önleme konusunda etkili olmayan, kullanıcı adı ve şifrelerin güvenli bir kaynakta saklanmasını gerekli kılan bir yapıdır.

3.2. Şifreleme

Verinin korunması ile ilgilenen bilim dalı, kriptoloji olarak isimlendirilir. Kriptoloji, kriptografi ve kriptoanaliz olmak üzere iki kısma ayrılır. Kriptografi bilimi, mesajların içeriğini gizleyerek bir şifreleme sisteminin oluşturulmasıyla uğraşır. Bir kriptografik algoritma, matematiksel fonksiyonlar kullanarak şifreleme ve deşifreleme işlemlerini gerçekleştirir. Şifreleme ve deşifreleme işlemleri yapılırken ayrıca, bir anahtar kullanılmaktadır. Kriptoanaliz ise, şifreleme anahtarı bilinmeden şifreyi çözme yöntemleriyle uğraşmaktadır [35,37].

Gönderilmek istenen orijinal mesaj açık mesaj (plain text) ve bu mesajın şifrelenmiş hali şifreli mesaj (cipher text) olarak adlandırılır.

Günümüze kadar oluşturulan şifreleme algoritmaları her zaman bir önceki şifreleme algoritmasının dezavantajlarını ortadan kaldırmayı amaçlamıştır. Bu dezavantajlar algoritmanın hızlı çalışmaması veya teknolojinin gelişimiyle birlikte daha hızlı bilgisayarlar sayesinde güvenliğinin azalması yani daha kolay kırılması olarak gösterilebilir.

Tüm şifreleme kriptosistemlerinin şunları sağlayabilmesi gerekir:

- Şifreli metin ile açık metin hemen hemen aynı uzunlukta olmalıdır.
- Şifreli metin sadece anahtar yardımıyla okunabilmelidir.
- Şifreleme algoritmasının bilinmesi şifreleme güvenliğini etkilememelidir.
- Şifreli metin parçası ve ona karşılık gelen açık metin bilindiğinde, anahtarın bulunması için gerekli işlem sayısı, mümkün anahtar sayısına eşit olmalıdır. [53]
- Anahtarların tahmin edilmesi için gerekli minimum işlem sayısı günümüzdeki bilgisayarların yeteneklerinden büyük olmalıdır.

- Algoritmalar yazılımsal kullanıma sahip olduğu kadar donanımsal kullanıma da imkân sağlamalıdır.
- Şifreleme ve şifre çözme işleminin zorluğu ihtiyaç duyulan güvenlikle doğru orantılı olmalıdır. Çok önemli olmayan bir bilginin şifrelenmesi için bilginin kendisinden daha fazla işgücü ve zaman harcanması verimli olmayacaktır.
- Şifrelemede yapılan hatalar sonraki adımlara yansımamalı ve mesajın tamamını bozmamalıdır.

Günümüzde kullanılan kriptografik algoritmalar ikiye ayrılır. Bunlar, kullandıkları anahtar biçimine göre simetrik veya asimetrik olarak adlandırılırlar. Simetrik ve asimetrik şifrleme algoritmalarının karşılaştırması Tablo 3.1'de yer almaktadır.

Tablo 3.1: Simetrik ve asimetrik şifreleme algoritmalarının karşılaştırması

Konu	**Simetrik Kriptografi**	**Asimetrik Kriptografi**
Gizlilik	Sağlar	Sağlar
Bütünlük	--	Sağlar
Kullanıcı doğrulama	--	Sağlar
İnkâr Edememezlik	--	Sağlar
Performans	Hızlı	Yavaş
Güvenlik	Anahtar uzunluğuna bağlı	Anahtar uzunluğuna bağlı

3.2.1. Simetrik anahtarlı şifreleme sistemleri

Simetrik anahtarlı şifreleme sistemleri, haberleşmek isteyen taraflar arasında, önceden paylaşılan ve saklı tutulması gereken tek bir anahtarın, şifreleme ve deşifreleme işlemlerinde kullanılmasıyla, gizli iletişimin sağlandığı sistemlerdir [51]. Bu sistem algoritmalarının, çok hızlı şifreleme ve deşifreleme işlemleri yapabilmeleri ve donanımla kolay gerçeklenmeleri avantajlarıdır. Emniyetli anahtar dağıtımının, bütünlük ve kullanıcı doğrulama hizmetlerinin gerçeklenmesinin zor olması ise dezavantajları olarak gösterilebilir.

Simetrik algoritmalara örnek olarak AES, DES, 3DES, Blowfish, IDEA, RC4, SPN, FEAL ve SAFER algoritmaları verilebilir.

Blok şifreleme ve akış (stream) şifreleme olmak üzere iki temel simetrik şifreleme tipi vardır. Bunlardan blok şifreleme, orijinal metni veya şifreli metni bloklara bölerek şifreleme/deşifreleme işlemini yapar. Akış şifrelemede ise bir bit veya byte üzerinde şifreleme ve deşifreleme işlemleri yapılır.

3.2.1.1. Blok şifreleme

Blok şifreler, Shannon'un önerdiği karıştırma (confusion) ve yayılma (diffusion) tekniklerine dayanır. Karıştırma, şifreli metin ve açık metin arasındaki ilişkiyi gizlemeyi amaçlarken, yayılma açık metindeki izlerin şifreli metinde sezilmemesini sağlamak için kullanılır. Karıştırma ve yayılma, sırasıyla yer değiştirme ve lineer transformasyon işlemleri ile gerçeklenir. Feistel ağları ve yer değiştirme-permütasyon ağları olmak üzere iki ana blok şifreleme mimarisi vardır. Her ikisi de yer değiştirme ve lineer transformasyonu kullanır. [52]

Blok şifreleme algoritmalarına, SPN (Substitution-Permutation Network), DES (Data Encryption Standard), AES (Advanced Encryption Standard), FEAL örnek verilebilir. Blok şifreleme algoritmalarının gücü söz konusu olduğunda algoritmada kullanılan S kutuları, döngü sayısı, anahtarların XOR işlemine sokulması, blok uzunluğu, anahtarın uzunluğu ve özelliği büyük önem taşımaktadır.

Blok şifreleme algoritmalarında anahtarın uzunluğu ya da bit sayısı, en temel saldırı şekli olan geniş anahtar arama saldırısına karşın güçlü olmalıdır. Örneğin DES algoritması 56-bit anahtar kullanırken, AES algoritması DES'in bu zaafını örter niteliktedir ve 128, 192, 256 bit anahtar seçenekleri mevcuttur. Ayrıca anahtarın rastlantısal olması gerekmektedir. [55]

Blok şifreleme algoritmalarında döngü sayısı iyi seçilmek zorundadır. Çünkü lineer transformasyon ve yer değiştirmelerin bu seçilen değerle algoritmaya yeterli gücü vermesi gerekmektedir. Saldırıların başarısız olması için en önemli şartlardan biridir.

S kutuları, blok şifreleme algoritmasının en önemli elemanıdır. Çünkü algoritmadaki tek non-lineer yapıdır ve dolayısıyla algoritmaya gücünü vermektedir. [11]

3.2.1.2. AES (Advanced Encryption Standard)

AES (Rijndael) algoritması 128 bit veri bloklarını 128, 192, 256 bit anahtar seçenekleri ile şifreleyen bir algoritmadır. SPN [28], [24] algoritmasının geniş bir çeşididir. Döngü sayısı anahtar genişliğine göre değişmektedir. 128 bit anahtar için 10 döngüde şifreleme yaparken 192 ve 256 bit anahtarlar için sırasıyla 12 ve 14 döngüde şifreleme yapmaktadır. AES algoritmasında her döngü dört katmandan oluşur. İlk olarak 128 bit veri 4×4 byte matrisine dönüştürülür. Daha sonra her döngüde sırasıyla byte'ların yerdeğiştirmesi, satırların ötelenmesi, sütunların karıştırılması ve anahtar planlamadan gelen o döngü için belirlenen anahtar ile XOR'lama işlemleri yapılır. [7]

AES algoritması hakkında [11,14,55] kaynaklarından detaylı bilgi edinilebilir.

3.2.1.3. DES (Data Encryption Standard)

Açık anahtarlı bir kriptografik algoritma olan DES, 1970 yılında IBM tarafından geliştirilen Lucifer algoritmasının geliştirilmiş halidir. En çok bilinen algoritmalar arasında yer alan DES, 64 bit blok boyutu olan bir blok şifrelemedir. 64 bitlik düzyazı bloklarını 56 bitlik anahtarlar kullanarak 64 bitlik şifreli yazı bloklarına çevirmektedir. Düzyazı, bazı permutasyon ve yedeklemelerle işleme tabi tutulmaktadır. Daha sonra güvenli bir şifreli yazı bloğu oluşturmak için çıktılar orijinal düzyazı ile birleştirilmektedir. Söz konusu kriptolama serisi 16 kez tekrarlanmakta ve her seferinde farklı anahtar ve bit grupları kullanılmaktadır. [45]

DES'in en büyük dezavantajı anahtar uzunluğunun 56 bit olmasıdır. Bu algoritma günümüzde geliştirilen modern bilgisayarlar tarafından yapılan saldırılar karşısında yetersiz kalmaktadır. Daha güvenli şifreleme ihtiyacından dolayı DES, Triple-DES olarak geliştirilmiştir. DES algoritmasının güvenli olmadığı ve anahtar boyutunun artırılması gerektiği Bruce Schneier tarafından dile getirilmişti. [46]

DES algoritmasının detayları [4,21,55] kaynaklarından edinilebilir.

3.2.1.4. Triple-DES (3-DES)

Triple-DES, DES`in daha çok güvenlik sağlayan bir çeşididir. Bu metod kriptolama anahtarındaki bitleri 3 katına çıkaran, DES`in 3 kez kullanımına dayalıdır. Yani DES'in toplam anahtar uzunluğu 64 bit iken Triple-DES'in ki 192 bit yani 24 byte (24 karakter) tır. Triple-DES kullanımı DES kullanımına göre 2 kat daha fazla güvenlik sağladığına inanılmaktadır. [22]

Triple-DES üç kat daha fazla iş yaptığı için DES'e göre daha yavaştır. Fakat yavaş olduğu oranda da güvenlidir. Bu şifreleme algoritmasında dikkat edilmesi gereken tek husus üç tane anahtar bloğunun veya üçünden ikisinin aynı olmasından kaçınmaktır. Çünkü eğer anahtarlar aynı olursa işlem DES işleminden farksız ve bir o kadar da ağır yapılır.

3-DES algoritması hakkında [21,55] kaynaklarından geniş bilgi edinilebilir.

3.2.2. Asimetrik anahtarlı şifreleme sistemleri

Asimetrik şifreleme algoritmaları, şimetrik şifreleme algoritmalarının gizli anahtar problemini ortadan kaldırabilecek sistemler gerçekleştirebilirler. Bu şifreleme algoritmaları, şifreleme ve deşifreleme işlemleri için iki ayrı anahtar kullanmaktadır. Şifreleme işlemini gerçekleştirmek için açık anahtar ve deşifreleme işlemini gerçekleştirmek için gizli bir anahtar kullanmaktadır. Böylece gizli anahtarı paylaşmaya gerek kalmamaktadır. Asimetrik şifreleme algoritmaları çok kullanıcılı sistemler için çok büyük avantajlar sağlamaktadır. Asimetrik şifreleme algoritmalarının en büyük dezavantajı hızlı algoritmalar olmamalarıdır.

Asimetrik algoritmalara örnek olarak RSA, ECC, Diffie-Hellman ve El Gamal algoritmaları verilebilir.

3.2.2.1. RSA şifreleme algoritması

RSA şifreleme sistemi, yaygın olarak kullanılan hem şifreleme hem de sayısal imza atma imkânı tanıyan açık anahtarlı bir şifreleme yapısıdır. Sistem, 1977 yılında Ronald Rivest, Adi Shamir ve Leonard Adleman tarafından geliştirmiştir. RSA sisteminin güvenliği çarpanlarına ayırma probleminin zorluğu temeline dayanır. Çarpanlarına ayırma işleminin kolay bir yönteminin bulunması, RSA algoritmasının kırılması anlamına gelir. [29]

Basit matematiksel ilişkilerle çalışan bu yöntem de iki ayrı anahtar kullanılmaktadır. Anahtarlardan birisi herkese açık, diğeri ise gizlidir. Herkes açık anahtarını yayımlar ve kendisine şifreli bir mesaj göndermek isteyen kişiler bu anahtarı kullanarak mesajı şifreler ve ilgili kişiye gönderir. Şifrelenmiş mesajı sadece gizli anahtarın sahibi çözebilir.

RSA da anahtar oluşturma algoritması şu şekildedir:

- P ve Q gibi çok büyük iki asal sayı seçilir.
- Bu iki asal sayının çarpımı N = P x Q ve bir eksiklerinin çarpımı ?(N)=(P-1) x (Q-1) hesaplanır.
- 1'den büyük ?(N)'den küçük ?(N) ile aralarında asal bir E tamsayısı seçilir.
- Seçilen E tamsayısının mod ?(N)'de tersi alınır, sonuç D gibi bir tamsayıdır.
- E ve N tamsayıları genel anahtarı, D ve N tamsayıları ise özel anahtarı oluşturur.
- Şifreleme : $C = (m^E) \bmod N$
  Deşifreleme : $m = (C^D) \bmod N$

Genel ve özel anahtarları oluşturduktan sonra gönderilmek istenen bilgi genel anahtar ile şifrelenir. Şifreleme işlemi şu şekilde yapılmaktadır: Şifrelenecek bilginin (m) sayısal karşılığının E'ninci kuvveti alınır ve bunun mod N deki karşılığı şifrelenmiş metni (C) oluşturmaktadır. Genel anahtar ile şifrelenmiş bir metin ancak özel anahtar ile açılabilir. Bu yüzden şifrelenmiş metin (C), yine aynı yolla, şifrelenmiş metnin sayısal karşılığının D'ninci kuvveti alınır ve bunun mod N deki karşılığı orijinal metni (m) oluşturur.

RSA algoritmasında şifrelenecek olan mesaj metnini m kabul edersek bu mesaj binary olarak $2^k<N$ olacak şekilde k bitlik kısımlara ayrılır. m = m(1) + m(2) + m(3) + ……+ m(n). Daha sonra şifreleme için her bir kısıma $C(i) = m(i)^E$ mod N işlemi uygulanır. Böylece şifreleme yapılmış olur. Girişte kullanılan açık metin m, şifrelenmiş olarak C şeklinde elde edilmiş olur. Deşifreleme işleminde ise $m(i) = C(i)^D$ mod N şeklinde olur.

Örnek: RSA şifreleme algoritması kullanılarak ÖMER sözcüğünün şifrelenmesi:

Öncelikle genel ve özel anahtarlarımızı oluşturalım.

- P=47 ve Q=71 gibi iki asal sayı seçelim.
- Bu iki asal sayının çarpımı N = P x Q; N = 3337 ve bu iki asal sayının bir eksiklerinin çarpımı ?(N) = (P - 1) x (Q - 1); ?(N) = 3320 olarak hesaplanır.
- 1'den büyük 3320'dan küçük 3320 ile aralarında asal bir E = 79 tamsayısı seçelim.
- Seçilen E = 79 tamsayısının mod 3320'da tersi alınır, sonuç D = 1019 gibi bir tamsayıdır.
- 79 ve 3337 tamsayıları genel anahtarı, 1019 ve 3337 tamsayıları ise özel anahtarı oluşturur.

Şimdi oluşturduğumuz {79, 3337} ve {1019, 3337} anahtarlarımızı kullanarak şifreleme yapalım.

ÖMER sözcüğünü ikili bloklar halinde düzenlenirse:

ÖMER = 1916 0621 Harfleri rakama çevirmek için şöyle bir algoritma uygulanır.

A=01, B=02, C=03, …

Her blok aşağıdaki formüle göre şifrelenir:

$C1 = 1916^{79}$ mod 3337 = 2768 $C2 = 621^{79}$ mod 3337 = 1201

Şifrelenmiş hali C = 27681201 olur.

Şifrelenmiş verinin deşifrelenmesi:

$m(1) = 2768^{1019} \bmod 3337 = 1916$ $m(2) = 1201^{1019} \bmod 3337 = 0621$ olur.

Deşifrelenmiş hali m = 19160621 = ÖMER

Bu algoritmada iki asal sayının çarpımını kullanarak anahtar oluşturulmasının sebebi, iki asal sayının çarpımını asal çarpanlarına ayırmak asal olmayan sayıları ayırmaktan daha zorlu olmasıdır.

İki tamsayının aralarında asal olup olmadığının tespiti için matematikten de bildiğimiz Öklit algoritması kullanılır. RSA ile günümüzde 1024 bitlik bir anahtar basit uygulamalar için yeterli bir şifreleme tekniği olarak kullanılabilir.

Açık anahtarı kullanarak şifreleme işlemi yapılmasında gönderen kişinin doğruluğu ise sertifika yoluyla sağlanmaktadır.

3.2.3. Özetleme fonksiyonları

Bir özetleme fonksiyonu, herhangi bir uzunluktaki metni, giriş değeri olarak alır ve sonuç olarak sabit uzunluklu bir değer üretir. Bu değere mesaj özeti (message digest) adı verilir. Burada üretilen özet, fonksiyona giren metnin karakterini taşımaktadır denilebilir. Giriş metninde yapılacak tek bir karakter değişikliği bile üretilecek özette büyük değişikliklere yol açar. Ayrıca, özetleme fonksiyonu tek yönlü olduğundan, özetten asıl metne geri dönüş yoktur. Özetleme fonksiyonları, uzun metinlerin, asimetrik bir algoritma ile şifrelenmeleri sırasında, asimetrik algoritmanın başarım dezavantajını ortadan kaldırmak amacıyla kullanılırlar. Tüm mesaj metni değil de yalnızca mesajın özeti alınarak asimetrik algoritmayla şifrelenir. Özetleme algoritmalarına örnek olarak SHA–1, DSS, MD2, MD4, MD5 verilebilir.

3.2.4. Sayısal imza ve açık anahtar altyapısı (AAA)

Bir sayısal imza, şifrelenmiş bir özet (hash) değeridir. Sayısal imzalar yardımıyla, alıcı taraf göndericinin kimliğinin sınamasını yapar ve göndericinin kim olduğundan tam olarak emin olur. Bunun yanında, sayısal imza teknolojisi, gönderilen verilerin

bütünlük sınamasında da kullanılabilir. Buna göre sayısal imza teknolojisi, ilk bölümde bahsedilen kullanıcı doğrulama ve veri bütünlüğü prensiplerinin gerçekleştirilmesinde kullanılırlar.

Sayısal imzalar, gerçek hayatta kullanılan ve elle atılan imzanın (ıslak imzanın) bilişim dünyasındaki karşılığı olarak görülebilir. Bir sayısal imza, imzaladığı içeriğin, imzalandığı andan itibaren değişmediğinin kanıtlanmasında kullanılabilir. Sayısal imzalama, asimetrik kripto algoritmaları yardımı ile yapılır. [44] Sayısal imzalama, mesaj bir mektup zarfına konulduğunda üzerinin mühürlenmesi gibi düşünülebilir. Sayısal imzalar, bilgisayar ağları yoluyla yapılan finansal işlemlerin güvenli bir şekilde yapılması ve veritabanı bütünlüğünün kontrolü gibi kullanım alanları bulmuşlardır.

Sayısal imzalar, açık anahtar altyapısı (Public Key Infrastructure–PKI) teknolojisinin de belkemiğini oluşturur. PKI, çok geniş bir coğrafi alana yayılmış kullanıcılar arasında, güvenli bir haberleşme altyapısı kurmayı hedefleyen bir teknolojidir. Açık anahtar altyapısını oluşturan elemanlardan başlıcaları şunlardır:

3.2.4.1. Sayısal sertifikalar

Sertifika, bir sertifika otoritesinin gizli anahtarı ile imzalanmış kullanıcı kimliği ve kullanıcı açık anahtarından oluşur. Bir sayısal sertifika, gerçek hayatta kullanılan bir kimlik kartının, bilişim güvenliğindeki karşılığıdır. Bir sertifikanın içinde; sahibinin kimlik bilgileri, yetki derecesi, sertifikanın son kullanma tarihi, sahibinin kripto anahtarı bilgisi yer alır. Bir sayısal sertifika, bir kullanıcının bir sisteme girerken kimlik sınamasının yapılmasında ya da kriptolu e-posta mesajlarının gönderilmesinde kullanılabilir. Sayısal sertifikalar için uluslararası ITU (International Telecommunications Union) kurumu tarafından X.509 standardı yayımlanmıştır ve bu standart yaygın olarak kullanılmaktadır. [38]

3.2.4.1.1. X.509 Sertifikalar

AAA sistemlerinde yaygın olarak kullanılan sertifika standartlarının başında X.509 standardı gelir. "Internet X.509 Public Key Infrastructure Certificate and CRL Profile" adını taşıyan bu standart dâhilinde X.509 Sertifikası V1, V2, V3 sürümleri ve X.509 Sertifika İptal Listesi (SİL) V1, V2 sürümleri tanımlanmıştır.

X.509 Sürüm 1 dokümanı aşağıdaki temel alanları tanımlar:

- Sertifikayı yayınlayanın tekil kayıt adı, ör: ou=UEKAE, o=TUBITAK,
- Sertifika sahibinin tekil kayıt adı, ör: cn=Muhsin Ak, ou=UEKAE, o=TUBITAK,
- Sertifikalandırılan açık anahtarın kendisi,
- Geçerlilik süresi (Başlangıç-Bitiş) (Yerel Sistem saatine ya da GMT'ye göre),
- Açık anahtar algoritması (RSA, DSA, Elliptic Curve),
- Sertifika makamının imzalama algoritması (RSAMD5, RSASHA-1, DSASHA-1, ECDSA),
- Sertifika makamının imzası.

X.509 sürüm 2, sürüm 1'e ek olarak tekil kayıt adındaki "cn" alanının tekilliğini sağlamak için başka bir tekil id kullanır. Sürüm 2 artık kullanılmamaktadır.

X.509 sürüm 3, sürüm 1'e ek olarak eklentiler içerir. Bunlar, standart eklentiler ve özel eklentiler olarak iki tipte olabilirler. X.509 sürüm 3 sertifika yapısı tüm beklentilere yanıt verebilecek esneklikte ve kapasitededir.

X.509 sertifikasının bütünlüğünün korunması için sertifika içinde yer alan tüm bilgiler sertifikayı veren makam tarafından elektronik olarak imzalanır ve oluşan elektronik imza bu sertifikanın arkasına eklenir. Böylece sertifika alanlarından herhangi birisi üzerinde sonradan değişiklik yapılıp yapılmadığı kontrol edilebilir. Bir elektronik sertifikanın üstündeki elektronik imza doğrulanarak sertifikanın geçerliliği kontrol edilir.

X.509 sertifikaları byte dizilerinden oluşan elektronik dokümanlardır. Bir elektronik sertifikanın uzunluğu, kullanılan anahtar uzunluğu ve eklentilere bağlı olarak değişmekle beraber genellikle birkaç kilobyte'ı geçmez.

3.2.4.2. Sertifikasyon otoriteleri

Bir AAA sistemine dâhil olan kullanıcılar için sayısal sertifika üretim ve saklama merkezleridir. Bir kullanıcıya mesaj gönderilirken ya da gelen bir mesajdaki sayısal imzanın doğruluğunun sınanırken, o alıcının açık anahtarına ihtiyaç duyulur. Bu açık anahtarı elde etmek için, sertifikasyon otoritesinden, kullanıcının kimliği yardımı ile kullanıcının sayısal sertifikası elde edilir. Kullanıcının açık anahtarı, bu sertifika içerisinden alınarak kullanılır.

3.3. Uygulama Güvenliği

Belli bir amaç için geliştirilen bir uygulamanın, amacının dışında kullanılmaması, kötü amaçlı istismar edilmemesi, dolayısıyla sistem güvenliğini tehlikeye atmaması için alınması gereken tedbirler ve uygulanması gereken kurallar bütünü uygulama güvenliği olarak tanımlanmaktadır. [30]

Uygulama güvenliğinin önemli bir sorun haline gelmesinde, hızla büyüyen internet'in rolü çok fazladır. İnternet, kötü niyetli saldırganlar için bulunmaz bir ortam yaratmış, yazılım açıklarını kısa zamanda keşfederek, uzak veya yakın noktalardan saldırılar düzenlemelerine olanak vermiştir. Ayrıca yazılım ürünlerindeki çeşitlilik ve sayı artışı, bunların güncelleme ve genişlemelerinin çoğalması, yazılımların karmaşıklığının gün geçtikçe artması, yazılımların yeterince test edilmeden kullanılmaya başlanması uygulama güvenliğini tehlikeye atan diğer unsurlardır.

Günümüzde, bir şirketin web uygulamasını hedefleyen saldırıların %70'i ağ ve sistem katmanına değil, uygulama katmanında gerçekleşmektedir.[Gartner Group]

Web uygulamalarında en çok rastlanan 10 güvenlik problemi şunlardır: [2]

- Kontrol edilmeyen parametreler,
- Erişim kontrol eksiklikleri,
- Kullanıcı ve oturum yönetiminde eksiklikler,
- Cross-site scripting (XSS) güvenlik açıkları,
- Ara bellek (buffer) taşmaları,
- Komut ekleme açıkları (özellikle SQL İnjection),
- Hata mesajları,
- Güvensiz şifreleme kullanımı,
- Uzaktan yönetim açıkları,
- Web ve uygulama sunucusu ayar yanlışlıkları,

3.4. İşletim Sistemi Güvenliği

İşletim sistemi güvenliğinin arttırılması için gerekli yamalar ve önerilen güvenlik yapılandırmaları takip edilmeli ve yapılmalı, bilgisayarlardaki gereksiz bileşenler kaldırılmalı, antivirus yazılımları kurulmalı ve sürekli güncellenmeli, çeşitli erişim kısıtlamaları getirilmeli, uzaktan yönetim protokolü veriyi ağ üzerinde şifrelenmiş olarak iletebilmelidir.

3.5. Ağ ve İnternet Güvenliği

İnternet ve ağ güvenliğini sağlamak için kullanılan bazı yazılım ve araçlar vardır.

- Güvenlik Duvarı (Firewall): Ağ ile ağ dışında kalan bilgisayarların veri iletişimini, trafiğini kontrol eden ve belirlenen bir politika çerçevesinde izin veren bir ağ elemanıdır.
- Antivirüs yazılımları: Ağdaki kullanıcı bilgisayarlarına ve sunucu bilgisayarlara trojan, virüs, solucan, truva atı gibi zararlı yazılımların bulaşmasını engelleyerek ağı korur.
- Saldırı tespit sistemi (Intrusion Detection System): Sisteme ve ağ kaynaklara olan saldırıları tespit etmek, sistemi izleyip normal olmayan durumları saptamak ve

bunlara karşı gerekli önlemleri almayı hedefleyen güvenlik sistemleridir. [49]

- Honeypot alarm: Bilişim sistemlerine karşı gerçekleştirilen saldırıların tespit edilmesi için kurulmuş tuzaklardır. [39]
- Zayıflık tarama yazılımları: Ağdaki bilgisayar ve yazılımların açıklarının tespit edilmesi amacıyla kullanılır.
- Ağ dinleme ve yönetim yazılımları: Yerel ağdaki bütün trafiğin dinlenerek beklenmeyen durumların ortaya çıkarılması ve bilgisayarların yönetilmesi işlemlerini yürütür. [13]
- Loglama yazılımları: Sistemde yapılan tüm işlemlerin kaydedilmesi ve bunların belirli aralıklarla takip edilmesi sayesinde saldırıların önceden tespiti için kullanılabilmektedir.
- Yedekleme araçları: Verilerin zarar görmesi durumunda söz konusu verileri en kısa sürede, eksiksiz olarak tekrar aynı şekilde elde edebilmek için belirli aralıklarla yedeklerini farklı bir ortama yedekleme işlemidir.

3.5.1. Güvenlik duvarları (Firewalls)

Güvenlik duvarı, bilgisayar ağına, internetten veya diğer ağlardan erişimi kısıtlayıp, gelecek saldırılara karşı koruyan donanım ve yazılıma denir. Güvenlik duvarı, internet ile yerel ağ arasında bulunur. Bu sayede internetten gelen ve internete giden paketleri kontrol edebilir. 3 çeşit güvenlik duvarı vardır:

- Paket filtrelemeli güvenlik duvarı (Packet Filtering Firewall),
- Uygulama seviyesi güvenlik duvarı (Application Level Firewall),
- Durum denetlemeli güvenlik duvarı (Stateful Inspection),

Paket filtrelemeli güvenlik duvarı, kaynak ve hedef IP adreslerine bakarak belirlenen kuralları taşıyan bağlantılara izin verir, diğerlerini engeller. Paket filtreleme, içeriği kontrol etmez. Paketin IP veya port bilgileri değiştirilebileceğinden yeterince güvenilir değildir.

Uygulama seviyesi güvenlik duvarı, proxy sunucu gibi çalışan bir güvenlik duvarı türüdür. Proxy sunucu bağlantıları ayarlayarak ve sınırlayarak trafiği kontrol eder.

Intranet bağlantılarını maskeleyerek ağı korur. Bir proxy sunucu ağ ortamında hangi servislere izin verileceğinin kontrolünü sağlar. [48] Performans sınırlamaları olduğundan sınırlı kullanım sağlarlar. Fakat çok etkilidirler. Her bir yeni işlem başlamasında o kullanıcı için bir oturum başlar. Uygulama seviyesi güvenlik duvarı, güvenlik duvarı tipleri içinde en güvenilir olanıdır. [16]

Durum denetlemeli güvenlik duvarı, ağ katmanında çalışan dinamik paket filtreleme olarak da adlandırılan güvenlik duvarı tipidir. Bu güvenlik duvarı, aktif TCP oturumlarını inceler. Kaynak IP adresi, hedef IP adresi ve port numaraları gibi oturum giriş kayıtlarının her birine bakar. Durum denetleme, paket filtrelemeli yöntemden daha güvenilirdir. Çünkü yalnızca izin verilen oturuma ait paketlere müsaade edilmektedir. [43]

Ayrıca, hibrid güvenlik duvarı denilen, paket filtreleme, durumsal denetim ve uygulama seviyesi yöntemlerinin avantajlarını birleştiren bir yapı da kullanılabilmektedir.

3.5.2. Saldırı tespit sistemleri

Güvenlik hedeflerini tehlikeye atmaya çalışan etkilere saldırı adı verilir. Saldırı engelleme için doğrulama, şifreleme ve benzeri teknikler yeterli değildir. O yüzden saldırı tespiti yapmaya ihtiyaç vardır. [58]

Saldırı tespit sistemleri, internet veya yerel ağlardan gelebilecek ve sistemlere zarar verebilecek saldırıları fark etmek üzere tasarlanmış sistemlerdir. Temel amaçları belirlenen kurallar çerçevesinde bu saldırıları tespit ederek e-posta, sms, snmp mesajları gibi araçlarla haber vermek ve gerekiyorsa bu saldırıyı önlemektir. [15]

Bilgisayar üzerinde çalışan kötü niyetli aktivitelerin tespitine yarayan sistemlere sunucu tabanlı saldırı tespit sistemleri (Host-based ID systems) denir. Benzer şekilde bilgisayar ağı üzerindeki verilerin akışlarında çalıştırılan sistemlere ağ tabanlı saldırı tespit sistemleri (Network-based ID systems) adı verilir. [17]

Saldırı tespiti alanındaki çalışmalar iki temel kategoriye ayrılabilir; kötüye kullanım tespiti (misuse detection) ve anormallik tespiti (anomaly detection).

Kötüye kullanım tespiti, saldırıları meydana çıkarmak için özel tanımlı örüntülerden faydalanır. Bu örüntüler, "imzalar" (signatures) olarak adlandırılırlar. Sistemin kaynak girdisi (ağ trafiği, sistemlere ilişkin günlük kayıtları vb.) üzerinde saldırı biçimleri kullanılarak yapılan analizler ile bilinen zaaflara yönelik saldırıların ya da ihlallerin gerçekleşip gerçekleşmediği tespit edilmeye çalışılır; tespit edilmesi durumunda alarm üretilir. Kötüye kullanım tespitinin avantajı, imza kriterleri iyi bir şekilde tanımlanabildiği takdirde yanlış uyarıların düşük olma olasılığıdır. Dezavantajı ise, yeni saldırıların kaçırılabileceğidir, çünkü bunlar tanımlanmamıştır.

Anormallik tespitinde ise, ilk adım, normal kullanıcı davranış profillerinin (veri tabanı) oluşturulmasıdır. Kaynakların (kullanıcılar, programlar, sistemler vb.) normal durum davranışı istatistiksel yöntemler ile tespit edilir ve profil olarak adlandırılan özet bilgiler oluşturulur. Kaynakların davranışındaki tüm değişimler düzenli olarak değerlendirilerek ilgili profilin de güncellenmesi sağlanır. Kaynağın davranışında istatistiksel olarak ani bir değişim görüldüğünde bir “anormallik” olduğu öne sürülerek alarm üretilir.

Saldırı tespit sistemlerine ilişkin bir diğer sınıflama, sistemin girdisine göre yapılan sınıflamadır.

3.5.2.1. Ağ tabanlı saldırı tespit sistemleri

Ağ temelli sistemlerde, ağ üzerinden akan trafik, sistem tarafından yorumlanmaktadır. Ağ temelli sistemler, daha az çabayla daha yüksek sayıda kaynağın davranışının izlenebilmesine ve politika ihlallerinin tespit edilebilmesine imkân verdiği için tercih edilmektedir. Ağ temelli sistemlerin şifreleme teknolojisinden faydalanan uygulamalar için, trafiği inceleyememesi önemli bir zaaftır. [3] Ayrıca ağ tabanlı sistemlerin genellikle, korumayı arttırmak amacıyla, belirlenen bir IP aralığını düzenli tarayarak, port bazında anormal aktiviteyi bulma ve oturumlar üzerinde işlem yapma özelliği bulunmaktadır.

3.5.2.2. Sunucu tabanlı saldırı tespit sistemleri

Sunucu tabanlı saldırı tespit sistemlerinde, yüklenen bir yazılım, sistemin log dosyalarını kullanarak saldırıları tespit eder. Sunucu tabanlı saldırı tespit sistemleri, hem sunucuya giren ve çıkan trafiği izler, hem de sistem dosyalarının bütünlüğünü ve şüpheli aktiviteyi kontrol eder. Sistemin girdisi, sistem günlükleri ve uygulama kayıtları gibi doğrudan sistem üzerinden temin edilebilen bilgilerin tamamıdır.

Sunucu tabanlı saldırı tespit sistemleri genellikle ajan-yönetici yapısında çalışan sistemlerdir. Sunucular üzerindeki network paketlerini, bağlanma girişimlerini, login girişimlerini, kritik sistem dosyalarına erişimleri, bu dosyalardaki değişimleri, kullanıcı haklarındaki değişimleri vb. kontrol ederler.

3.5.3. Sanal özel ağ (VPN)

VPN (Virtual Private Network/Sanal Özel Ağ) internet üzerinden şifreli ve güvenli veri iletişimi sağlamak için kullanılan bir teknolojidir. [41] Kiralık hatlar (Lease-line) gibi daha güvenli, sağlam çözümlerin yerine VPN kullanılmasının temel nedeni, maliyet ve kolay yapılandırmadır.

Temelde iki tip VPN teknolojisi vardır. Amaca göre bu iki VPN teknolojisinden biri seçilebilir. Bu teknolojiler “Remote Access VPN ” ve “Site-to-site VPN” dir.

Remote Access olarak tanımlanan VPN türü, firmaların gezgin çalışanlarının firma ağına her yerden güvenli iletişimlerini sağlamak için kullanılır. Ya da büyük bir firmanın farklı lokasyonlardaki şubelerini merkeze bağlamak için kullanılır.

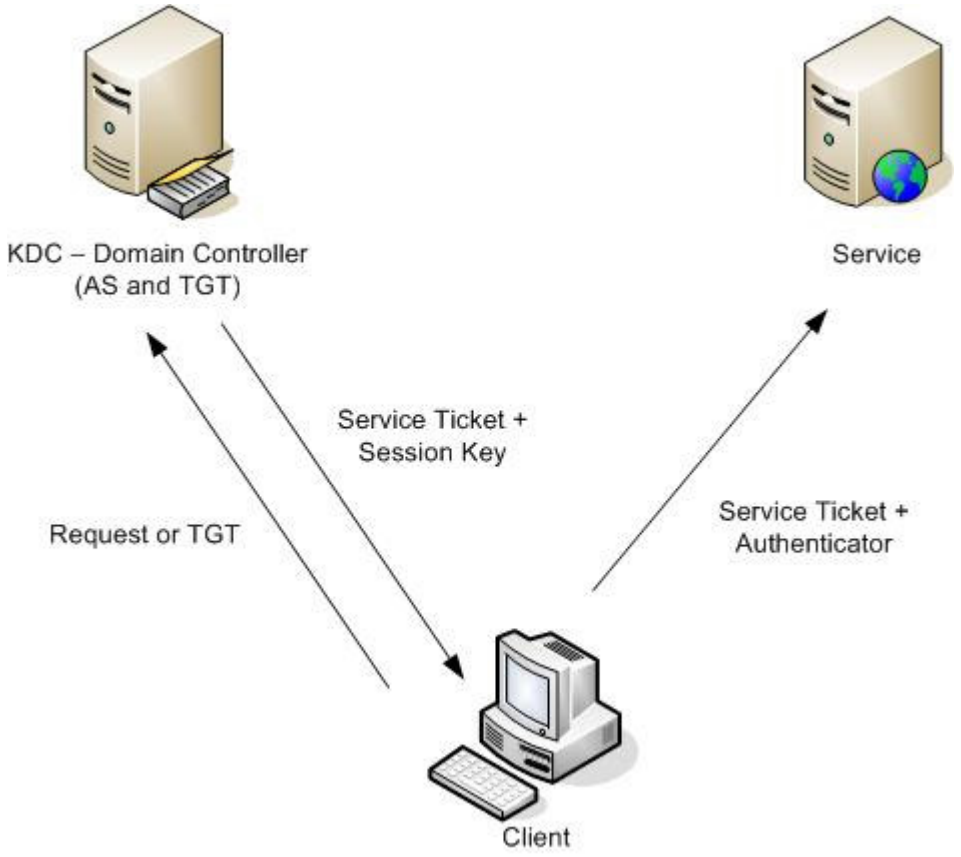
Site-to-site VPN ise, genellikle farklı firmaların birbirleri ile VPN üzerinden güvenli iletişim kurmaları için düşünülmüştür. Aynı zamanda firmanın farklı lokasyonlardaki şubelerinin merkeze bağlanmasını da sağlar. Remote Acces VPN’den temel farkı VPN işlemini gören iki uçta VPN sunucu olmasıdır.

3.5.4. Kerberos

Kerberos, simetrik anahtarlı şifreleme kullanan bir kimlik doğrulama protokolüdür. Athena projesinin bir parçası olarak MIT (Massachusetts Institute of Technology) tarafından geliştirilmiştir. Kerberos Needham-Schroeder protokolünü temel alır. [25]

Kerberos açık bir ağda güvenli kimlik doğrulama sağlamak için şifreleme teknolojisini ve hakem olarak KDC (Key Distribution Center, Anahtar Dağıtım Merkezi) olarak isimlendirilen güvenilen üçüncü bir taraf kullanır.

Kerberos'un kullanımda olan iki sürümü vardır; sürüm 4 ve sürüm 5. Sürüm 1'den sürüm 3'e kadar olan sürümler iç geliştirme sürümleri olup hiçbir zaman yayımlanmamıştır. Sürüm 4'ün ise birçok zayıf yönü bulunduğundan kullanılması uygun değildir. Burada Sürüm 5 incelenecektir. Windows 2000, XP, Vista, Windows Server 2003 ve Windows Server 2008 varsayılan kimlik doğrulama protokolü olarak Kerberos kullanır. Ayrıca, UNIX/Linux sistemlerde de Kerberos kullanılmaktadır.



Şekil 3.1: Kerberos doğrulamanın çalışma şekli.

Anahtar dağıtım merkezi (KDC) iki kısımdan oluşur: Doğrulama sunucusu (Authentication Server, AS) ve Bilet sunucusu (Ticket Granting Server, TGS). Kerberos kullanıcıların kimliklerini doğrulamak için bilet (ticket) kullanır. KDC ağdaki her bir istemci ya da sunucu için gizli anahtarları tutan bir veritabanına sahiptir. Bu gizli anahtarlar sadece KDC ve ait olduğu istemci tarafından bilinir.

Kerberos'un çalışma şekli ise şöyledir:

1- Kullanıcı istemci üzerinde kullanıcı ismini ve şifresini girer.
2- İstemci kullanıcı şifresi üzerinde tek yönlü bir hash algoritması uygular ve bu istemcinin gizli anahtarı olur.
3- İstemci doğrulama sunucusuna (AS) oturum açma bilgilerini gönderir. Oturum açma bilgileri, kullanıcı adı ve domain bilgilerini içerir. Dikkat edilirse burada kullanıcının şifresi ya da gizli anahtarı doğrulama sunucusuna (AS) gönderilmez.
4- Doğrulama sunucusu (AS) istemcinin kendi veritabanında bulunup bulunmadığını kontrol eder, eğer mevcutsa, istemciye iki adet mesaj gönderir.
Mesaj A: İstemci/TGS oturum anahtarı (session key), bu kullanıcının gizli anahtarı ile şifrelenir (Bu gizli anahtarı sadece KDC ve istemci biliyor. Bu yüzden bu oturum anahtarını sadece istemci açabilir).
Mesaj B: Ticket-Granting Ticket. Bu bilet kullanıcı ismi ve biletin kullanım süresi gibi bilgileri içerir ve TGS'nin kendi gizli anahtarı ile şifrelenir.
5- İstemci bu iki mesajı aldıktan sonra İstemci/TGS oturum anahtarını almak için Mesaj A'yı açar. Mesaj A kullanıcının kendi şifresinden üretilen gizli anahtarı ile şifrelendiğinden kullanıcı bu mesajı açabilir. Bu gizli anahtar TGS ile daha sonra yapılacak olan iletişimde kullanılacaktır. Burada istemci Mesaj B'yi açamaz. Çünkü Mesaj B sadece KDC tarafından bilinen KDC'nin gizli anahtarı ile şifrelenmiştir.

Bu işlemin sonucunda istemci kimlik doğrulamasını başarıyla yapmıştır. Kerberos protokolünü kullanarak domaindeki diğer kaynaklara erişmeye hazırdır. Buraya kadarki adımlar kullanıcının oturum açmasına kadar olan kısımdı. Şimdide istemcinin bir sunucuya erişimi sırasındaki adımları inceleyelim;

1- İstemci aynı etki alanındaki bir sunucuya erişmek istediğinde TGS'ye şu iki mesajı gönderir;

Mesaj C: Mesaj B'de aldığı TGT ve erişilmek istenilen sunucu ID'si.

Mesaj D: İstemci ID'si ve zaman bilgisi (TimeStamp) istemci/TGS oturum anahtarı ile şifrelenir.

2- TGS bu iki mesajı aldıktan sonra, Mesaj D'yi istemci/TGS oturum anahtarı ile açar ve sonrasında istemciye şu iki mesajı gönderir;

Mesaj E: İstemci-Sunucu bileti. Bu bilet İstemci ID, istemci ağ adresi, geçerlilik süresi ve İstemci/Sunucu oturum (session) anahtarı bilgilerini içerir. Bu mesaj sunucunun gizli anahtarı ile şifrelenir.

Mesaj F: İstemci/Sunucu oturum (session) anahtarı, İstemci/TGS oturum anahtarı ile şifrelenmiştir.

3- Mesaj E ve F'yi istemci aldıktan sonra, istemci artık sunucuya bağlanmak için gerekli bilgilere sahiptir. Ve istemci sunucuya bağlanıp şu iki mesajı gönderir;
Mesaj E: Bir önceki adımdaki Mesaj E' yi gönderir.

Mesaj G: İstemci ID'si ve zaman bilgisi (Timestamp) bilgisini gönderir. Bu mesajıda İstemci/Sunucu oturum anahtarı ile şifreler.

4- Sunucu kendi gizli anahtarı ile Mesaj E'yi açar. Ve bu mesajın içerisinden İstemci/Sunucu oturum anahtarını alır (Burada dikkat edilmesi gereken husus Mesaj E'yi sadece bağlanılan sunucunun açabilmesidir. Sunucu bu mesajı açtığı zaman bunun güvenilen bir kaynaktan geldiğini bilir. Çünkü bu mesaj sunucunun gizli anahtarı ile şifrelenmiştir ve bu anahtar sadece kendisinde ve KDC' de mevcuttur). Bu anahtarla da Mesaj G' yi açar. Daha sonra istemciye kimlik doğrulamasını onaylamak için şu mesajı gönderir;

Mesaj H: Mesaj G'deki zaman bilgisi (Timestamp) değerini bir artırır. Bu mesajı istemci/sunucu oturum anahtarı ile şifreler.

5- İstemci mesajı aldıktan sonra zaman bilgisinin (Timestamp) doğru olarak güncellendiğini kontrol eder. Eğer bu işlem doğru olarak yapıldıysa istemci artık sunucuya güvenebilir ve sunucuya istenen servisler için bağlantı kurar. Burada dikkat edilirse şifreler hiç bir zaman ağ üzerinde dağıtılmıyor. Bağlanılan sunucuda bağlantı isteğini sadece kendi gizli anahtarını kullanarak onaylıyor. KDC' ye bağlanmaya hiç bir şekilde ihtiyaç duymamaktadır.

3.5.5. SSL (Secure Sockets Layer)

SSL protokolü iki alt protokolü içermektedir: SSL Record Protokolü ve SSL Handshake Protokolü.

SSL handshake protokolü, istemci ile sunucu arasında bir SSL bağlantısı kurulmak istendiğinde gerçekleşir. Oturumun şifreleme parametreleri SSL record protokolünün üzerinde çalışan, SSL handshake protokolünde belirlenir.

SSL handshake protokolü, SSL aktif hale getirilmiş bir istemci ile sunucunun ilk bağlantı kurduklarında bir seri mesajın değiş tokuş yapılması için SSL record protokolünün kullanılmasını kapsar. Bu seri mesaj değiş tokuşu aşağıdaki işleri kolaylaştırmak için tasarlanmıştır: [20]

- Sunucuyu istemciye doğrulamak,
- Sunucu ve istemciye şifreleme algoritmasını kullanmaya ve şifre seçmeye izin vermek (her ikisi de destekliyorsa),
- İsteğe bağlı olarak istemcinin sunucuya doğrulamasını sağlamak,
- Paylaştırılmış gizli şeyleri üretmede genel anahtar şifreleme tekniklerini kullanmak,
- Şifrelenmiş bir SSL bağlantısı kurmak.

SSL Record Protokolü kendinden daha üst seviyedeki protokollerden gelen taşınacak bilgiyi 2^{14} byte'ı geçmeyecek record yapılarına dönüştürür. Her mesajın başında 2 veya 3 byte'lık bir header bilgisi bulunur. Bütün recordlar mevcut oturumda belirlenmiş bir sıkıştırma algoritması ile sıkıştırılırlar. Bu sıkıştırma algoritması ile düz text yapıdaki recordlar sıkıştırılmış bir yapıya dönüştürülürler. Ayrıca bu sıkıştırılmış yapıdaki veri mevcut CipherSpec tarafından belirlenmiş bir MAC (Message Authentication Code) ve şifreleme ile de korunur.

3.6. Fiziki Güvenlik

Fiziksel güvenlik saldırganların bir veri kaynağına veya veriye ulaşmalarını engellemek veya caydırmak amaçlı alınan önlemlerdir. Ayrıca veri kaynağını yangın,

güç kaybı, su ve ısı gibi doğal afet ve çevresel tehlikelere karşı korunmasıdır. [44] Disket, taşınabilir disk, CD-ROM, doküman gibi bilgi saklayan ortamlar kullanılmadıkları zamanlar açıkta ve yetkisiz insanların erişebileceği bir yerde bırakılmamalı, kaydedilen veriler gizlilik derecelerine uygun olarak saklanmalıdırlar.

Veri kaynağının bulunduğu donanıma olan fiziksel erişimi kısıtlanmak ve kontrol altında tutmak için daha gelişmiş yöntemler de kullanılabilir (örn: akıllı kart, gelişmiş kimlik doğrulama sistemleri vb).

3.7. Yedekleme ve Kurtarma

Düzenli ve sistematik bir şekilde çalışılan verileri yedeklemek, meydana gelebilecek bir saldırı ya da doğal bir afetin vereceği zararı minimuma indirebilir. Bir sistemin yedeğinin alınmasınının maliyeti, değerli veriyi kaybettikten ya da veritabanındaki tüm kayıtlar silindikten sonra ortaya çıkacak zarardan çok daha düşüktür.

Yedekleme işleminde sistemdeki tüm verinin yedeği alınmamalıdır. Bu tür bir yedekleme, yedekleme maliyetini oldukça arttıracaktır. Yedeklenecek verilerin ve yedekleme zamanlamasının iyi yapılmasıyla, sistemin performansını ve işleyişini en az şekilde etkileyecek şekilde bir yedekleme gerçekleştirilmelidir.

Yedeklerin veri kaynağından uzak bir yerde saklanması önemlidir. Yedekleme işlemi ağ üzerinden farklı bir bilgisayara alınabileceği gibi yüksek kapasiteli teyp, flash bellek, taşınabilir harddisk, CDROM, DVD-Rom gibi ortamlara da yapılabilir.

Log alma yöntemiyle de veritabanındaki, verilerin değiştirilme zamanı bilgisi, değişikliğin ne olduğu bilgisi vb. bilgiler loglarda yer alabilir.[12]

4. VERİTABANLARINDA GÜVENLİK

4.1. Microsoft SQL Server Güvenliği

4.1.1. Kullanıcı doğrulama (Authentication)

SQL Server kullanıcı doğrulaması, Windows doğrulama ve Karışık (Mixed Mode) doğrulama olmak üzere 2 şekilde yapılabilmektedir.

Windows kimlik doğrulama, NTLM (NT LAN Manager) veya kerberos protokolleri ile single sign on (SSO) sağlayarak kullanıcının veritabanına erişmesini sağlar. NTLM, single sign on gibi mekanizmalar gibi Microsoft sistemlerinin bütünleştirilmesinde kullanılan bir doğrulama protokolüdür. Windows kimlik doğrulama ise, SQL Server'ın varsayılan ve Microsoft tarafından tavsiye edilen kimlik doğrulama metodudur. Windows şifre yönetimi, lokal ya da Active Directory ile yapılır ve GPO (Group Policy Object) ile güvenliği sağlanabilir.[34]

Karışık doğrulama, SQL Server veya yine Windows işletim sistemi yoluyla yapılabilmektedir. Bu yöntem daha çok SQL Server doğrulama olarak bilinmekte ve kullanılmaktadır. Karışık doğrulamada, kullanıcı adı ve şifreler direkt olarak SQL Server tarafından kontrol edilir ve SQL Server tarafından yönetilen şifre politikaları ile de sınırlandırılırlar.

4.1.1.1. Şifre politikası (Password Policy) sistemi

SQL Server da yer alan önemli bir güvenlik özelliği şifre politikalarıdır. Politika uygulama (Policy enforcement) özelliği yerel Windows şifre politikalarını izleyerek, kurumsal ölçekte sadece Windows Server sistemlerde değil, SQL Server sistemlerde de tutarlı bir güvenlik politikası oluşturulmasını mümkün kılmaktadır. [33] SQL

Server, parola karmaşıklığı, parola geçerlilik süresi ve hesap kilitlenmesi gibi şifre politikalarını destekler. Tüm bu güvenlik önlemleri, varsayılan kurulumda aktif haldedir ancak kurulum sonrasında değiştirilebilmektedir. SQL Server şifre politikalarını sys.sql_logins katalog görüntüsünde (view) saklamaktadır.

4.1.2. Yetkilendirme (Authorization)

SQL Server veritabanının yetkilendirme sisteminin prensibi, veritabanı yöneticisinin kime hangi yetkilerin verildiğini tam olarak kontrol edebilmesini ve kullanıcının mümkün olan en az yetki ile çalışmasını sağlamaktır. SQL Server'in yetkilendirme ile ilgili bazı güvenlik özellikleri şunlardır:

- Hiyerarşik güvenlik seviyeleri (Scopes) : Hiyerarşik güvenlik seviyeleri (scopes) farklı seviyelerdeki nesnelerin, kullanıcıların (principals), emniyete alınabilir varlıkların (securables) ve izinlerin (permissions) hiyerarşik düzenlenmesini sağlayarak, güvenliğin daha iyi ve kolay yönetilebilmesini sağlar.
- Kullanıcı ve şemanın bağımsızlığı: SQL Server'da nesne isim uzayları nesne sahibinden bağımsız olarak, şema kullanılarak tanımlanır. Nesne sahibi ve nesne isim uzayının birbirinden bağımsız hale getirilmesi, daha esnek ve yönetilebilir bir iskelet sağlar.
- Sınırlı metadata görüntülenmesi: SQL Server'da katalog görüntüleri (catalog views), metadatanın sınırlı olarak görüntülenmesini sağlar. Varsayılan olarak kullanıcılar, izinli oldukları nesnelerin metadatalarını görebilirler. [34]
- Çalıştırma koşulu (Execution context): CREATE komutu içerisinde yordam (procedure) ve foksiyonlar gibi programlanabilir modüllerin hangi güvenlik yetkileriyle çalışabileceği ayarlanabilmektedir.

SQL Server da güvenlik; kullanıcılar (principals), emniyete alınabilir varlıklar (securables) ve izinler (permissions) tabanlıdır.

4.1.2.1 Kullanıcılar (Principals)

Sisteme erişen bireysel veya grup olarak var olan kullanıcılardır. Bunlar 3 seviyede olabilmektedir.

- Windows seviyesinde, Windows gruplarını, etki alanı kullanıcı hesaplarını ve yerel kullanıcı hesaplarını içerir.
- SQL Server seviyesinde, SQL Server girişlerini ve SQL Server rollerini kapsar.
- Veritabanı seviyesinde, kullanıcılar, veritabanı rolleri ve uygulama (application) rolleri olmak üzere 3 seviye vardır. Uygulama rolleri, istemci uygulama tabanlı alternatif güvenlik katmanı sağlamak için kullanılırlar.

4.1.2.2. Emniyete alınabilir varlıklar (Securables)

Sistemde güvenliğinin sağlanması gereken nesnelerdir. Emniyete alınabilir varlıklar farklı seviyelerde olurlar:

- Sunucu seviyesi,
- Veritabanı seviyesi,
- Şema seviyesi.

En geniş seviye olan sunucu seviyesi, girişler (logins), http endpoints, serfifikalar, olay bildiricileri (event notification) gibi emniyete alınabilir varlıklar içerirler. Veritabanı seviyesi ise, servisler, assemblyler, XML şemalar, roller, DDL olayları gibi emniyete alınabilir varlıklar içerirler. Şema seviyesi de, tablolar, görüntüler (view), yordamlar (prosedürler), sıralar (queue) gibi emniyete alınabilir varlıklar içerir. [33]

4.1.2.3. İzinler (Permissions)

Bir kullanıcının veritabanı nesneleri üzerinde belirli işlemleri gerçekleştirmesi için gerekli olan yetkilerdir. SQL Server'da GRANT, REVOKE ve DENY komutları emniyete alınabilir varlıklar (securable) üzerinde kullanıcıların (principals) işlem

yapmalarını kontrol ederler. Daha üst sevideki bir izin, daha düşük seviyelerdekilerde de geçerli olur. İzinlerin bazıları aşağıdaki tabloda listelenmiştir.

Tablo 4.1: SQL Server veritabanında izinler.

Emniyete alınabilir varlık (Securable)	**İzin (Permission)**	**Tanımı**
Sunucu	ALTER ANY LOGIN	Sunucu seviyesindeki herhangi bir login'i değiştirir.
	CONNECT_SQL	SQL Server'a bağlanır.
	CREATE LOGIN	Kullanıcı oluşturur.
	CONTROL SERVER	Sunucu yönetim kontrolü.
Login	IMPERSONATE	Kullanıcının başka bir kullanıcının yetkisini kullanmasını sağlar.
	ALTER	Kullanıcıyı değiştirir.
Veritabanı	CREATE TABLE	Veritabanında tablo oluşturur.
	CONTROL	Veritabanın tüm kontolünü sağlar.
	ALTER ANY USER	Veritabanındaki herhangi bir kullanıcıyı değiştirir.
Şema	SELECT	Şemadaki herhangi bir nesneden satırları seçer.
	ALTER	Şemadaki herhangi bir nesneyi değiştirir.
	TAKE OWNERSHIP	Şemanın sahipliğini değiştirir.
Kullanıcı	ALTER	Belirtilen kullanıcıyı değiştirir.
Tablo	SELECT	Tablodan satırları seçer
	ALTER	Tabloyu değiştirir.
	CONTROL	Tabloyu kontrol eder.

4.1.2.4. Roller ve şemalar

Roller, kullanıcı gruplarına, veritabanı sunucusu veya veritabanları üzerinde izin vermeyi kolaylaştıran metodlardır.

Roller, sunucu rolleri, değiştirilemeyen veritabanı rolleri, değiştirilebilen veritabanı rolleri ve uygulama rolleri olarak 4'e ayrılır.

- Sunucu rolleri, sunucunun yönetiminde kullanılan rollerdir. Örneğin, sysadmin.
- Değiştirelemeyen veritabanı rolleri, veritabanının yönetimi için kullanılıp değiştirilemezler. Örneğin, db_ddladmin, db_owner gibi.

- Değiştirilebilir veritabanı rolleri, çeşitli izinlerin grubundan oluşan rollerdir.
- Uygulama rolleri, kullanıcı yerine uygulamaya haklar verilmesidir. Üyesi yoktur.

Şemanın amacı bir isim alanı gibi davranarak ilişkili veritabanı nesnelerini bir arada tutmak ve isim çakışmalarını önlemektir. İki tablonun isminin aynı olması gerekiyorsa, farklı şemalar altında olmak şartıyla kullanılabilir.[18]

SQL Server veritabanları birden çok şema içerebilir. Her şema kendisine özgü bir kullanıcı veya rol olan kullanıcıya (principal) aittir. İsim çözümleme işlemleri için her kullanıcı varsayılan bir şemaya sahiptir, veritabanı nesneleri ise şema içinde yer alır. Şema içinde yeni veritabanı nesneleri oluşturmak için ilgili nesne için CREATE, ilgili şema üzerinde ise ALTER veya CONTROL yetkileri olması gerekir. Aşağıdaki kod, bir şema oluşturulmasını ve kullanıcıya verilmesini örneklemektedir.

```
/* Yeni bir giriş (login) oluştur */
CREATE LOGIN UserA WITH PASSWORD = 'ABC123#$'
GO
/* Oluşturulan login için bir kullanıcı oluştur. Şema var olmak zorunda değildir */
CREATE USER UserA FOR LOGIN UserA
  WITH DEFAULT_SCHEMA = MySchema
GO
/* Şemayı oluştur ve sahibi ile ilişkilendir */
CREATE SCHEMA MySchema AUTHORIZATION UserA
GO
/* Yeni şemanın içinde bir tablo oluştur */
CREATE TABLE MySchema.Table1 (col1 char (20))
GO
```

Örneğin ilk satırı UserA adlı yeni bir login oluşturuyor ve bir parola belirliyor. İkinci satır oluşturulan login için yeni bir kullanıcı oluşturuyor ve varsayılan şemayı belirtiyor. Bu aşamada belirtilen şemanın varolması gerekmemektedir. Yeni kullanıcı oluştururken varsayılan şemayı belirtilmezse dbo ile aynı varsayılan şema kullanılacaktır. Sonraki adımda CREATE SCHEMA ifadesi ile MySchema adlı şema oluşturuluyor. AUTHORIZATION ifadesi şemanın sahibinin UserA olduğunu belirtiyor ve şema içinde Table1 adlı bir tablo oluşturuluyor.

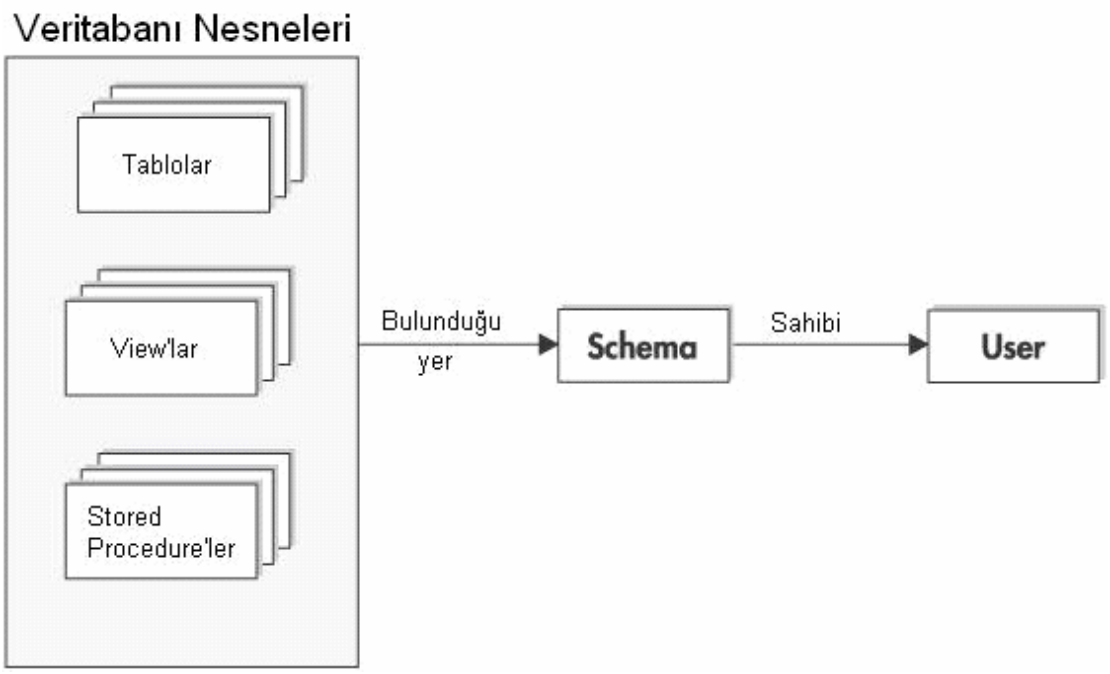
4.1.2.5. Kullanıcı - şema ayrımı

SQL Server 2000 veritabanında, veritabanı nesneleri doğrudan kullanıcılara aitti ve bu kullanıcılar sys_users sistem tablosunda kayıtlı tutulmaktaydı. SQL Server 2005'te bu yapı tümüyle değiştirildi ve veritabanı nesnelerinin sahipliği şemalara verildi. Kullanıcılar ise artık doğrudan veritabanı nesnelerinin sahibi durumunda değil, şemaların sahibi olmaktadırlar. SQL Server 2005 ile birlikte kullanıcılar ve diğer güvenlik kullanıcıları sys.database_principals adlı görüntü (view) ile görüntülenebilmektedir. SQL Server 2005'in şema listesine ise sys.schemas görüntüsü (view) ile erişilebilir. [33]

Şema, çeşitli nesneleri barındıran bir yapı olarak düşünülebilir. Şema, SQL Server tarafından kullanılan dört bölümlü isimlendirme yapısının üçüncü bölümü tarafından tanımlanmaktadır. Aşağıdaki örnek SQL Server isimlendirme söz dizimi örneklemektedir.

```
sunucu_adi.veritabani_adi.schema_adi.nesne_adi
```

SQL Server'ın önceki tüm sürümlerinde şema ismi ve sahibinin ismi genellikle aynı olurdu. SQL Server 2005 ile birlikte sahibi (owner), şemadan ayrılmış durumdadır. SQL Server 2005'te kullanıcı ve şemanın ayrılmasının temel nedeni, bir kullanıcıya ait çok sayıda nesnenin sahipliğini değiştirme ihtiyacıydı. Ek olarak, bir veritabanı nesnesinin isminin değişmesi aynı zamanda bir isim değişikliğine neden oluyordu. Örneğin MyDB veritabanında yer alan Tablo1'in sahibi KullanıcıA'dan KullanıcıB olarak değiştirildiğinde tablonun ismi MyDB.KullanıcıA.Tablo1'den MyDB.KullanıcıB.Tablo1'e dönüşmektedir. Bu problemi önlemek için çoğu kurum tüm veritabanı nesnelerinin dbo tarafından sahiplenilmesi şeklinde bir standart uygulama kullanmaya başlamıştı.



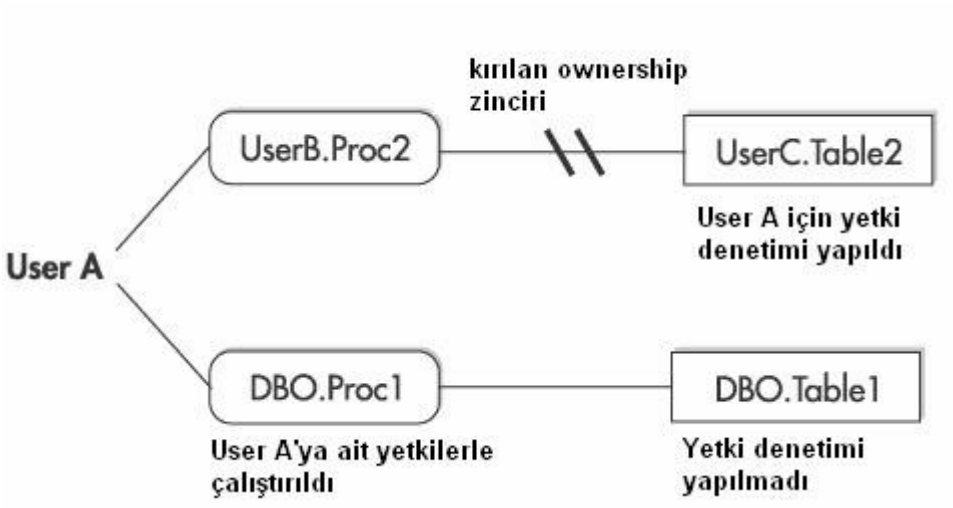
Şekil 4.1: SQL Server 2005 Veritabanı sahiplik zinciri.

SQL Server 2005 ile birlikte veritabanı nesneleri şemalar içinde bulundurulmaya ve şemalar kullanıcılar tarafından sahiplenmeye başlandı. Bu yeni soyutlama modeli veritabanı nesnelerinin sahibinin belirlenmesi ve değiştirilmesini kolaylaştırarak daha kolay yönetilebilir bir yapı oluşmasını sağlamıştır. SQL Server 2005'te veritabanında herhangi bir nesnenin sahibi durumundaki bir kullanıcının silinmesi sadece, veritabanı yöneticisinin ilgili tüm nesnenin/nesnelerin sahipliğini değiştirmesi değil, sadece bu nesneleri içeren şemanın sahibinin değiştirilmesi anlamına gelmektedir. Bu veritabanı yöneticisinin yapması gereken işlem ve müdahale edilmesi gereken nesne sayısını azaltarak, ilgili şemanın güncellenmesinden sonra ilgili kullanıcı hesabının drop edilerek işlerin çok daha hızlı tamamlanabilmesini sağlamaktadır. Ayrıca nesnenin / nesnelerin ismi değişmemiş olur.

4.1.2.6. Çalıştırma içeriği (Execution Context)

Bu özellik kayıtlı yordam (stored procedure) çalıştırma koşulu (Stored Procedure Execution Context) olarak bilinse de, kayıtlı yordamlar dışındaki modüller için de (fonksiyonlar veya assemblyler) geçerlidir. [33] Bir modül için çalıştırma içeriği (execution context) belirtmek modülün içerdiği tüm ifadelerin belirtilen çalıştırma içeriğine (execution context) bağlı olarak yetki denetiminden geçirilmesine neden

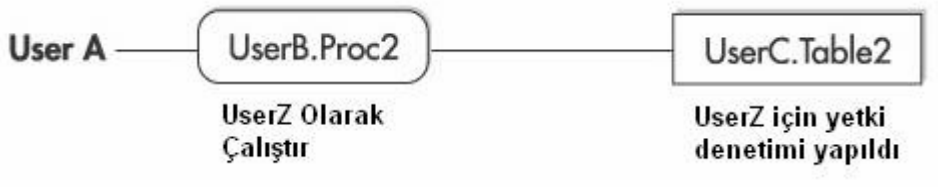
olacaktır. Diğer bir deyişle, bir modül için çalıştırma içeriği (execution context) belirtilmesi, modülün kapsadığı tüm ifadelerin, modülü çalıştıran kullanıcı yerine belirtilen kullanıcının sahipliğinde çalıştırılması anlamına gelmektedir (identity impersonation). Şekil 4.2' SQL Server 2005'in sahiplik mimarisi göstermektedir.



Şekil 4.2: SQL Server 2005 sahiplik mimarisi örneği.

UserA'nın dbo.Proc1'i çalıştırabilmesi için UserA'nın bu nesne üzerinde execute yetkisinin olması gerekiyor. Ancak dbo.Proc1, dbo.Table1'e eriştiğinde herhangi bir yetki denetimi yapılmıyor, çünkü dbo her iki nesneninde sahibi durumunda. Sonraki senaryoda, UserA'nın UserB.Proc2'yi çalıştırabilmesi için, bu nesne üzerinde yetkiye gereksinimi var. Sonrasında UserB.Proc2, UserC.Table2 nesnesine erişmek istediğinde, UserA için SELECT yetkisi denetlenmek durumunda. Bu senaryoda, UserB.Proc2 ve UserC.Table2 farklı sahiplere sahip olduğundan sahiplik zinciri kırılıyor.

SQL Server 2005 bu senaryoyu şekil 4.3'te görüldüğü üzere oldukça kolaylaştırıyor. Bu senaryoda UserA, UserB.Proc2'yi çalıştırmak istiyor ve SQL Server, UserA'nın UserB.Proc2 için execute yetkisinin olup olmadığını denetliyor. Eğer UserB.Proc2 nesnesi, UserZ'nin çalıştırma içeriğinde (contextinde) çalıştırılacak şekilde oluşturulursa, UserB.Proc2 kayıtlı yordamı UserC.Table2 nesnesine erişmeye çalıştığında SQL Server sadece işlemi gerçekleştiriyor olarak kabul edilecek kullanıcının, yani bu örnek için UserZ'nin yetkilerini denetleyecektir. İşlemi başlatan kullanıcı olan UserA için SELECT yetkisi aranmayacaktır.



Şekil 4.3: SQL Server 2005 çalıştırma içeriği (execution context) örneği

Aşağıdaki kod Proc2 adlı bir stored procedure'ın çalıştırma içeriğinin (execution context) nasıl değiştirebileceği örneklemektedir.

```
ALTER PROC MySchema.Proc2 WITH EXECUTE AS USER UserZ
```

Bu ifade WITH EXECUTE ifadesini kullanmaktadır. Burada MySchema içinde yer alan Proc2 adlı stored procedure UserZ'nin sahipliği içeriğinde (context) çalıştırılmak üzere güncelleniyor. Bu işlemde bir kullanıcı adı belirtmek gerekir, bir rol ismi kullanılamaz. "EXECUTE AS" yapılabilmesi için kullanıcının, stored procedure ya da fonksiyon üzerinde EXECUTE iznine sahip olması gerekir. Çalıştırma içeriğinde (execution context) yapılan değişiklikler sys.sql_modules view'ı aracılığıyla görüntülenebilmektedir. [33]

4.1.2.7. Katalog güvenliği

Farklı veritabanlarında ve master veritabanında saklanan SQL Server 2000 tarafından kullanılan sistem tabloları, SQL Server 2005'te katalog görüntülerinde (catalog view) toplanmıştır. Bu görüntüler tarafından sunulan metadata varsayılan olarak korunma altına alınmıştır ve çok kısıtlı public yetki söz konusudur. SQL Server 2005'in katalog görüntüleri (catalog view), içerdikleri verilere sadece nesnelerin sahibi olunması veya gerekli yetkilere sahip olma durumunda erişebilmeyi sağlayan, kayıt seviyesi güvenlik (row-level security) uygulamaktadır. [34]

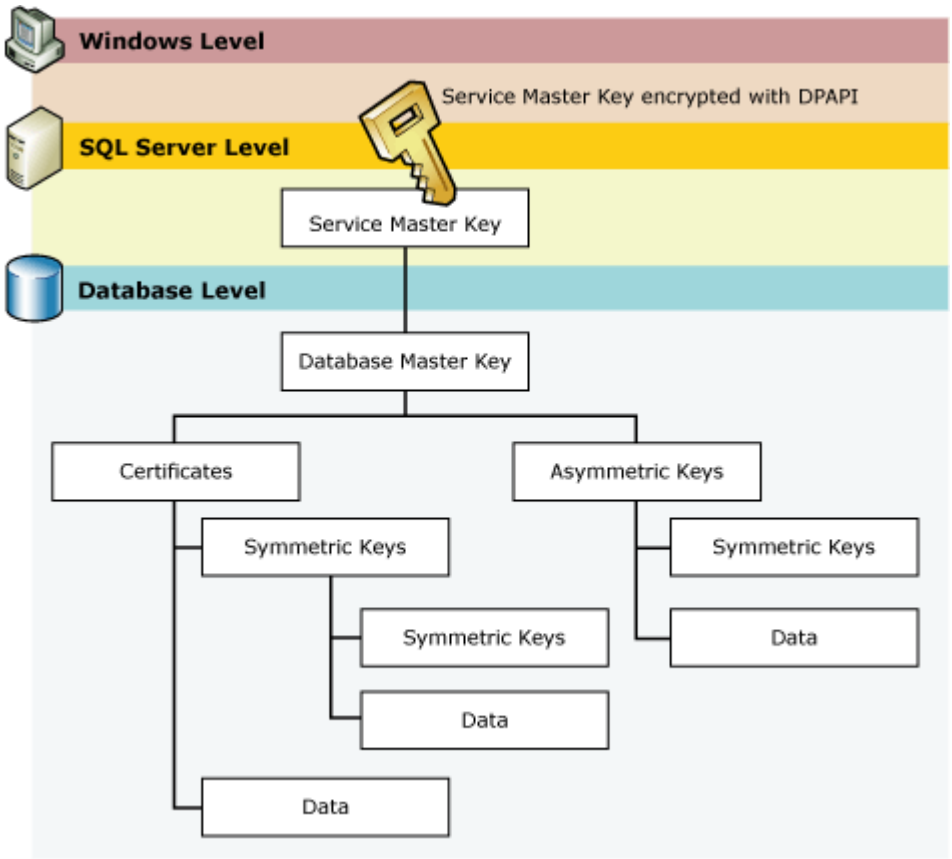
Bir kullanıcı veya rolün metadata'ya erişmesini sağlamak için veritabanı yöneticisinin VIEW DEFINITION yetkisini kullanması gerekir. VIEW DEFINITION yetkisi bir kullanıcıya, sahibi olmadığı veya erişim hakkı olmayan bir nesne üzerinde, nesnenin metadata'sına erişme imkânı vermek amacıyla kullanılır. VIEW DEFINITION yetkisi sunucu, veritabanı, şema veya nesne seviyesinde uygulanabilir.

4.1.2.8. Ağ üzerinde şifreleme (Network encryption)

SQL Server, ağ trafiğinin şifrelenmesi için Windows tarafından desteklenen IPSec ve SSL protokollerini kullanabilir. SSL, IIS ve SQL Server entegrasyonu ile sağlanabildiği gibi, SQL Server 2005 içindeki sertifika sunucusu bileşeni de kullanılabilir. [33]

SQL Server 2005, ağ üzerinde şifreleme için gerekli olan SSL protokolünü sağlayacak şekilde, kendi sertifika sunucusu ile birlikte gelir. Üretilen sertifikalar, X.509 v3 standartlarını sağlar ve SSL bağlantıları oluşturmak için SQL Server 2005 tarafından kullanılır.

4.1.3. Veri şifreleme (Data Encryption)



Şekil 4.4: SQL Server veri şifreleme mimarisi. [33]

SQL Server veritabanında, şifreleme mekanizması yukarıdaki grafikte gösterildiği üzere hiyerarşik olarak yapılmaktadır. Şifrelemenin temelinde ise sertifikalar ve

anahtarlar bulunmaktadır. Sertifikalar, SQL Server tarafından oluşturulabileceği gibi ticari amaçlı kullanılan X.509 sertifikaları veya Windows 2003 Server CA tarafından verilen sertifikalar olabilir. [33] Sertifikaların haricinde simetrik ve asimetrik olmak üzere iki tip anahtar kullanılır.

Service Master Key, SQL Server şifreleme hiyerarşisinin en tepesinde yer alır. Sunucu ilk kurulduğunda otomatik olarak üretilir. Windows Data Protection API ve yerel makine anahtarı kullanılarak şifrelenir. [33] Bu anahtara yalnızca Windows servis hesabı altından erişilebilir.

Service master key üzerinde yapılabilecek işlemler:

- Service master key'in yapısını değiştirme:

```
ALTER SERVICE MASTER KEY
   [ { <regenerate_secenegi> | <recover_secenegi> } ]
   |
   [ { ADD | DROP } ENCRYPTION BY MACHINE KEY ]
<regenerate_secenegi> ::=
   [ FORCE ] REGENERATE
<recover_secenegi> ::=
   { WITH OLD_ACCOUNT = 'hesap_adi' , OLD_PASSWORD = 'sifre' }
   |
   { WITH NEW_ACCOUNT = 'hesap_adi' , NEW_PASSWORD = 'sifre' }
```

REGENERATE: Service master key'in yeniden üretilmesi için kullanılır.
OLD_ACCOUNT = 'hesap_adi': Eski Windows servis hesap adı.
OLD_PASSWORD = 'sifre': Eski Windows servis hesabının şifresi.
NEW_ACCOUNT = 'hesap_adi': Yeni Windows servis hesap adı.
NEW_PASSWORD = 'sifre': Yeni Windows servis hesabının şifresi.

- Service master key'i yedekleme:

```
BACKUP SERVICE MASTER KEY TO FILE = 'dosya_yolu' ENCRYPTION BY
PASSWORD = 'sifre'
```

- Service master key'i yükleme:

```
RESTORE SERVICE MASTER KEY FROM FILE = 'dosya_yolu' DECRYPTION BY
PASSWORD = 'sifre' [FORCE]
```

Şifreleme işlemini yapabilmek için yukarıdaki grafikte görüldüğü üzere her bir veritabanı için bir database master key'e ihtiyaç duyulmaktadır. Bu oluşturulan database master key'in amacı, sertifikaların gizli anahtarlarını ve asimetrik anahtarları güvenli bir şekilde veritabanında muhafaza etmektir. Database master key, bir simetrik anahtar olup, Triple DES algoritması ve kullanıcının belirleyeceği bir parola ile şifrelenir. Herhangi bir zamanda, database master key'e ulaşmak için bu anahtarın bir kopyası, service master key'le şifrelenerek, master veritabanında saklanır. Service master key tarafından şifrelenmiş database master key'ler master veritabanındaki sys.databases görüntüsünün (view) is_master_key_encrypted_ by_server sütununda bulunur. Database master key'ler hakkında bilgiler ise sys.symmetric_keys görüntüsünde (view) bulunur.

Database master key üzerinde yapılabilecek işlemler:

- Database master key oluşturma:

```
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'sifre'
```

- Database master key'i açma:

```
OPEN MASTER KEY DECRYPTION BY PASSWORD = 'sifre'
```

- Database master key'i kapama:

```
CLOSE MASTER KEY
```

- Database master key'i yedekleme:

```
BACKUP MASTER KEY TO FILE = 'dosya_yolu' ENCRYPTION BY PASSWORD = 'sifre'
```

- Database master key'i yedeklenmiş bir dosyadan yükleme:

```
RESTORE MASTER KEY FROM FILE = 'dosya_yolu'
   DECRYPTION BY PASSWORD = 'sifre'
   ENCRYPTION BY PASSWORD = 'sifre'
   [ FORCE ]
```

- Database master key'i yeniden üretme:

```
ALTER MASTER KEY < degistirme_secenegi>
< degistirme_secenegi > ::=
   <uretme_secenegi> | <sifreleme_secenegi>
< uretme_secenegi > ::=
   [ FORCE ] REGENERATE WITH ENCRYPTION BY PASSWORD = 'sifre'
< sifreleme_secenegi > ::=
   ADD ENCRYPTION BY [ SERVICE MASTER KEY | PASSWORD = 'sifre']
   |
   DROP ENCRYPTION BY [ SERVICE MASTER KEY | PASSWORD = 'sifre']
```

- Database master key'i silme:

```
DROP MASTER KEY
```

Sertifikalar üzerinde yapılabilecek işlemler:

- Sertifika oluşturma:

```
CREATE CERTIFICATE sertifika_adi [ AUTHORIZATION kullanici_adi ]
   { FROM <mevcut_anahtarlar> | <yeni_uretilecek_anahtarlar> }
   [ ACTIVE FOR BEGIN_DIALOG =  { ON | OFF } ]
<mevcut_anahtarlar> ::=
   ASSEMBLY assembly_adi
   | {
      [ EXECUTABLE ] FILE = 'dosya_yolu'
      [ WITH PRIVATE KEY ( <private_key_secenekler> ) ]
     }
<yeni_uretilecek_anahtarlar> ::=
   [ ENCRYPTION BY PASSWORD = 'sifre']
   WITH SUBJECT = 'setifikanin_konusu'
   [ , <date_secenekler> [ ,...n ] ]
<private_key_secenekler> ::=
   FILE = 'private_key_dosya_yolu'
   [ , DECRYPTION BY PASSWORD = 'sifre' ]
   [ , ENCRYPTION BY PASSWORD = 'sifre' ]
<date_secenekler> ::=
   START_DATE = 'mm/dd/yyyy' | EXPIRY_DATE = 'mm/dd/yyyy'
```

Burada, AUTHORIZATION user_name, sertifikanın sahibi olacak kullanıcı, ASSEMBLY assembly_adi, veritabanına yüklenmiş olan assembly adı, [EXECUTABLE] FILE sertifikayı barındıran dosya, ACTIVE FOR BEGIN_DIALOG ise ilk standart değeri ON olup sertifikayı başlatmak için kullanılır.

- Mevcut bir sertifikanın yapısını değiştirme:

```
ALTER CERTIFICATE sertifika_adi
   REMOVE PRIVATE KEY
   |
   WITH PRIVATE KEY ( <private_key_ozellik> [ ,... ] )
   |
   WITH ACTIVE FOR BEGIN_DIALOG = [ ON | OFF ]
<private_key_ozellik> ::=
   FILE = 'private_key_dosya_yolu'
   |
   DECRYPTION BY PASSWORD = 'key_sifre'
   |
   ENCRYPTION BY PASSWORD = 'sifre'
```

- Sertifikayı yedekleme:

```
BACKUP CERTIFICATE certname TO FILE = 'dosya_yolu'
   [ WITH PRIVATE KEY
     (
       FILE = 'private_key_ dosya_yolu' ,
       ENCRYPTION BY PASSWORD = 'encryption_sifre'
       [ , DECRYPTION BY PASSWORD = 'desifreleme_sifresi' ]
     ) ]
```

- Bir Sertifikayı silme

```
DROP CERTIFICATE sertifika_adi
```

Simetrik anahtar üzerinde yapılabilecek işlemler:

- Simetrik anahtar üretme:

```
CREATE SYMMETRIC KEY anahtar_adi [ AUTHORIZATION sahibinin_adi ]
   WITH <key_secenekler> [ , ... n ]
   ENCRYPTION BY <sifreleme_mekanizmasi> [ , ... n ]
<sifreleme_mekanizmasi> ::=
   CERTIFICATE sertifika_adi
   |
   PASSWORD = 'sifre'
   |
   SYMMETRIC KEY symmetric_anahtar_adi
   |
   ASYMMETRIC KEY asym_anahtar_adi
<key_secenekler> ::=
```

```
   KEY_SOURCE = 'sifre_bilgisi'
   |
   ALGORITHM = <algoritma>
   |
   IDENTITY_VALUE = 'identity_bilgisi'
<algoritma> ::=
   DES | TRIPLE_DES | RC2 | RC4 | RC4_128
   | DESX | AES_128 | AES_192 | AES_256
```

Anahtar adı veritabanında daha önce kullanılmamış bir anahtar ismi olmalıdır. AUTHORIZATION sahibinin_adi veritabanı veya uygulama kullanıcı adı olup anahatarın sahibi olacak kişi, CERTIFICATE kullanılacak sertifika adı, ALGORITHM simetrik anahtar için kullanılacak algoritmadır. Oluşturulacak anahtarda şifreleme algoritması olarak DES, TRIPLE_DES, RC2, RC4, RC4_128, DESX, AES_128, AES_192, AES_256 algoritmaları da kullanılabilir.

- Simetrik anahtarı açma:

```
OPEN SYMMETRIC KEY anahtar_adi DECRYPTION BY <desifreleme_mekanizmasi>
<desifreleme_mekanizmasi> ::=
   CERTIFICATE sertifika_adi [ WITH PASSWORD = 'sifre' ]
   |
   ASYMMETRIC KEY asym_anahtar_adi [ WITH PASSWORD = 'sifre' ]
   |
   SYMMETRIC KEY desifreleme_anahtar_adi
   |
   PASSWORD = ‘desifreleme_sifresi’
```

- Simetrik anahtarın yapısını değiştirme

```
ALTER SYMMETRIC KEY anahtar_adi <alter_secenegi>
<alter_secenegi> ::=
  ADD ENCRYPTION BY <sifreleme_mekanizmasi> [ , ... n ]
  |
  DROP ENCRYPTION BY <sifreleme_mekanizmasi> [ , ... n ]
<sifreleme_mekanizmasi> ::=
  CERTIFICATE sertifika_adi
  |
  PASSWORD = 'sifre'
  |
  SYMMETRIC KEY Simetrik_Anahtar_Adi
  |
  ASYMMETRIC KEY Asimetrik_Anahtar_Adi
```

Asimetrik anahtar üzerinde yapılabilecek işlemler:

- Asimetrik anahtar oluşturma:

```
CREATE ASYMMETRIC KEY Asimetrik_Anahtar_Adi
  [ AUTHORIZATION veritabani_kullanici_adi]
  {
    FROM <Asym_Key_Kaynagi>
    |
    WITH ALGORITHM = { RSA_512 | RSA_1024 | RSA_2048 }
  }
    [ ENCRYPTION BY PASSWORD = 'sifre' ]
<Asym_Key_Kaynagi>::=
  FILE = ‘dosya_yolu’
  |
  EXECUTABLE FILE = ‘exe_dosya_yolu’
  |
  ASSEMBLY Assembly_Name
```

Asimetrik anahtar oluşturma da ise algoritma olarak RSA_512, RSA_1024, RSA_2048 kullanılabilir.

- Asimetrik anahtarın yapısını değiştirme:

```
ALTER ASYMMETRIC KEY Asimetrik_Anahtar_Adi <alter_secenegi>
<alter_secenegi> ::=
   <sifre_degistirme_secenegi>
   |
   REMOVE PRIVATE KEY
<sifre_degistirme_secenegi> ::=
   WITH PRIVATE KEY ( <sifre_secenegi> [ , <sifre_secenegi> ] )
<sifre_secenegi> ::=
   ENCRYPTION BY PASSWORD = 'sifre'
   |
   DECRYPTION BY PASSWORD = 'eski_sifre'
```

4.1.3.1. Veri şifreleme fonksiyonları

SQL Server veri şifreleme ve deşifrelemek için aşağıdaki fonksiyonları kullanır:

- EncryptByCert
- DecryptByCert
- EncryptByKey
- DecryptByKey
- EncryptByAsymKey
- DecryptByAsymKey

Bu fonksiyonların kullanım şekilleri şu şekildedir:

- EncryptByCert (certificate_ID , { 'acik_metin' | @acik_metin })
- DecryptByCert (certificate_ID , { 'sifreli_metin' | @sifreli_metin } [, { 'cert_sifre' | @cert_sifre }])
- EncryptByKey (key_GUID , { 'acik_metin' | @acik_metin } [, { add_authenticator | @add_authenticator } , { authenticator | @authenticator }])
- DecryptByKey ({ 'sifreli_metin' | @sifreli_metin } [, add_authenticator , { authenticator | @authenticator }])
- EncryptByAsymKey (Asym_Key_ID , { 'acik_metin' | @acik_metin })
- DecryptByAsymKey (Asym_Key_ID , { 'sifreli_metin' | @sifreli_metin } [, 'Asym_Key_Sifresi'])

Veri şifrelemesi için örnek bir uygulama yapalım:

Adım 1: Öncelikle verilerin bulunacağı veritabanı ve tabloyu oluşturalım.

```
CREATE DATABASE VeriSifrelemeOrnek
GO
USE VeriSifrelemeOrnek
--Db Kullanıcısı ve giriş oluşturuluyor
CREATE LOGIN [VeriSifrelemeDbKullaniciGiris] WITH PASSWORD = "Sifre#123"
CREATE USER [VeriSifrelemeDbKullanici] FOR LOGIN [VeriSifrelemeDbKullaniciGiris]
G0
CREATE TABLE KrediKartBilgileri
(
    KartId INT PRIMARY KEY IDENTITY,
    KartNo VARBINARY(256)
)
```

Adım 2: Şifrelemede kullanmak için bir database master key oluşturalım.

```
USE VeriSifrelemeOrnek;
CREATE MASTER KEY ENCRYPTION BY PASSWORD = '23987hxJ#KL95234nl0zBe';
GO
```

Adım 3: Database master key'i oluşturduktan sonra sıra sertifikaya geldi. Mevcut bir sertifikamız olmadığı için sertifika oluşturalım.

```
CREATE CERTIFICATE [sertifikam] WITH SUBJECT = ‘Şifreleme Sertifikasi’, ENCRYPTION PASSWORD = ‘jfdsij380fukanfjcxvDJEOD#$fksdwr’, EXPIRY_DATE = ‘12/31/2008’;
```

Adım 4: Şifreleme işlemini yapacak olan anahtari üretelim.

```
CREATE SYMMETRIC KEY [anahtarim] WITH ALGORITHM = Triple_Des ENCRYPTION BY CERTIFICATE [sertifikam]
```

Adım 5: Oluşturulan anahtarın bir adım önce oluşturulan sertifikayı kullanacağı belirlendi. Artık sıra verileri tablolara eklerken şifreleyip eklemek ve verilerimizi çekerken şifreleri çözüp çekmektir. Verileri eklerken veya çekerken mutlaka ilk önce simetrik anahtar (symmetric key) açılmalıdır. Anahtarı açarken, bu anahtarın verileri belirtilen sertifikadan çözeceği belirtilir.

```
OPEN SYMMETRIC KEY [anahtarim] DECRYPTION BY CERTIFICATE [sertifikam]
```

Adım 6: Anahtarımızı açtıktan sonra key_Guid() fonksiyonundan yararlanmak gerekiyor. Bu fonksiyon veritabanındaki simetrik anahtar (symmetric key) GUID’ini verir. Döndürülen bu GUID ise mevcut verilerimizi şifrelerken kullanacağımız EncryptByKey methodunda kullanılacaktır.

```
DECLARE @KeyGuid AS UNIQUEIDENTIFIER
SET @KeyGuid = key_guid("Anahtarim")
IF( @KeyGuid is not null )
BEGIN
INSERT INTO [KrediKartBilgileri] VALUES (EncryptByKey(@KeyGuid, N"1234-1234-1234-1234"))
INSERT INTO [KrediKartBilgileri] VALUES (EncryptByKey(@KeyGuid, N"4546-5678-5678-5678"))
 --PRINT @KeyGuid
END
ELSE
BEGIN
 PRINT "simetrik key GUID hatalı"
END
```

Key_Guid() : Bu fonksiyon bir adet parametre alır ve aldığı parametre ise simetrik anahtar (symmetric key) adıdır. Simetrik anahtar (symmetric key) GUID’ini döndürür. sys.key_encryptions görüntüsünden (view) bilgilerine ulaşılır.

EncryptByKey(): Bu fonksiyon ise verileri şifrelemeyi sağlar. İki parametre alır bunlardan ilki şifrelemede kullanılacak olan simetrik anahtarın key_Guid değeri, ikincisi ise şifrelenecek veridir. VarBinary veri tipinde dönüş yapar.

Adım 7: İşlem bittikten sonra anahtar kapatılır.

```
CLOSE SYMMETRIC KEY [anahtarim]
```

Veriler artık şifreli olarak tablodadır. Mevcut halini listeleyecek olursak;

```
SELECT * FROM KrediKartBilgileri
```

	KartId	KartNo
1	1	0x000C625255574545A1A64EBCF0D3151901000000196B618...
2	2	0x000C625255574545A1A64EBCF0D3151901000000A05CA6...

Şekil 4.5: SQL Server veri şifreleme örneği (şifrelenmiş).

Şifreli verileri çözümleyebilmek için ise decryptbykey fonksiyonunu kullanılır. VarBinary cinsinden veri alır ve VarBinary cinsinden dönüş yapar. Şifre çözme işleminden önce anahtar açılmadıysa Null değer geri döner.

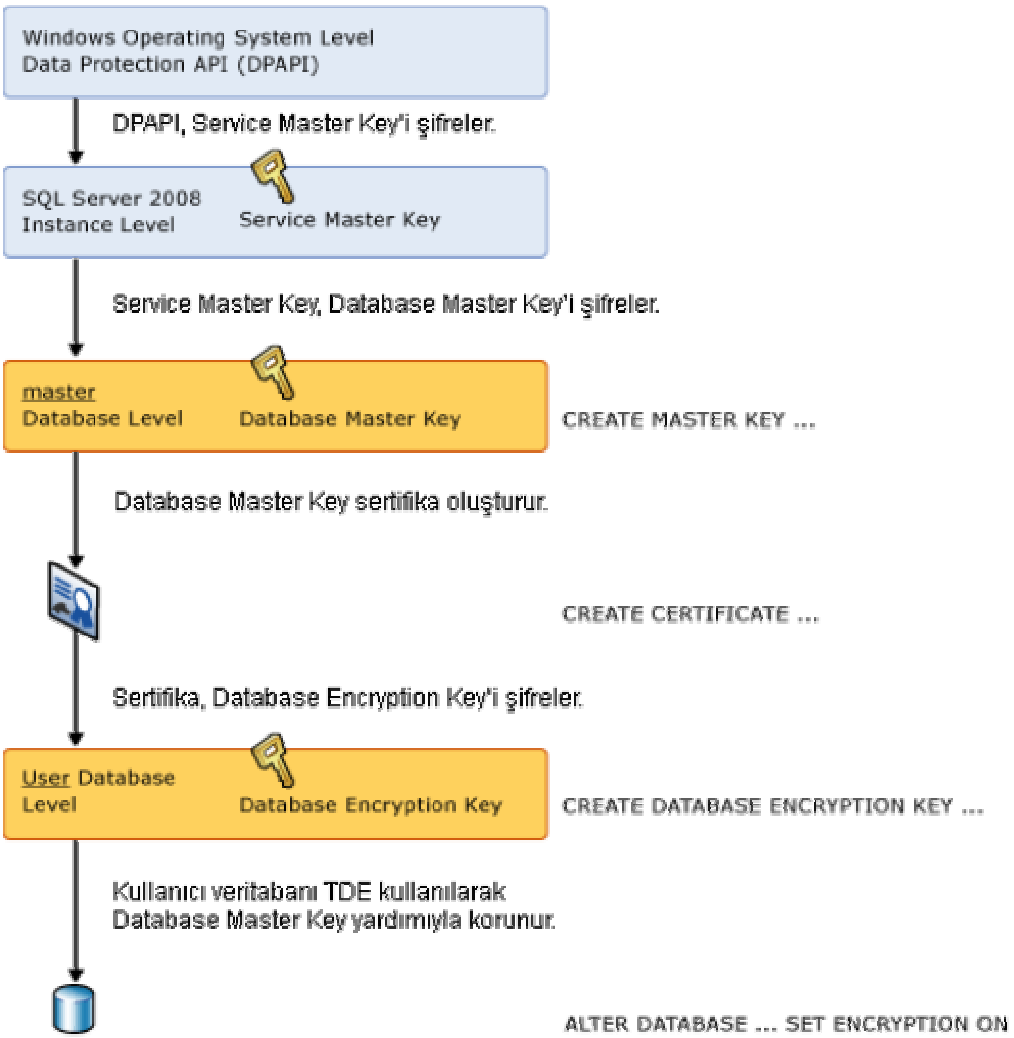
```
OPEN SYMMETRIC KEY [anahtarim] DECRYPTION BY CERTIFICATE [sertifikam]
SELECT KartId, convert( NVARCHAR(100), decryptbykey( KartNo )) as "Kredi Karti Numarasi" FROM KrediKartBilgileri
CLOSE SYMMETRIC KEY [anahtarim]
```

	KartId	Kredi Kartı Numarası
1	1	1234-1234-1234-1234
2	2	4546-5678-5678-5678

Şekil 4.6: SQL Server veri şifreleme örneği.

4.1.3.2. Saydam Veri Şifreleme (TDE, Transparent Data Encryption)

TDE (Transparent Data Encryption) iki şu iki özelliği sağlamaktadır: Veriyi kullanan uygulamaları etkilemeden verilerin şifrelenmesi ve tüm veritabanını şifreleme. TDE, veritabanı seviyesinde gerçek zamanlı şifreleme gerçekleştirmektedir. Hem veriler hem de log dosyaları diskte şifrelenmiş olarak tutulmaktadır ve hafızaya okunduklarında deşifrelenmektedirler. Bu da şifrelemeyi uygulamalar için tamamen saydam olarak sağlıyor. Bir veritabanını TDE ile şifreleme Veritabanı Şifreleme Anahtarı (Database Encryption Key - DEK) adı verilen özel bir anahtar ile yapılmaktadır. DEK, standart DDL (CREATE/ALTER/DROP) ile yönetilebilmektedir. Şifreleme durumu ALTER DATABASE komutu ile ve SET ENCRYPTION [ON | OFF] ile açılıp kapatılabilmektedir.



Şekil 4.7 : SQL Server Veritabanında Saydam Veri Şifreleme (TDE)

```
CREATE DATABASE ENCRYPTION KEY
    WITH ALGORITHM = { AES_128 | AES_192 | AES_256 | TRIPLE_DES_3KEY }
  ENCRYPTION BY SERVER
{
 CERTIFICATE sertifika_adi |
 ASYMMETRIC KEY asimetrik_anahtar_adi
}
```

CREATE DATABASE ENCRYPTION KEY komutu dışında, ALTER DATABASE ENCRYPTION KEY komutuyla bir veritabanını şifrelemek için kullanılan anahtar değiştirilebilir. DROP DATABASE ENCRYPTION KEY komutuyla şifreleme anahtarı düşürülebilir. ALTER DATABASE SET <db_sifreleme_secenegi> ::= ENCRYPTION { ON | OFF } komutuyla ise veritabanı için şifreleme açılıp kapatılabilmektedir.

Musteri veritabanını TDE kullanarak şifreleme işlemi yapan örnek bir kod:

```
USE master;
GO
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'sifre';
GO
CREATE CERTIFICATE Sertifikam WITH SUBJECT = 'DEK Sertifikam'
GO
USE Musteri
GO
CREATE DATABASE ENCRYPTION KEY
WITH ALGORITHM = AES_128
ENCRYPTION BY SERVER CERTIFICATE Sertifikam
GO
ALTER DATABASE Musteri
SET ENCRYPTION ON
GO
```

4.1.4. Hesap Denetimi

SQL Server logları, sisteme erişimleri ve belirlenen olayların kayıt altına alınmasını, denetleme ise, sisteme yetkili ya da yetkisiz erişimlerin izlenmesini ve bu erişimlere müdahale edilmesini sağlar.

SQL Server 2005, denetlemeyi "Windows Security Event Log", SQL Profiler ve Data Defination Language (DDL) trigger'ları ile sağlar.[27]

Windows Security Event Log, nesnelere erişim, hakların kullanımı ve kimlik doğrulama işlemlerinin görüntülenebildiği bir araçtır.

SQL Profiler, Security Log'larından daha detaylı bilgi verebilme kabiliyetine sahiptir. Trace özelliklerinden güvenlik denetim özellikleri konfigüre edilebilir. SQL Server, sistem yöneticisi haklarına sahip olmayan kullanıcıların da SQL Profiler çalıştırabilmelerine olanak tanıyan bir role de sahiptir. Bu sayede yazılım geliştiriciler veya sistem denetleyicilerinin de, minumum haklar ile sistemin güvenlik denetimlerini yapabilmeleri sağlanabilir.[33]

SQL Server 2005, veri tanımlama dili (DDL) sayesinde ise nesneler yaratıldığı veya silindiğinde, "SQL Server Notification Services" aracılığı ile bildirilmesi sağlanır.

4.2. Oracle Veritabanı Güvenliği

Oracle, veri ve kullanıcının çeşitli veritabanı kaynaklarına erişimi için geniş kapsamlı bir erişim kontrolü sağlamaktadır. Haklar yoluyla nesneler üzerinde kullanıcıların izinlerini düzenler. Hak, nesneye erişim için verilen izindir.

4.2.1. Doğrulama (Authentication)

Oracle veritabanı, yetkisiz kullanıcıları önlemek için kullanıcı kontrolü yapmaktadır. Bu kullanıcı doğrulama işlemi aşağıdaki yöntemlerin herhangi biri kullanılarak yapılabilir.

- İşletim sistemiyle kullanıcı doğrulama,
- Ağ yoluyla kullanıcı doğrulama,
- Oracle veritabanıyla kullanıcı doğrulama,
- Çok katmanlı kullanıcı doğrulama,
- Güvenli soket katmanlı (SSL) protokolüyle kullanıcı doğrulama,
- Veritabanı yöneticilerini doğrulama.

4.2.1.1. İşletim sistemiyle kullanıcı doğrulama

Oracle, Windows, UNIX, Linux ve Mac gibi işletim sistemlerinde kullanıcıların doğrulaması işlemini işletim sistemi yoluyla gerçekleştirebilir. Yararları şunlardır:

- İşletim sistemi tarafından bir kez doğrulama yapıldığında kullanıcı özel bir kullanıcı adı ve şifresi olmadan Oracle veritabanına bağlanabilir.
- Kullanıcı doğrulaması üzerindeki kontrol yetkisi işletim sistemindedir. Oracle veritabanı, şifrelerin saklaması ve bakımını yapma gereği duymaz.
- Hesap denetimi için veritabanı ve işletim sisteminde aynı kullanıcı adları kullanılabilir.
- İşletim sistemi kullanıcıları ve işletim sistemi dışındaki kullanıcıların her ikisi de aynı sistemde doğrulanabilir.
  - İşletim sistemi tarafından kullanıcıları doğrulama: CREATE USER komutu ve IDENTIFIED EXTERNALLY'i kullanarak kullanıcı hesabı yaratılır. OS_AUTHENT_PREFIX başlangıç parametresi olarak "OPS$" default değeri kullanılır. Bu parametre, Oracle veritabanı tarafından, bağlantı isteğinde bulunan kullanıcıları doğrulamada kullanır.[36]
  - İşletim sistemi dışında kullanıcı doğrulama: Bu kullanıcılar şifre belirlenerek veritabanı tarafından doğrulaması yapılır.
  - Oracle Veritabanı Kullanıcı Güvenliği kullanıcılarını doğrulama: Bu kullanıcı hesapları, CREATE USER komutunun IDENTIFIED GLOBALLY deyimi kullanılarak yaratılır ve Oracle Internet Dizini (OID) tarafından aynı veritabanında doğrulaması yapılır.

4.2.1.2. Ağ (Network) yoluyla kullanıcı doğrulama

Oracle aşağıdaki ağ kullanıcı doğrulama metotlarını desteklemektedir.

- Üçüncü parti tabanlı kullanıcı doğrulama,
- Açık anahtar altyapısı tabanlı kullanıcı doğrulama,
- Uzaktan kullanıcı doğrulama.

4.2.1.2.1. Üçüncü parti tabanlı kullanıcı doğrulama

Oracle veritabanı, PKI, Kerberos, RADIUS ve dizin tabanlı servisler gibi üçüncü parti yazılımları kullanarak ağ üzerinde kullanıcıları doğrulayabilir. [10]

4.2.1.2.2 Açık anahtar altyapısı tabanlı kullanıcı doğrulama (PKI)

PKI tabanlı doğrulamada dijital sertifikalar kullanılır. Bir doğrulama sunucusuna gerek olmadan direkt olarak sunucular üzerinde doğrulama imkânı sağlar. Oracle, PKI açık anahtarları ve sertifikaları kullanarak açık anahtar altyapısının kullanılmasını sağlar.[40]

4.2.1.2.3. Kerberos kullanarak kullanıcı doğrulama

Oracle, kullanıcıların doğrulamasını Kerberos doğrulamayla gerçekleştirebilir. Kerberos doğrulama ile ilgili detaylı bilgi Bölüm 3.5.4.' te yer almaktadır.

4.2.1.2.4. Uzaktan kullanıcı doğrulama

Oracle, kullanıcıların Remote Dial-In Kullanıcı Servisi (RADIUS) yoluyla uzaktan doğrulamasını destekler. Bu doğrulama için RADIUS sunucular kullanılmaktadır.

RADIUS (Remote Authentication Dial-in User Service) sunucular, uzaktan bağlanan kullanıcılar için kullanıcı adı, şifreyle doğrulama, hesap denetimi ve yetkilendirme işlemlerini yapar. Örneğin, işyeri dışından şirket ağına bağlanmak için bu yapı kullanılabilir, kullanıcılar tek tek yetkilendirilebilir. [19]

4.2.1.2.5. Dizin tabanlı servisleri kullanarak kullanıcı doğrulama

Oracle veritabanı için, merkezi bir dizin kullanarak etkili bir doğrulama yönetimi yapılabilir. Dizin tabanlı servisler şunları içerir:

- Oracle internet dizini: LDAP (Lightweight Directory Access Protocol) kullanarak, dağıtık ortamda yaratılmış olan kullanıcılar hakkındaki bilgileri saklamak ve yönetmek için merkezi bir depo kullanır. LDAP, TCP/IP üzerinde çalışan dizin servislerini sorgulama ve değiştirme amacıyla kullanılan uygulama katmanı protokolüdür. Bu protokol, OpenLDAP, Sun Directory Server, Microsoft Active Directory gibi dizin sunucuları tarafından kullanılmaktadır.[40]
- Oracle güvenlik yöneticisi, Oracle internet dizininden rolleri alma ve saklama imkânı sağlar. Merkezi olarak hakların yönetimini daha kolay yapılmasına ve güvenlik seviyesinin arttırılmasına katkı sağlar.

4.2.1.3. Oracle veritabanı yoluyla kullanıcı doğrulama

Oracle veritabanı, veritabanına bağlanmak isteyen kullanıcılarının doğrulamasını veritabanında bulunan bilgiyi kullanarak yapabilir. Oracle veritabanı, kullanıcı doğrulaması için, kullanıcı bir bağlantı isteğinde bulunduğunda kullanıcıdan kullanıcı adı ve şifre ister. Kullanıcının şifreyi yanlış girmesi durumunda veritabanının yetkisiz kullanımını engeller, bağlantısını reddeder. Oracle, kullanıcı şifresinin, yetkisiz kullanımı ve değişiklik yapılmasını engellemek için şifrelenmiş olarak veri sözlüğünde saklar. Kullanıcı şifresini herhangi bir zaman değiştirebilir.

Oracle veritabanı doğrulamasında şifre güvenliği, parola şifreleme, hesap kilitleme, şifre ömrü tamamlaması ve sona ermesi, şifre karmaşıklığı gibi güvenlik özelliklerini sağlamaktadır.

- Parola şifreleme: Şifre koruma gizli ve güvenilir olmalıdır. Oracle her zaman şifreleri ağ üzerinden göndermeden önce AES kullanarak şifreler. [40]
- Hesap kilitleme: Oracle, kullanıcının hesabını önceden belirlenmiş belli sayıda başarısız giriş denemesinden sonra kilitleyebilir. Hesabın belli bir zaman sonra

veya belli sayıda başarısız girişten sonra otomatik kilitlenmesi ayarlanabilir. Veritabanı yöneticileri hesabı elle de kilitleyebilir. Kilitlenen kullanıcı hesabı yine veritabanı yöneticileri tarafından açılabilir.

- Şifre ömrü tamamlanması ve sona ermesi: Veritabanı yöneticileri, şifreler için şifreye özel bir yaşam ömrü belirleyebilir ve bu sona erdiğinde yeniden giriş yapabilmek için kullanıcı şifresini değiştirmek zorunda kalır. Belli bir peryod da söz konusu süre dolduğunda her giriş işleminde şifrenin değiştirilmesi gerektiği kullanıcıya uyarı mesajı şeklinde her defasında hatırlatılır. O peryod süresinde şifre yine de değiştirilmez ise hesap kilitlenir. Veritabanı yöneticisinin müdahelesine kadar artık girişe izin verilmez.

  Veritabanı yöneticileri şifreyi sonlandırabilir ve kullanıcı hesabı şifresinin sona ermesini sağlayabilir. Kullanıcı veritabanına girmeden önce, kullanıcı veya veritabanı yöneticisi şifre değişikliğini yapmak zorundadır.

  Şifre geçmişi kontrol edilerek yeni şifrenin belli bir zaman öncesine kadar tekrar kullanılmamış olması veya belli bir sayı kadar öncesine kadar kullanılmamış olması gerekir.
- Şifre Karmaşıklığı: Şifre karmaşıklığı kontrolü her bir şifre için yapılmalı ve davetsiz misafirlere karşı yeterince koruma sağlayıp sağlamayacağı kontrol edilmelidir. Şifrelerin tahmin edilerek sisteme girişlerin engellenmesi sağlanmalıdır.

  Oracle varsayılan olarak şifre karmaşıklığı için her bir şifrenin aşağıdaki kriterleri taşımasını ister:

  - Minimum 4 karakter uzunluğunda olmalı,
  - Kullanıcı ID'siyle aynı olmamalı,
  - En az bir karakter harf, bir sayısal karakter ve bir karakter de noktalama işareti içermeli,
  - İlk bakışta akla gelecek basit kelimeler kullanılmamalı. Örneğin merhaba, hesap, veritabanı, kullanıcı vb.
  - Önceki şifreden en az 3 karakter farklı olmalı.

4.2.1.4. Çok katmanlı kullanıcı doğrulama

Çok katmanlı bir ortamda, Oracle, orta katman uygulamaların güvenliğini hakları sınırlandırmayla, kimlikleri tüm katmanlardan saklamayla ve istemcilerin davranış ve hareketlerinin hesap denetimini yaparak kontrol eder. Uygulamalarda iş izleme işlemi, orta katmana bağlanan istemcilerin kimliğinin saklanması gibi işlemlerin yapıldığı bir orta katman kullanılır. Orta katmanın bir avantajı bağlantı havuzunun olmasıdır, bu havuz bir veri sunucusuna birçok kullanıcının her biri için ayrı bir bağlantı gerektirmeksizin bağlanmasını sağlar. Bu tür ortamlarda, bağlantılar çok hızlı kurulabilir ve kesilebilir. [31]

Çok katmanlı bir ortamda bir uygulama sunucusu (Application server) istemcilerin doğrulamasını gerçekleştirebilir. Doğrulama işlemi aşağıdaki gibi gerçekleşir:

- Kullanıcı uygulama sunucusuna bir şifre veya sertifika yoluyla (SSL kullanarak) doğrulanma isteğinde bulunur.
- Uygulama sunucusu kendisi ve veritabanı sunucusu için doğrulamayı gerçekleştirir. Uygulama sunucusu bu doğrulama işlemi için, Oracle internet dizinini kullanır. Oracle internet dizini cüzdandan kullanıcı bilgisini alıp uygulama sunucusuna gönderir. Uygulama sunucusu da kullanıcı kimliğini veritabanı sunucusundaki cüzdandaki kullanıcı bilgisiyle karşılaştırarak kontrol eder ve bu kullanıcıya haklarını verir. [26]
- Veritabanı sunucusu da, uygulama sunucusunu doğrular ve mevcut kullanıcıyı onaylar.

Bu ortamlar için, Oracle veritabanı yöneticileri Oracle Call Interface(OCI) ile basit oturumlar oluşturabilir, her bir kullanıcı için veritabanı şifresiyle doğrulamayı gerçekleştirebilir. Orta katman yoluyla her bir kullanıcı için farklı bir veritabanı bağlantısı olmaksızın gerçek kullanıcının kimliği saklanabilir.

4.2.1.5. Veritabanı yöneticilerini doğrulama

Veritabanı yöneticileri normal kullanıcıların yapamayacağı birçok operasyonu (veritabanının başlatılması veya sonlandırılması gibi) gerçekleştirebilir. Oracle veritabanı yöneticileri için daha güvenli bir doğrulama imkânı sağlar.

Veritabanı yöneticileri doğrulaması için işletim sistemi doğrulaması veya şifre dosyaları arasında seçim yapabilir.

Bir veritabanı yöneticisi için işletim sistemi hesap denetimi, kullanıcı adlarının yetki verilmiş bir gruba dâhil edilerek gerçekleştirilir.

Diğer bir yöntem olan Oracle şifre dosyası (Password file) ise, içerisinde SYSDBA ve SYSOPER yetkisine sahip kullanıcıların bilgilerinin bulunduğu dosyadır. Yani veritabanı, şifre dosyalarını kullanarak veritabanı kullanıcılarının adlarını bir dosyada tutar, bu kullanıcılara SYSDBA ve SYSOPER hakları verilmiş olup aşağıdaki operasyonları gerçekleştirebilirler: [40]

- SYSOPER'a sahip veritabanı yöneticileri STARTUP, SHUTDOWN, ALTER DATABASE OPEN/MOUNT, ALTER DATABASE BACKUP, ARCHIVE LOG, RECOVER ve RESTRICTED SESSION yetkisi.
- SYSDBA, ADMIN OPTION ile tüm sistem hakları ve SYSOPER sistem yetkisi.

Şifre dosyası içinde olan kullanıcıların bilgisine aşağıdaki komutla ulaşılabilir. [40]

```
SELECT * FROM v$pwfile_users;
```

4.2.2. Yetkilendirme (Authorization)

Yetkilendirme iki şekilde gerçekleşmektedir:

- Yalnızca belirli kullanıcıların veriye erişimine, veriyi işlemesine veya değiştirmesine izin vermek yoluyla,

- Kullanıcıların erişim ve hareketlerine çeşitli sınırlamalar uygulama yoluyla. Sınırlamalar kullanıcılar üzerinde olup nesneler için uygulanabilir veya kaldırılabilir. Örneğin şemalar, tablolar, kayıtlar, kaynaklar ve süreler (CPU kullanım, bağlantı süresi vb.) için olabilir.

4.2.2.1. Kullanıcı kaynak sınırlandırmaları ve profiller

Kullanıcılar için sistem kaynakları sınırlandırılabilir. Bu şekilde, CPU kullanımı gibi sistem kaynaklarının kontrolsüz tüketilmesi önlenebilir. Bu işlem sistem kaynaklarının çok pahalı olduğu çok kullanıcılı sistem de çok yararlıdır. Kaynakların aşırı tüketimi veritabanı kullanıcısının zarar görmesine neden olabilir.

Kullanıcının kaynak yönetimi sınırlandırma işlemi kendisine veya profiline yapılır. Güvenlik yöneticisi, profil için kaynak sınırlandırmaları belirleyebilir ve sınırlamaları kaldırabilir. Kaynakları sınırlandırılmış kullanıcıların oturumlar açması, performansta bir parça azalmaya neden olabilir. Çünkü Oracle, kullanıcı veritabanına bağlandığı zaman kullanıcı için tüm kaynakları sınırlama çerçevesinde yükler.

4.2.2.2. Sistem kaynakları ve sınırlamalar

Oracle, sistem kaynaklarını sınırlandırabilir. Bu işlem şu kaynakları kapsayabilir:

- Oturum Seviyesi: Kullanıcı bir veritabanına bağlandığı zaman bir oturum yaratılır. Her bir oturum, CPU ve Oracle'ın çalıştığı bilgisayardaki belleği tüketir. Oturum seviyesinde çeşitli kaynaklar sınırlandırılabilir. Eğer kullanıcı kendisine ayrılan kaynak kullanım sınırını aşarsa, Oracle, oturum için sınıra ulaşıldığının mesajını verir ve aktif durumu sonlandırır.
- Çağrı Seviyesi: Bir SQL cümlesi çalıştığı zaman, işlem için çeşitli aşamalar gerçekleşir. İşlem anında, veritabanının farklı bölümlerinde, farklı yürütme aşamaları meydana gelebilir. Aşırı şekilde sistemi kullanan çağrılardan herhangi birini önlemek için, Oracle çağrı seviyesinde kaynakları sınırlayabilir. Eğer kullanıcı bir çağrı kaynak sınırını aşarsa, Oracle o durumdaki işlemi sonlandırır ve bir hata yayımlar. Aktif işlemde herhangi bir kayıp meydana gelmez.

- CPU: Oracle veritabanına SQL cümleler ve diğer tip çağrılar yapıldığında, bu çağrıları işlemek için CPU kullanımı gerekir. Ortalama çağrılar küçük bir CPU kullanımına ihtiyaç duyar. Başka bir SQL cümlesi ise büyük miktarda bir veri içerebilir veya kontrolsüz bir sorgu, potansiyel olarak CPU kullanımını büyük bir miktarda tüketebilir. Bu da diğer işlemler için uygun CPU kaynağının azalmasına yol açar. CPU'nun kontrolsüz kullanımını önlemek için, oturum esnasında, Oracle, CPU kullanımını sınırlandırabilir. Bunun için CPU kullanım planlanması yapılması gerekir. [54]
- Mantıksal Okumalar: Mantıksal veri bloğu disk ve belleğin her ikisinden okunan veri bloklarını içerir. Oracle her bir çağrı ve oturumun mantıksal veri bloğu okumasında kaynakların aşırı I/O işlemi yapmasını önlemek için sınırlandırmalar yapabilmektedir.
- Diğer Kaynaklar: Oracle oturum seviyesinde çeşitli kaynaklar içinde sınırlandırmalar sağlayabilir:
  - Her bir kullanıcı için aktif oturum sayısının limiti belirlenebilir.
  - Bir oturum için boş zaman (idle time) limiti belirlenebilir.
  - Her bir oturum için bağlantı zamanı süresi limiti belirlenebilir. Eğer bir oturumun süresi, üst zaman sınırını aşıyorsa, aktif işlem geri alınır, oturum düşer ve oturumun kaynakları sisteme geri verilir. Bu dakika limiti sayısal olarak belirlenebilir.

4.2.2.3. Profiller

Profil, Oracle veritabanında geçerli bir kullanıcıya tahsis edilmiş belirli sistem kaynaklarının belli sınırlar dâhilinde gruplanıp adlandırılmasıdır. Profiller kaynakların limitler çerçevesinde kolayca yönetilmesini sağlar. Profillerle şifre politikası da yönetilebilir.

Veritabanı güvenlik politikası için kaynakların sınırlandırılması bir gereksinimdir. Profilleri kullanmak için, veritabanındaki kullanıcı tiplerini ilişkilendirip sınıflandırmak gerekir. Roller yalnızca ilişkili kullanıcıların haklarını yönetmek için kullanılırken, profiller ilişkili kullanıcıların kaynak sınırlamalarını yönetmek için kullanılır.

Profilleri oluşturmadan önce onlarla ilişkili kaynakların limitleri ayarlanmalı, her bir limit için verilecek değerler belirlenmelidir. Bu değerler standart bir kullanıcının yaptığı işlerin sayısı temel alarak belirlenebilir. Genellikle en iyi yöntem, kullanıcının geçmişini inceleyerek belirlemektir. İstatistikler ve kullanılan limitler Oracle Enterprise Manager'ın Monitor aracıyla veya SQL PLUS kullanarak izlenebilir. [40]

4.2.2.4. Haklar

Hak, bir SQL cümlesini çalıştırmak veya bir başka kullanıcının nesnesine erişim için verilen ayrıcalıktır. Kullanıcılara, işleriyle ilgili görevlerini yerine getirebilmeleri için haklar verilir. Sadece gerekli olan kişilere, gerekli olan haklar verilmelidir. Kullanıcıya gereksiz aşırı hak verilmesi güvenliği tehlikeye düşürebilir. Bir kullanıcıya 2 şekilde hak verilebilir:

- Hak, direkt olarak kullanıcının kendisine verilebilir. Örneğin, SCOTT kullanıcısına personel tablosuna kayıt ekleme hakkı verilmesi gibi.
- Bir rol için hak verebilir. Böylece, bu role sahip olan bir veya birden fazla kullanıcıya bu hak verilmiş olur. Örneğin, personel tablosundan kayıt seçme, update ve kayıt silme hakları bir role verildiğinde, bu rolü kullanabilen tüm kullanıcılar da bu hakları kullanabilir.

Hakların kullanılmasında roller daha kolay ve etkili kullanım imkânı sağlar. Bu nedenle hakları, kullanıcılara vermek yerine rollere verilmesi daha iyi bir yöntemdir.

Haklar 2'ye ayırılabilir:

- Sistem hakları,
- Nesne hakları.

4.2.2.4.1. Sistem hakları

Sistem hakkı, veritabanları üzerindeki işleri yapmak için kullanılan çok güçlü haklardır. Veritabanı şemalarındaki işlemleri gerçekleştirmek için de kullanılabilir. Örneğin, bir veritabanındaki tablespace'leri yaratmak veya herhangi bir tablodaki kayıtları silmek için sistem hakları kullanılır. 100'ün üzerinde sistem hakkı vardır. ALTER DATABASE, ALTER PROFILE, CREATE ROLE, DELETE ANY TABLE, DROP PROFILE, EXECUTE ANY PROCEDURE, INSERT ANY TABLE, UPDATE ANY TABLE, CREATE VIEW sistem haklarından bazılarıdır. [9]

4.2.2.4.2. Nesne hakları

Nesne hakkı, bir SQL cümlesini çalıştırmak veya başka bir kullanıcının nesnesine erişebilmek için verilen haktır. Şema nesnesi hakları, tablo hakları, görüntü (view) hakları, procedure hakları ve tip hakları gibi nesne hakları vardır. Bazı nesne hakları Tablo 4.2.'de yer almaktadır.

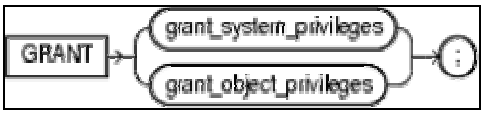
Tablo 4.2. Oracle veritabanında nesne hakları.

Nesne hakları	Table	View	Sequence	Procedures, Functions, Packages	User - defined Type
ALTER	X		X		
DELETE	X	X			
EXECUTE				X	X
INDEX	X				
INSERT	X	X			
READ					
REFERENCES	X	X			
SELECT	X	X	X		
UPDATE	X	X			

Oracle veritabanında hak verme işlemleri şu şekilde yapılmaktadır:

Veri kontrol dili cümleleri (Data Control Language (DCL)); tablolar, görüntüler, sequences, sinonimler ve yordamlarla (prosedür) ilgili hakları kullanıcılara ve rollere vermek/geri almak için kullanılır. Bunlar, kullanıcılara ve rollere hakları vermek için kullanılan GRANT ve verilmiş hakları geri almak için kullanılan REVOKE' tır.

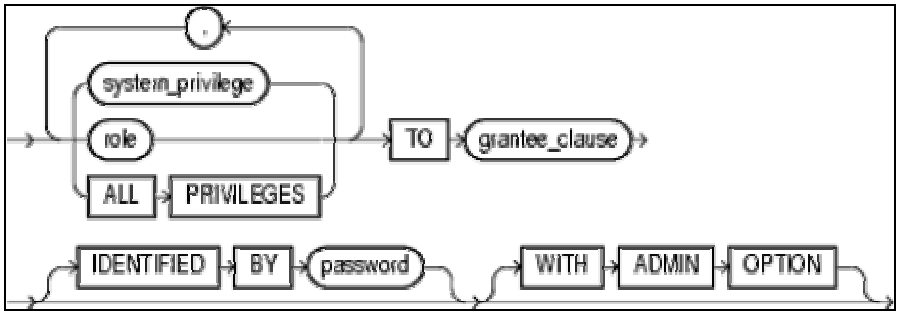
grant::=



Şekil 4.8: Oracle veritabanında GRANT deyiminin kullanılışı.

GRANT { grant_system_privileges | grant_object_privileges };

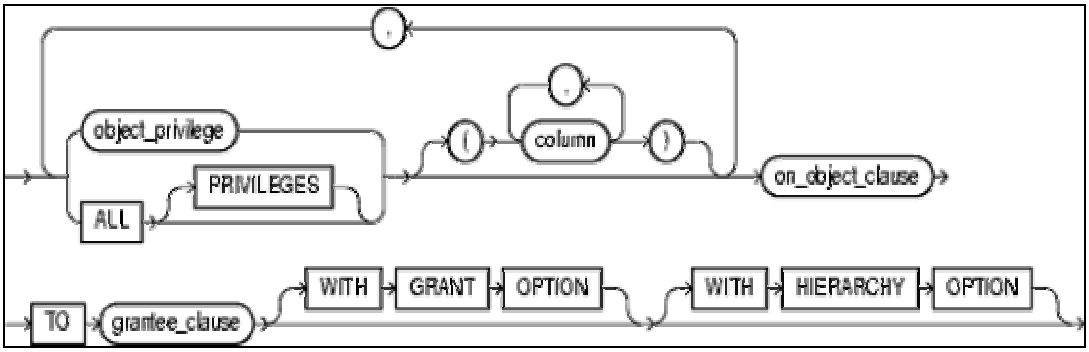
grant_system_privileges**::=**



Şekil 4.9: Oracle veritabanında GRANT deyiminin sistem hakları için kullanılışı.

{ system_privilege | role | ALL PRIVILEGES }
[, { system_privilege | role | ALL PRIVILEGES }]...
TO grantee_clause [IDENTIFIED BY sifre] [WITH ADMIN OPTION]

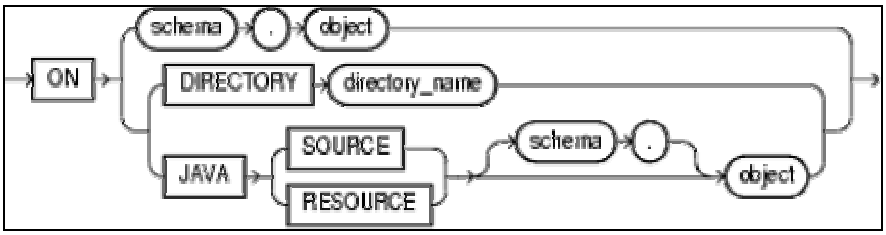
grant_object_priviges::=



Şekil 4.10: Oracle veritabanında GRANT deyiminin nesne hakları için kullanılışı.

{ object_privilege | ALL [PRIVILEGES] } [(column [, column]...)]
[, { object_privilege | ALL [PRIVILEGES] } [(column [, column]...)]]...
on_object_clause
TO grantee_clause
[WITH GRANT OPTION] [WITH HIERARCHY OPTION]

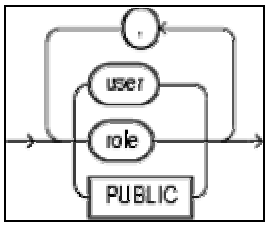
on_object_clause::=



Şekil 4.11: Oracle veritabanında GRANT deyiminin nesne üzerinde kullanılışı.

{ schema.object
 | { DIRECTORY directory_name
 | JAVA { SOURCE | RESOURCE } [schema .] object
  }
}

grantee_clause::=



Şekil 4.12: Oracle veritabanında GRANT deyiminin kullanıcı veya rol için kullanılışı.

{ user | role | PUBLIC } [, { user | role | PUBLIC }]...

Örnek hak verme sorgu ifadeleri:

- GRANT SELECT, UPDATE, INSERT ON musteriTablosu TO Muhsin;
- GRANT SELECT ON musteriTablosu TO Muhsin WITH GRANT OPTION;
- GRANT CREATE PROFILE, ALTER PROFILE, DROP PROFILE, CREATE ROLE, DROP ANY ROLE, GRANT ANY ROLE, AUDIT ANY, AUDIT SYSTEM, CREATE USER, BECOME USER, ALTER USER, DROP USER TO security_admin WITH ADMIN OPTION;

Bir hakkın veya rolün ADMIN OPTION özelliğiyle verilmesi şu yeteneklere sahip olması anlamına gelmektedir:

- ADMIN OPTION'le rolü veya hakkı alan kullanıcı, herhangi bir kullanıcı veya role bu hakkı veya rolü verebilir. Aynı zamanda da bu role sahip herhangi bir kullanıcı veya rolden bu hakkı geri alabilir.
- Hak veya rolü ADMIN OPTION'le verebilir.
- Bir rolü değiştirebilir veya düşürebilir.

4.2.2.5. Roller

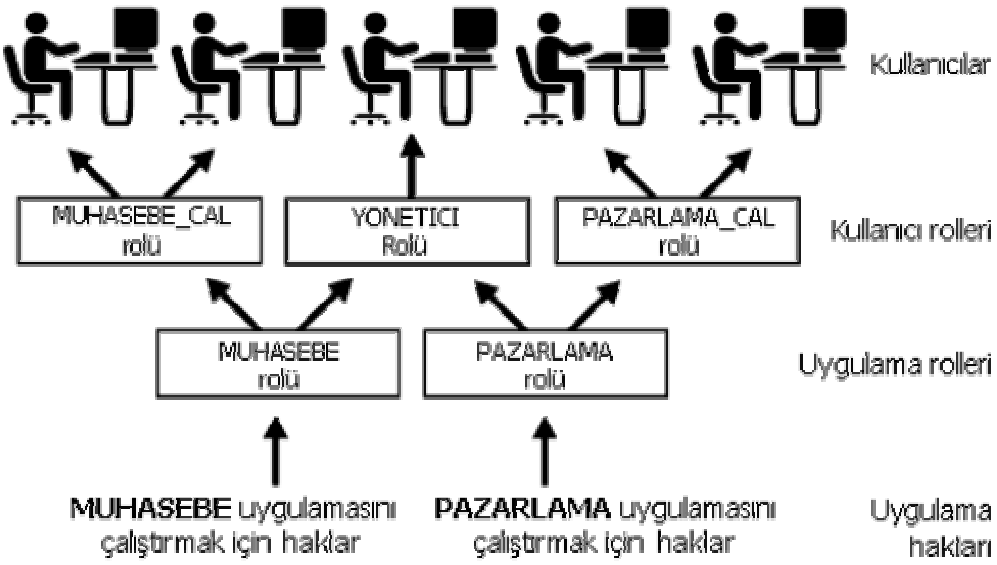
Hakların yönetimi ve kontrolü roller kullanılarak daha kolay yapılır. İlişkili hakların gruplanmasıyla oluşan roller, bir gruba, kullanıcılara veya diğer rollere verilebilir. Bir veritabanında her rol tek olmalıdır. Bütün kullanıcı adlarından ve rollerden farklı isimlendirilmelidir. Şema nesnelerine benzemeyen roller, herhangi bir şema içinde değildirler. Bundan dolayı kullanıcı için hiçbir etkisi kalmamış rol silinebilir. Roller son kullanıcılı sistemin ve şema nesnesi haklarının yönetimini kolaylaştırır. Oracle veritabanında rollerin yararları şunlardır:

- Hak yönetimini azaltma: Kullanıcılara aynı hakları vermektense, ilişkili kullanıcılardan oluşacak gruba bu haklardan oluşacak bir rolü vermek gerekir. Böylece her bir üyeye hak vermek yerine gruba rol vermek yeterli olacaktır.
- Dinamik hak yönetimi: Eğer grubun haklarının değişmesi gerekirse yalnızca rolün haklarını değiştirmek yeterli olacaktır. Grubun tüm kullanıcılarına verilmiş olan haklar otomatik olarak rol tarafından değiştirilmiş olup değişiklikler hemen yansıyacaktır. [5]
- Hakların seçmeli kullanılabilirliği: Roller herhangi bir kullanıcı için aktif veya pasif yapılabilir.
- Uygulama için özel güvenlik sağlama: Rol bir şifreyle korunabilir. Uygulamalar yoluyla şifresini bilen kullanıcılar o rolün haklarını kullanabilir.

Bir rol genel olarak 2 amaç için kullanılabilir. Bir veritabanı uygulamasının haklarını yönetmek için veya bir kullanıcı grubunun haklarını yönetmek için.

Uygulama rolleri: Veritabanı yöneticileri veritabanı uygulamaları için roller yaratırlar. Uygulamayı çalıştırmak için gerekli olan tüm haklar güvenli bir uygulama rolüne verilmelidir. Uygulama rolü, diğer rollere ve kullanıcılara verilebilir. Bir uygulama, her biri farklı haklardan oluşan daha fazla ya da daha az veri erişim özelliği olan değişik rollere sahip olabilir.

Kullanıcı rolleri: Veritabanı kullanıcı grubunun gereksinim duyduğu haklardan oluşan rollerdir. Kullanıcı hakları, uygulama rolleri ve kullanıcı rolü haklarıyla yönetilebilir.



Şekil 4.13: Oracle veritabanında örnek rol dağılımı.

YONETICI rolünü Muhsin kullanıcısına ADMIN OPTION'lı olarak veren SQL cümlesi örneği:

GRANT YONETICI TO muhsin WITH ADMIN OPTION;

4.2.3. Sanal kişisel veri tabanı (Virtual Private Database)

Kullanıcıların kritik verilere direk erişimini sağlayan, sunucu merkezli kayıt bazında güvenlik mekanizmasıdır. Sanal kişisel veritabanı özelliği, veritabanındaki kayıtların tutulduğu tablolara erişimin kullanıcı bazında ayarlanabilmesini sağlamaktadır. Kullanıcılar veri tabanına ne şekilde erişirlerse erişsinler bir tabloda sadece kendilerinin görmelerine izin verilmiş olan kayıtlara ulaşabilmektedirler. Bu filtreleme işlemi, veritabanı yöneticisi tarafından veritabanı düzeyinde tanımlanmaktadır. Ayrıca uygulamayı kullanmak yerine bir başka yöntemle, örneğin bir raporlama aracıyla, veri tabanına ulaşan kullanıcının bu güvenlik mekanizmasını aşması engellenmiş olmaktadır.

Sanal kişisel veri tabanı özelliği kapsamında, tablolara eklenen filtreler dinamik olarak kullanıcı bazında ayarlanabilmekte, ayrıca aynı kullanıcının, sisteme bağlandığı gün, saat ya da son kullanıcı terminaline bağlı olarak farklı kayıtlara erişmesinin sağlanması da mümkün olmaktadır. Böylelikle sözgelimi, kullanıcının hafta içi iş saatlerinde ulaşabildiği veri ile hafta sonu evinden sisteme bağlandığında ulaşabileceği veri farklı olabilmektedir. [40]

SQL sorgularına otomatik olarak bir doğrulama koşulu eklenerek fine-grained erişim kontrolü sağlanmış olur.Örneğin,

"SELECT * FROM muhasebe" sorgusu otomatik olarak,
"SELECT * FROM muhasebe WHERE rollno = userID()"

şekline dönüşür. Ve kullanıcı sadece erişim yetkisi olan kayıtları görebilir.

Sanal kişisel veri tabanı özelliğini kullanmak için WHERE sözcüğüne eklenmek üzere bir fonksiyon oluşturulur ve bu fonksiyon yardımıyla politika, istenen nesneler için uygulanır. Politikayı tabloya veya görüntüye eklemek için DBMS_RLS.ADD_POLICY procedure kullanılır. Bu fonksiyonlar genelde güvenlik yöneticilieri tarafından hazırlanır. Bir politika ekleme örneği;

```
BEGIN
DBMS_RLS.ADD_POLICY(
object_schema => 'hr',
object_name => 'employees',
policy_name => 'secure_update',
policy_function => 'check_updates',
...
```

4.2.4. Veri şifreleme

Oracle veritabanı, kullanıcı doğrulama, yetkilendirme ve hesap denetimi şeklinde güvenlik sağlar. Bu güvenlik mekanizmaları etkili bir şekilde veritabanındaki veriyi korumasına rağmen verinin saklandığı ortamdaki işletim sistemi dosyaları için erişimi önlemez. Bunun için verinin şifrelenmesi önem kazanmaktadır. Oracle, saydam veri şifreleme ve veri şifreleme API'leri kullanarak veri şifreleme yapmaktadır. [42]

Saydam veri şifreleme, veritabanı sütunlarındaki hassas verinin şifrelenmesini sağlar. Saydam veri şifreleme bir anahtar tabanlı kontrol sistemi olup gizli tutulan bir anahtarla şifreleme yapılarak gerçekleştirilir. Verilen bir tablodaki şifrelenmiş sütunların sayısı ne olursa olsun her bir veritabanı tablosu için yalnızca bir anahtar olabilir. Her bir tablonun sütun şifreleme anahtarı, veritabanı sunucusunun master key'iyle şifrelenmiştir. Anahtarlar veritabanında saklanmaz. Harici güvenlik modülünün bir parçası olan Oracle cüzdanında saklanırlar. Harici bir güvenlik modülü kullanmanın en önemli özelliği güvenliği artırmasıdır.

Ayrıca Oracle, tablespace şifrelemeye de imkân tanımaktadır. Tablespace içindeki tüm nesneler şifrelenebilir. Tablespace şifreleme, saydam veri şifreleme mimarisini kullanır. Şifreleme master key'i Oracle cüzdanında saklanır.

4.2.4.1. Saydam Veri Şifreleme (Transparent Data Encryption)

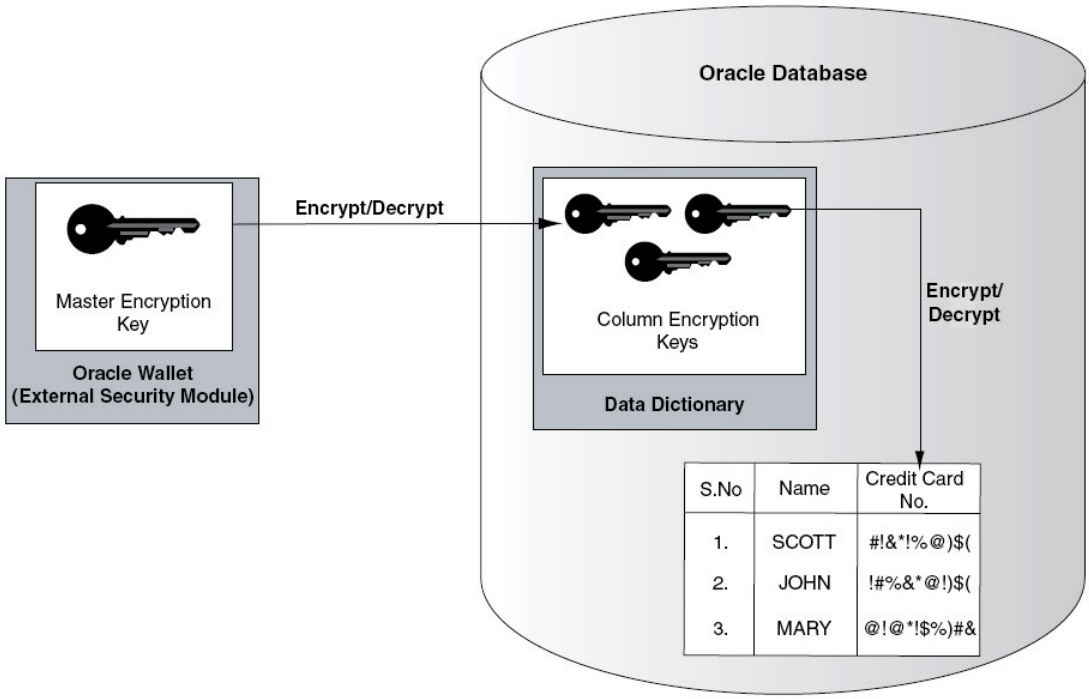
Oracle veritabanı, veri dosyalarını korumak için saydam veri şifreleme özelliğine sahiptir. Saydam veri şifreleme veri dosyalarında saklanan hassas verileri şifreler. Saydam veri şifreleme, yetkisiz deşifrelemeyi önlemek için de, şifreleme anahtarlarını veritabanı dışındaki bir güvenlik modülünde saklar. Bu şifreleme, bir medya ortamındaki veya herhangibir yerde saklanan veri çalınsa bile koruma sağlar. Saydam veri şifrelemenin yararları şunlardır:

- Hassas veriyi güvenli bir ortamda veya veri dosyasında saklayabilme imkânı sağlar.
- Veriyi deşifrelemek için tetikleyiciler (trigger) veya görüntüler (view) yaratmak gerekmez. Veri, kullanıcı tarafından tablolardan deşifre edilebilir.
- Veritabanı kullanıcılarının verinin saklanış şekli hakkında bilgileri olmalarına gerek olmadan deşifreleme işlemini yapabildiklerinden, onlar için herhangi bir işlem gerektirmemektedir.
- Uygulamalar şifrelenmiş veriler için değiştirilmek zorunda kalmazlar. Veri şifreleme/deşifreleme veritabanı tarafından yapılmaktadır.

Not: Saydam veri şifrelemeyi foreign key kısıtlamaları olan sütunların şifrelenmesi ve deşifrelenmesinde kullanılmaz.

Saydam veri şifreleme nasıl çalışır:

Saydam veri şifreleme anahtar tabanlı bir kontrol sistemidir. Şifrelenmiş veri ele geçirilse bile, yetkili bir deşifreleme olana kadar o veri anlaşılamaz. Bir tabloda şifrelenmiş sütun sayısı ne kadar olursa olsun tek bir anahtar kullanılır. Bu anahtar, sütun şifreleme anahtarı olarak adlandırılır. Tüm tablolar için kullanılan sütun şifreleme anahtarları, veritabanı sunucusu master şifreleme anahtarıyla şifrelenmiş olarak veritabanında bir sözlük tablosunda saklanır. Anahtarlar şifrelenmemiş olarak saklanmazlar. Şekilde master şifreleme anahtarının veritabanı dışındaki harici bir modülde saklandığı görülmektedir. Bu anahtara yalnızca güvenlik yöneticisi tarafından erişilebilir. Harici güvenlik modülü için, Oracle, bir Oracle cüzdanı kullanır. Master şifreleme anahtarının bu yolla yetkisiz kullanımı engellenip saklanması sağlanmış olur. [47]



Şekil 4.14: Saydam veri şifreleme.

Harici güvenlik modülü kullanmak alışılmış program fonksiyonlarını, şifreleme işlemlerinden ayırmayı sağlar ve veritabanı yöneticileriyle güvenlik yöneticileri arasındaki görevlerin ayrılmasına da yardımcı olur. Güvenlik bu şekilde daha iyi sağlanır. Çünkü cüzdanın şifresi veritabanı yöneticisi tarafından dâhi bilinmeyebilir. Gerekirse, şifreyi bilen güvenlik yöneticisi bu şifreyi veritabanı yöneticisine verebilir.

Saydam veri şifrelemeyi sağlayabilmek için, ALTER SYSTEM hakkı ve geçerli bir Oracle cüzdanı şifresine sahip olma zorunluluğu vardır. Eğer bir Oracle cüzdanı yoksa SQL komutuyla ve bir şifre belirleyerek yeni bir tane oluşturulabilir. Yeni master key oluşturmak ve saydam veri şifrelemeye başlamak için şu komut kullanılabilir:

ALTER SYSTEM SET [ENCRYPTION] KEY IDENTIFIED BY "sifre"

Bu komut, sunucu tarafından her bir tablo için sütun şifrelemede kullanılmak üzere veritabanı master şifreleme anahtarını üretir. Veritabanındaki tabloların sütunları sunucu tarafından master key ayarlanana kadar şifrelenemez. Master şifreleme anahtarı veritabanı instance'ı kapanana kadar veritabanı için erişilebilir durumdadır. Veritabanı başladıktan sonra, master şifreleme anahtarını yüklemek için şu komut kullanılır:

ALTER SYSTEM SET [ENCRYPTION] WALLET OPEN IDENTIFIED BY "sifre"

Şifreli sütunlu yeni bir tablo yaratmak için şu komut kullanılır:

CREATE TABLE table_name (column_name column_type ENCRYPT,....);

Eğer var olan bir tablonun bir sütununu şifrelemek istiyorsak şu komut kullanılır:

ALTER TABLE table_name MODIFY (column_name column_type ENCRYPT,...);

Veritabanındaki tüm şifrelenmiş sütunların erişimi engellenmek isteniyorsa şu komut kullanılabilir:

ALTER SYSTEM SET [ENCRYPTION] WALLET CLOSE;

Bu komut cüzdandaki master anahtara erişimi engeller ve bu da şifrelenmiş sütunlardaki veriye erişimin engellenmesi anlamına gelir. Cüzdanı tekrar açmak

gerekirse, ALTER SYSTEM SET WALLET OPEN IDENTIFIED BY “Password” komutu kullanılarak master şifreleme anahtarına erişim sağlanır.

Saydam veri şifreleme Oracle 10g versiyon 2 ve sonraki sürümlerde bulunmaktadır. Harici güvenlik modülü anahtarları Oracle cüzdanında saklar ve nerde olduğu sqlnet.ora konfigürasyon dosyasında belirtilir. sqlnet.ora dosyasında, Oracle cüzdanının yeri için WALLET_LOCATION parametresi kullanılır. Harici güvenlik modülü cüzdanların otomatik açılabilmesine olanak sağlayabilir.

Veritabanı sütunları şifreleme ya da deşifreleme işleminden önce master şifreleme anahtarı belleğe yüklenmek zorundadır. Cüzdanın açılmasıyla, veritabanı master şifreleme anahtarına erişilebilir. Cüzdanı açmak için şu komut kullanılabilir:

```
ALTER SYSTEM SET [ENCRYPTION] WALLET OPEN IDENTIFIED BY “sifre”
```

Burada ki sifre, cüzdanı açma şifresidir. Cüzdan açıldığında veritabanı instance’ı kapanıncaya veya ALTER SYSTEM SET [ENCRYPTION] WALLET CLOSE komutuyla kapatılana kadar açık kalır. Şemanın ALTER SYSTEM hakkı yoksa veya şifre yanlış girilirse cüzdanı açma komutu hataya yol açar.

Şifrelenmiş sütunlara sahip tablolar yaratmak için, CREATE TABLE SQL komutuyla tablo oluşturma esnasında sütunlarda ENCRYPT SQL deyimi kullanılır. Saydam veri şifreleme, default olarak, 192 bit anahtarlı AES(AES192) şifreleme algoritmasını kullanır. Eğer bir tablo sütununu şifrelemek için özel olarak bir algoritma kullanılmazsa AES192 algoritması uygulanır.

Şifrelenmiş sütuna sahip yeni bir tablo oluşturma örneği:

```
CREATE TABLE employee ( first_name VARCHAR2(128), last_name VARCHAR2(128),
                        empID NUMBER, salary NUMBER(6) ENCRYPT );
```

Saydam veri şifreleme default olarak açık metni şifrelemeden önce bu metine salt adı verilen rasgele bir string ekler. Bunun amacı şifrelenmiş verinin güvenliğini arttırmaktır. Yani, saldırılara ve saldırganlara karşı veriyi güçlendirmek için kullanılır. Eğer şifrelenmiş sütun index olacaksa, o sütun için NO SALT kullanmak gerekir. [40]

Saydam veri şifrelemede AES192 default algoritması dışında 3DES168, AES128 ve AES256 algoritmaları da kullanılabilir.

```
CREATE TABLE employee ( first_name VARCHAR2(128), last_name VARCHAR2(128),
                        empID NUMBER ENCRYPT NO SALT, salary
                        NUMBER(6) ENCRYPT USING '3DES168' );
```

Bu örnekte ENCRYPT deyimiyle NO SALT ve şifreleme algoritması olarak 3DES168 kullanılmıştır.

Mevcut bir tabloya şifrelenmiş sütun eklemek için, var olan bir sütunu şifreleme veya deşifreleme için ADD veya MODIFY içeren ALTER TABLE SQL komutu kullanılır.

Var olan bir tabloya şifrelenmiş bir sütun eklemek için ALTER TABLE ADD komutu kullanılır, yeni sütun ENCRYPT deyimiyle tanımlanır. Örnek SQL kodu şöyle olabilir:

```
ALTER TABLE employee ADD (ssn VARCHAR2(11) ENCRYPT);
```

ssn sütunu AES192 algoritmasıyla şifrelenmiştir. Default olarak, Salt değeri veriye eklenmiştir.

Şifrelenmemiş bir sütunu şifrelemek için ENCRYPT ve ALTER TABLE MODIFY komutu kullanılır. Örnek SQL kodu:

```
ALTER TABLE employee MODIFY (first_name ENCRYPT);
```

Burada da, first_name sütunu AES192 algoritmasıyla şifrelenmiştir. Default olarak, salt değeri veriye eklenmiştir.

Performans veya başka bir nedenle şifrelemeyi etkisiz hale getirmek istenebilir. Sütun şifrelemesini etkisiz hale getirmek için DECRYPT deyimiyle ALTER TABLE MODIFY komutu kullanılır. Bir örnekle açıklamak gerekirse:

```
ALTER TABLE employee MODIFY (first_name DECRYPT);
```

Şifreli sütun üzerinde index yaratmak için se, CREATE INDEX komutu kullanılır. Sütun salt değeri eklenmeden şifrelenmelidir. Örnek SQL kodu:

```
CREATE TABLE employee ( first_name VARCHAR2(128), last_name VARCHAR2(128),
                        empID NUMBER ENCRYPT NO SALT, salary
                        NUMBER(6) ENCRYPT USING '3DES168');

CREATE INDEX employee_idx on employee (empID);
```

Salt değerini şifrelenmiş bir sütuna eklemek veya kaldırmak için örnek SQL kodları:
Şifreli bir sütuna Salt değeri eklemek:

```
ALTER TABLE employee MODIFY (first_name ENCRYPT SALT);
```

Şifreli bir sütundan Salt değerini kaldırmak:

```
ALTER TABLE employee MODIFY (first_name ENCRYPT NO SALT);
```

Her tablo, sütunlarını şifrelemek için bir sütun şifreleme anahtarına sahip olabilir. ALTER TABLE komutu kullanılarak sütun şifreleme anahtarı yeniden üretilebilir. Böyle bir durumda, cüzdanın yedeğini aldıktan sonra değişiklikleri yapmak daha doğru bir hareket olacaktır.

Şifreli sütunlar içeren employee tablosu için, 3DES168 algoritmasını kullanarak yeni bir anahtar üretme işleminin sql kodu şu şekilde olur:

```
ALTER TABLE employee REKEY USING '3DES168';
```

4.2.4.2. Veri şifreleme API'lerini kullanarak şifreleme

Oracle, DBMS_CRYPTO ve DBMS_SQLHASH PL/SQL paketlerini kullanarak veri şifreleme yapabilmektedir.

4.2.4.2.1. DBMS_CRYPTO paketini kullanarak veri şifreleme

Oracle Veritabanı DBMS_CRYYPTO paketiyle, verinin şifrelenmesini ve deşifrelenmesini sağlar. DBMS_CRYPTO, Oracle 10g'den önceki versiyonlarda yer alan DBMS_OBFUSCATION_TOOLKIT yerine gelen şifreleme paketidir.

DBMS_CRYPTO paketi değişik yöntemlerle güvenlik sağlar. Bu paket, AES şifreleme algoritması dâhil değişik şifreleme ve özetleme algoritmaları kullanır. [40]

DBMS_CRYPTO paketi, RAW ve LOB nesneler dâhil temel Oracle veri tipleri üzerinde şifreleme ve deşifreleme yapabilir. Aşağıdaki şifreleme algoritmaları desteklenmektedir:

- DES, Triple DES
- AES
- SHA-1
- SHA-1 Mesaj Doğrulama Kodu (MAC)

Blok şifreleme de DBMS_CRYPTO tarafından sağlanmaktadır. Metin için, açık anahtar şifreleme standartını PKCS#5 içeren değişik doldurma seçenekleri seçilebilir ve zincirleme blok şifreleme modları kullanılabilir (CBC gibi).

DBMS_CRYPTO'nun önceki versiyonu olan DBMS_OBFUSCATION_TOOLKIT ile özellik karşılaştırması şöyledir.

Tablo 4.3: DBMS_CRYPTO ve DBMS_OBFUSCATION_TOOLKIT Karşılaştırması.

	DBMS_CRYPTO	**DBMS_OBFUSCATION_ TOOLKIT**
Şifreleme Algoritmaları	DES, 3DES, AES, RC4, 3DES_2KEY	DES, 3DES
Doldurma yöntemleri	PKCS5, zeroes	Yok
Blok şifreleme modları	CBC, CFB, ECB, OFB	CBC
Hash algoritmaları	SHA-1, SHA-1, MD4	MD5
MAC Algoritmaları	HMAC_MD5, HMAC_SH1	Yok
Veri tipleri	RAW, CLOB, BLOB	RAW, VARCHAR2

DBMS_CRYPTO paketi bütünlük sağlama işlemini daha iyi yapabilmektedir. Güvenli rasgele sayı üretimi ile de şifrelemede kullanılan anahtarların daha zor tahmin edilmesine katkı sağlaması nedeniyle güvenliği arttırmaktadır.

DBMS_CRYPTO paketiyle veri şifrelemede dikkat edilmesi gereken bazı durumlar vardır. Bunlar, şifreleme anahtarlarını üretme, anahtarları taşıma, anahtarları saklama ve anahtarları değiştirme olarak sayılabilir. [40]

Bir şifreleme anahtarı güvenilir bir şifreleme üretim yöntemiyle oluşturulmalıdır. Oracle veritabanı, DBMS_CRYPTO'nun RANDOMBYTES fonksiyonuyla güçlü bir rasgele sayı üretim imkânı sağlar.

Eğer şifreleme anahtarı uygulama tarafından veritabanına gönderilmişse şifrelenmesi gerekir. Aksi takdirde, başka birisi anahtarı ele geçirebilir. Oracle Advanced Security network şifreleme, şifreleme anahtarlarını koruma imkânı sağlar.[40]

4.2.4.2.2. Veri bütünlüğü için DBMS_SQLHASH paketinin kullanılması

DBMS_SQLHASH paketi hash algoritmaları kullanarak veri bütünlüğünü kontrol edebilir. DBMS_SQLHASH, MD4, MD5 ve SHA-1 gibi değişik hash algoritmalarını destekler. DBMS_SQLHASH, içerdiği GETHASH fonksiyonu kullanarak sorgu sonucunda bir özet değer geri döndürür. Hash değerleri karşılaştırılarak veride değişiklik olup olmadığı anlaşılır.

```
DBMS_SQLHASH.GETHASH( sqltext IN varchar2, digest_type IN BINARY_INTEGER,
                                    chunk_size IN number DEFAULT 134217728)
RETURN raw;
```

Burada sqltext hash'ı alınacak sql cümlesi, digest_type kullanılan hash algoritması (HASH_MD4, HASH_MD5, or HASH_SH1), chunk_size elde edilen hash değerinin boyutudur. Default chunk_size 128 MB dir.

Zincirleme blok şifreleme ve PKCS #5 'le AES256 algoritması kullanılarak, girilen bir stringi şifreleyen ve deşifreleyen örnek bir program:

```
declare
        giris_string VARCHAR2 (200) := 'Gizli Mesaj';
        cikis_string VARCHAR2 (200);
        sifrelenmis_raw RAW (2000); -- şifrelenmiş binary text
        desifrelenmis_raw RAW (2000); -- deşifrelenmiş binary text
```

```
        sifreleme_anahtari_uzunluk NUMBER := 256/8; -- 256 bit anahtar uzunluğu (32
                        bytes)
        sifreleme_anahtari_raw RAW (32); -- 256 bit şifreleme anahtarı
        sifreleme_tipi PLS_INTEGER :=  DBMS_CRYPTO.ENCRYPT_AES256 +
                                       DBMS_CRYPTO.CHAIN_CBC +
                                       DBMS_CRYPTO.PAD_PKCS5; -- şifreleme tipi
begin
        DBMS_OUTPUT.PUT_LINE ('Orijinal string: ' || giris_string);
        sifreleme_anahtari_raw := DBMS_CRYPTO.RANDOMBYTES (sifreleme_
                                  anahtari_uzunluk);
        sifrelenmis_raw := DBMS_CRYPTO.ENCRYPT
                (
                src => UTL_I18N.STRING_TO_RAW (giris_string, 'AL32UTF8'),
                typ => sifreleme_tipi,
                key => sifreleme_anahtari_raw
                );

        desifrelenmis_raw := DBMS_CRYPTO.DECRYPT
                (
                src => sifrelenmis_raw,
                typ => sifreleme_tipi,
                key => sifreleme_anahtari_raw
                );
        cikis_string := UTL_I18N.RAW_TO_CHAR (desifrelenmis_raw, 'AL32UTF8');
        DBMS_OUTPUT.PUT_LINE ('Deşifrelenmiş string: ' || cikis_string);
end;
```

4.2.5. Hesap Denetimi

Oracle veritabanı, veritabanı değişikliklerinin otomatik olarak kaydedilmesine olanak sağlayan yerleşik bir hesap denetimi mekanizması içermektedir. Bununla birlikte, iki temel tip hesap denetiminden söz etmek olasıdır. Bu tipler;

- Veritabanı bazlı hesap denetimi,
- Standart hesap denetimi,
- Fine-Grained hesap denetimi,
- Uygulama bazlı hesap denetimi.

4.2.5.1. Veritabanı bazlı hesap denetimi

Bu tip işlem takibi, Oracle veritabanı sunucusu içindeki yerleşik hesap denetimi mekanizmalarının kullanılması ile mümkündür. Bu mekanizma, Oracle v9i den sonra ikiye ayrılmaktadır. Bu ayrılma, denetimi yapılacak olan bilginin sorgulanması ya da sorgulanmaksızın kayıt altına alınması şeklinde açıklanabilir. [23]

Standart hesap denetimi:

Bu hesap denetimi şekli için veritabanı parametrelerinden audit_trail değişkeninin düzenlenmesi gerekmektedir. Bu dinamik bir değişken olmayıp, veritabanının yeniden başlatılması gerekmektedir. Temel olarak pek çok olayın kaydını tutabilmektedir. Bu işlemlerin neler olduğu; SYS.AUDIT_ACTIONS tablosunda yeralmaktadır. Herhangi bir nesnenin hesap denetimi sistemini açmak için aşağıdaki komut kullanılmaktadır;

AUDIT [SELECT,INSERT,DELETE,UPDATE,..] ON scott.emp

Burada verilen işlem tipleri, yukarıda verilen tablo içinde bulunabilir. Bu metodun en güçlü yanı oldukça değişik 100'den fazla işlem için kayıt tutulabilmesidir.

Fine-Grained hesap denetimi:

SELECT, INSERT, DELETE ve UPDATE işlemleri için kayıt oluşturabilmekle beraber, koşullu kayıt tutabilmesi özelliği ile oldukça fonksiyonel bir özelliktir. [40]

Bu iki temel hesap denetimi işlemi sonucunda oluşan kayıtlara, iki ayrı sistem görüntüsü (view) üzerinden erişilebilmektedir. Bunlar; DBA_AUDIT_TRAIL ve DBA_FGA_AUDIT_TRAIL'dir. Bu viewlar içinde, kaydı tutulan işlem sırasında çalışan SQL de yer almaktadır. Bu şekilde log araştırması yapabilmek olasıdır. Pratik ve kolay bir metod olan bu yerleşik özellikler, başlangıçta off olarak gelmektedir.

Standart ve FGA hesap denetimi arasındaki temel farklılıklar:

- Standart hesap denetimi, AUDIT_TRAIL parametresi ile veritabanı seviyesinde aktifleştirilmelidir. Ardından veritabanının yeniden başlatılması gerekmektedir. FGA ise herhangi bir parametre gerektirmemektedir.
- FGA sadece SELECT, INSERT, UPDATE, and DELETE işlemlerini kayıt altına alabilmektedir. Standart hesap denetimi ise çok daha fazla işlemi kayıt altına alabilmektedir. Standart hesap denetimi oturum ya da erişim bazında kayıt tutabilir.

- FGA, kayıtları sadece veritabanında tutabilir. İşletim sistemi dosyalarına yazabilmek için ayrıca kod yazmak gerekmektedir. Standard hesap denetimi ise sistem loglarına ya da veritabanına kayıt yazabilir.
- FGA hesap denetimi çok daha esnek bir yapıya sahiptir. Özellikle koşullu kayıt özelliği, trail tablosunun şişmesini engellediği gibi, sadece isteğe bağlı kayıt tabanlı kayıt oluşturulmasına da olanak sağlamaktadır. Standart hesap denetimi bunu yapamamaktadır.

4.2.5.2. Uygulama bazlı hesap denetimi

Bu yöntemde ise temel yaklaşım, triggerlar içinden yapılacak bir hesap denetimi mekanizmasına dayanmaktadır. Herbir tablonun $AUD la biten bir kopyası çıkarılmakta ve tüm alanlar için new_ adlı alanlar (field) oluşturulmaktadır. DML (Data Manipulation Language, Veri tanımlama dili) işlemlerinin yapıldığı trigger içine de bu tabloya kayıt atacak şekilde bir kod eklemek gerekmektedir. Bu metodu kullanabilmek için, uygulama geliştirilirken uygun bir içeriğin düzenlenmesi gereklidir. [40]

4.3. IBM DB2 Güvenliği

4.3.1. Doğrulama (Authentication)

Bir DB2 instance veya veritabanına erişmek için ilk önce kullanıcının doğrulanması gerekir. Doğrulama tipi, bir instance için kullanıcının nasıl ve nerede kontrol edileceğini belirler. Doğrulama tipi, sunucuda veritabanı yönetim konfigürasyonu dosyasında saklanır.

Her bir instance için bir doğrulama tipi vardır, tüm veritabanları ve veritabanı sunucusuna erişimi kontrol altında tutar. Aşağıdaki doğrulama tipleri kullanılabilir:[25]

- SERVER: Yerel işletim sistemi güvenliği kullanılarak sunucu üzerinde doğrulama gerçekleştirmedir. Bağlantı esnasında girilen bir kullanıcı ID ve

şifresinin geçerli bir kullanıcı ID ve şifre olup olmadığı sunucuda kontrol edilir. Eğer başarılı olursa instance'a erişimine izin verilir. Bu varsayılan güvenlik mekanizmasıdır.

- SERVER_ENCRYPT: Sunucu, şifrelenmiş SERVER doğrulama tipini de kullanabilir. Bu doğrulamada kullanıcı bilgileri şifreli olarak sunucuya iletilir.

- CLIENT: Veritabanında bir doğrulama işlemi gerektiğinde işletim sistemi güvenliği çağırılarak doğrulama yapılır. Bir bağlantı veya başvuru esnasında geçerli bir kullanıcı olup olmadığını anlamak için kullanıcı ID ve şifre kullanılır. Doğrulama başarılı olursa kullanıcı ID'ye instance için erişim izni verilir. Doğrulama veritabanı sunucusu üzerinde olmayacaktır.

  Eğer, instance, CLIENT doğrulamaya sahipse, doğrulama diğer iki parametreye göre yapılacaktır: trust_allclnts ve trust_clntauth.

  Güvenilir istemciler, yerel güvenlik sistemi için emniyetli bir duruma sahip istemcilerdir. Doğrulama tipi olarak CLIENT seçildiği zaman, istemcilere karşı korunmak için ek bir seçenek belirlenebilir. Güvenli olmayan istemcilere karşı korunmak için, yönetici, trust_allclnts parametresini NO yaparak, Trusted Client Authentication doğrulama seçebilir. Bu, tüm güvenilir platformların sunucu adına kullanıcı doğrulamasını yapabileceği anlamına gelir. Güvenilir olmayan istemcilerin doğrulaması ise bir kullanıcı ID ve şifreyle sunucu üzerinde yapılır. İstemcilerin güvenilirliğini arttırmak için trust_allclnts konfigürasyon parametresini kullanılabilir. Parametrenin varsayılan değeri YES'tir. CLIENT doğrulama sadece güvenilir istemciler için kullanılmalıdır.

  Güvenilir istemciler için de sunucuda doğrulama yapmak istenebilir. Güvenilir istemcilerin nerede kontrolünün yapılacağını belirtmek için, trust_clntauth konfigürasyon parametresini kullanılır. Varsayılan parametre değeri CLIENT'tır. Eğer trust_clntauth değeri "CLIENT" ise, doğrulama istemcide yapılır. Eğer trust_clntauth değeri "SERVER", kullanıcı adı ve şifre yoksa doğrulama istemcide varsa doğrulama sunucuda yapılır.

DRDA (Distrubuted Relational Database Access) istemcilere karşı DB2'yı korumak için trust_allclnts parametresini DRDAONLY'e ayarlamak gerekir. İstemci taraflı doğrulamada yalnızca bu istemcilere güvenilebilir. Diğer tüm istemciler bir kullanıcı ID ve şifreyle sunucu tarafından doğrulanmalıdır. [25]

- KERBEROS: DB2, işletim sistemi üzerindeki kerberos doğrulamayı kullanarak kullanıcı doğrulama yapabilmektedir. Kerberos doğrulama hakkında detaylı bilgi Bölüm 3.5.4.'te bulunmaktdır.

- KRB_SERVER_ENCRYPT: Sunucu, KERBEROS doğrulamayı veya şifrelenmiş SERVER doğrulamayı kullanabilir. Eğer istemci doğrulaması KERBEROS ise istemcinin doğrulaması kerberos güvenlik sistemi kullanılarak yapılır. Eğer istemcinin doğrulaması SERVER_ENCRYPT ise, istemcinin doğrulaması bir kullanıcı ID ve şifre kullanılarak yapılır. Eğer istemci doğrulaması belirlenmemişse o zaman istemci için kerberos kullanacaktır.

- DATA_ENCRYPT: Sunucu, şifrelenmiş SERVER doğrulamayla ve kullanıcı verisini şifreleyerek doğrulama yapabilir. Bu doğrulama SERVER_ENCRYPT'le aynı şekilde çalışır. Aşağıdaki kullanıcı verileri bu doğrulama tipi kullanılarak şifrelenebilir:

  - SQL ve XQuery cümleleri.
  - Bir SQL veya XQuery cümlesinden oluşan veri.
  - Bir sorgu sonucundan oluşan veri kümesinin tamamı veya bir kısmı.
  - LOB (Large Object) geniş veri dosyaları nesneleri.
  - SQLDA (SQL Descriptor Area) tanımlamaları.

- DATA_ENCRYPT_CMP: Sunucu, şifrelenmiş SERVER doğrulamayla ve kullanıcı verisini şifreleyerek doğrulama yapabilir. Ayrıca, bu doğrulama tipi DATA_ENCRYPT doğrulama tipini desteklemeyen alt seviye ürünlerle uyumludur. [25]

- GSSPLUGIN: Sunucu, bir GSS-API plugin kullanarak doğrulama yapabilir. Eğer istemci doğrulaması belirlenmemişse sunucu, sunucu-destekli plugin'lerin bir listesini döndürür. srvcon_gssplugin_list veritabanı yönetim konfigürasyon parametresi listesindeki herhangi bir kerberos plugini içeren plugini istemci için belirler. İstemci, listede bulunan plugin dizininden ilk plugini seçer. Eğer, istemci listedeki herhangi bir plugini desteklemiyorsa, istemci, kerberos doğrulama tipini kullanılarak doğrulanır. [25]

- GSS_SERVER_ENCRYPT: Sunucu, plugin doğrulamayı veya şifrelenmiş sunucu doğrulamayı yapabilir. Eğer istemci doğrulaması bir plugin yolu ile oluyorsa, istemcinin doğrulaması server destekli pluginlerin listesindeki ilk istemci destekli plugin kullanılarak yapılır. Eğer istemci doğrulaması belirlenmemişse ve bir bağlantı yapılmış ise, sunucu, sunucu-destekli pluginlerin bir listesini, kerberos doğrulamayı ve şifrelenmiş sunucu doğrulamasını döndürür. İstemcinin doğrulaması, istemci plugin dizinindeki pluginlerden ilki kullanılarak yapılır. Eğer istemci listede yer alan pluginlerden herhangi birini desteklemiyorsa, kerberos doğrulama veya şifrelenmiş sunucu doğrulaması kullanılarak doğrulaması yapılır.

4.3.1.1. Uzak istemciler için doğrulama

Bir veritabanı için uzaktan erişim durumu varsa, doğrulama tipi veritabanında belirlenebilir. Eğer, doğrulama tipi belirlenmemişse, istemci için varsayılan olarak SERVER_ENCRYPT olacaktır. Eğer sunucu SERVER_ENCRYPT'ı desteklemiyorsa, istemci, sunucu tarafından desteklenen bir başka doğrulama tipini kullanarak tekrar deneyecektir. Eğer sunucu birden çok doğrulama tipini destekliyorsa istemci bunların arasından seçmeyecek ve bir hata döndürecektir. Bu hata doğru doğrulama tipini kullanmasını sağlayacaktır. Bu durumda, istemci desteklenen bir doğrulama tipi kullanarak veritabanı işlemini yapacaktır.[25]

AUTHENTICATION parametresini DATA_ENCRYPT yapmak için aşağıdaki komut kullanılabilir:

```
UPDATE DBM CFG USING AUTHENTICATION DATA_ENCRYPT
db2stop
db2start
```

4.3.2. Yetkilendirme

Kullanıcıların SQL veya XQuery sorgularını çalıştırabilmeleri için gerekli hakka sahip olmaları gerekir. Yine aynı şekilde tablo yaratmak için tablo yaratma hakkı, tablonun yapısını değiştirebilmek için değiştirme hakkı olması gerekir. DB2'da 2 tür yetki bulunmaktadır: Yönetim yetkisi ve haklar.

Veritabanı yöneticisi, her bir veritabanı fonksiyonunu gerçekleştirmek, özel bir işlem yapmak için kullanıcıdan yetki bilgisi ister.

- Yönetim yetkisi: Yönetim yetkisine sahip kişiler veritabanı yönetim kontrol görevlerini yerine getirerek; verinin bütünlüğünden ve güvenliğinden sorumlu olurlar. Yetki seviyeleri, hakların gruplandırılması ve yüksek seviyede veritabanı yöneticisi bakım işlemleri ve benzer yararlı operasyonların gerçekleştirilmesine imkân sağlar. Yönetim yetkileri olarak instance seviyesinde çalışan SYSADM, SYSCTRL, veritabanı seviyesinde çalışan DBADM, SECADM, SYSMAINT ve SYSMON vardır. Veritabanı yetkileri ise tablo yaratma, bir tablonun içine veri yükleme vb. işlemler için kullanılır. Örneğin, LOAD veritabanı yetkisi.

- Haklar: Haklar bir kullanıcıya işlerini gerçekleştirebilmeleri için verilen izinlerdir. Yetkilendirilmiş kullanıcılar nesneleri yaratabilir, onlara erişebilir, sahipliğini alabilir ve GRANT cümleciğini kullanarak diğer kullanıcılara bu nesneler üzerinde haklar verebilir. Haklar bireysel kullanıcılara, gruplara veya PUBLIC'e verilebilir. PUBLIC tüm kullanıcıları ve gelecekte yaratılacak kullanıcıları da içeren özel bir gruptur. Kullanıcılar bir grubun üyesi olabilirler, haklar gruba verilerek bazı avantajlar sağlanabilir.

- Bireysel haklar: Belirli nesneler üzerinde belirli görevleri yerine getirmek için kullanıcıya verilen izinlerdir. Yönetim yetkisine (SYSADM veya DBADM) veya CONTROL hakkına sahip kullanıcılar bu hakları, diğer kullanıcılara verebilir ve onlardan geri alabilir.

  WITH GRANT OPTION komutuyla tablo, görüntü, şema, paket, routine ve sequence hakları diğer kullanıcıla verilebilir. WITH GRANT OPTION komutuyla verilmiş olan bir hakkı geri alınamaz. Söz konusu hakkı geri almak için SYSADM, DBADM veya CONTROL yetkisine sahip olmak gerekir.

- Nesne Sahipliği: Nesne yaratıldığında, nesnenin sahibini belirten bir yetki ID belirlenir. Sahiplik, nesne üzerinde SQL ve XQuery sorgularını yapabilmekle yetkilendirilmiş kişidir. CREATE SCHEMA deyimi için AUTHORIZATION komutu kullanılarak nesnenin sahipliği bir başka kullanıcıya verilebilir.

  Örneğin,

  ```
  CREATE SCHEMA SCOTTSTUFF AUTHORIZATION SCOTT
  CREATE TABLE T1 (C1 INT)
  ```

  Bu komut, SCOTTSTUFF şemasını ve SCOTTSTUFF.T1 tablosunun her ikisini SCOTT ‘ın sahipliğinde yaratır.

Haklar, nesne yaratıldıktan nesne sahibine verilir:

- CONTROL hakkı, yeni yaratılan tablolar, indeksler ve paketler için verilir. Bu hak, nesneyi yaratanın veritabanına erişimine ve nesne üzerinde diğer kullanıcılara haklar verme ve verilmiş hakları geri alma imkânları sağlar. CONTROL hakkı, SYSADM veya DBADM yetkisine sahip kullanıcı tarafından verilebilir. Kontrol hakkı nesne sahibinden geri alınamaz.
- CONTROL hakkı, görüntü tarafından kullanılacak tüm tablolar, görüntüler, nicknameler üzerinde CONTROL hakkına sahipse yeni yaratılan görüntü için de verilebilir.
- Triggerlar, yordamlar, sequences, tablespaceler ve buffer pools gibi diğer nesneler için CONTROL hakkına sahip olunamaz.

Nesne sahibi otomatik olarak nesneyle ilgili her tür hakka sahip olur (GRANT deyiminin WITH GRANT seçeneği kullanılarak bu haklar diğer kullanıcılara verilebilir). Ayrıca, nesne sahibi nesneyi değiştirebilir veya silebilir. Bu yetkiler açık olarak nesne sahiplerine aittir ve geri alınamaz. Tabloyu değiştirmek gibi nesne üzerindeki haklar, sahibi, SYSADM veya DBADM tarafından verilebilir ve yine onlar tarafından geri alınabilir. [57]

Bir güvenlik yöneticisi (SECADM) veya nesne sahibi bir veritabanı nesnesinin sahipliğini değiştirmek için TRANSFER OWNERSHIP deyimini kullanabilir.

4.3.2.1. Yetkiler, haklar ve yetkilendirme

4.3.2.1.1. Sistem yönetim yetkisi (SYSADM)

SYSADM yetkisi, yönetim yetkisinin en üst seviyesidir. SYSADM yetkili kullanıcılar, yazılımları, veritabanı ve veritabanı yönetici programı komutlarını çalıştırabilir. Veritabanı instance'ının bir veritabanında bulunan tablonun LBAC tarafından korunmamış verisine erişim sağlayabilir. Tüm veritabanı nesneleri, veritabanları, tablolar, görüntüler, indeksler, paketler, şemalar, sunucular, aliaslar, veri tipleri, fonksiyonlar, procedureler, triggerlar, tablespaceler, buffer pools ve olay monitorleri üzerinde kontrol olanağı sağlar. SYSADM yetkisi sysadm_group konfigürasyon parametresiyle gruba verilebilir.

SYSADM yetkisine sahip bir kullanıcı aşağıdaki işlemleri gerçekleştirebilir:

- Bir veritabanını taşıma.
- Veritabanı yönetici konfigürasyon dosyasını değiştirme.
- DBADM yetkisini verme ve geri alma.
- SECADM yetkisini verme ve geri alma.

SYSADM yetkisi diğer yetkiler tarafından sağlanan yetkilerin çoğuna sahipken, SECADM yetkisinin imkânlarının hiç birini sağlayamaz. SECADM yetkisi diğer hiçbir yetki tarafından sağlanamaz. SYSADM yetkisi, LBAC tarafından korunan veriye erişim sağlayamaz.

4.3.2.1.2. Sistem kontrol yetkisi (SYSCTRL)

SYSCTRL yetkisi sistem kontrol yetkisinin en üst seviyesidir. Bu yetki, veritabanı yönetici instance ve onun veritabanlarında bakım ve işlemleri gerçekleştirme imkânı sağlar. Sistem kontrol yetkisi, kullanıcıların, önemli verilerin bulunduğu veritabanı yönetici instance'nı yönetmesi için tasarlanmıştır.

SYSCTRL yetkisi, sysctrl_group konfigürasyon parametresiyle gruba verilebilir. SYSCTRL yetkisine sahip bir kullanıcı aşağıdaki işlemleri gerçekleştirebilir:

- Bir veritabanını veya dağıtık bağlantı servislerini güncelleştirme.
- Sistemin dışındaki kullanıcıları engelleme.
- Bir veritabanı yaratma ve düşürme.
- Bir tablespace yaratma, değiştirme ve düşürme.
- Veritabanını geri yükleme.

Ayrıca, SYSCTRL yetkisine sahip bir kullanıcı, kullanıcıların sistem bakım yetkisine (SYSMAINT) ve sistem izleme yetkisine (SYSMON) sahip fonksiyonlarını gerçekleştirebilir. [20]

Kullanıcılar SYSCTRL yetkisiyle bir veritabanına bağlantı yapma hakkına sahiptirler. Ayrıca, kullanıcılar SYSCTRL yetkisiyle veritabanları yarattıklarında, veritabanı üzerinde DBADM yetkisine de otomatik olarak sahip olurlar.

4.3.2.1.3. Sistem bakım yetkisi (SYSMAINT)

SYSMAINT yetkisi, sistem kontrol yetkisinin ikinci seviyesidir. Bu yetki, instance ve onun veritabanlarında bakım ve bazı işlemleri gerçekleştirme imkânı sağlar. Veritabanlarındaki veriye direkt erişime izin vermez. Sistem bakım yetkisi, instance'daki veritabanların kullanıcılar tarafından bakımlarının yapılmasını sağlar.

SYSMAINT yetkisi grup için sysmaint_group konfigürasyon parametresi kullanılarak belirlenir. SYSMAINT yetkisine sahip bir kullanıcı aşağıdakileri yapabilir:

- Veritabanı konfigürasyon dosyalarını güncelleştirme.
- Veritabanı veya tablespace'in yedeğini alma.
- Veritabanını geri yükleme.
- Veritabanını kurtarma.
- Bir instance'ı başlatma veya durdurma.
- Bir tablespace'ı geri yükleme.
- Trace çalıştırma.
- İnstance'ı ve onun veritabanlarını izleme.

SYSMAINT yetkili kullanıcılar bir veritabanına bağlanma yetkisine açık olarak sahip olmakla beraber, sistem izleme yetkisine (SYSMON) sahip kullanıcıların işlemlerini de gerçekleştirebilir.

4.3.2.1.4. Güvenlik yönetim yetkisi (SECADM)

SECADM yetkisi yalnızca SYSADM tarafından bir kullanıcıya verilebilir, bir gruba verilemez. Şunları gerçekleştirebilir:

- Güvenlik etiketi bileşenlerini yaratma ve silme.
- Güvenlik politikalarını yaratma ve silme.
- Güvenlik etiketlerini verme ve geri alma.
- LBAC dokunulmazlıklarını verme ve geri alma.
- setsessionuser haklarını verme ve geri alma.
- Nesneler üzerinde TRANSFER OWNERSHIP SQL deyimini çalıştırma.

4.3.2.1.5. Veritabanı yönetim yetkisi (DBADM)

DBADM yetkisi veritabanı ve veritabanında işlemler gerçekleştirmek için kullanılan bir yönetim yetkisidir. DBADM yetkisi, LBAC tarafından korunmayan herhangi bir tablodaki veriye erişebilir. LBAC tarafından korunan veriye erişmek için LBAC muafiyetlerine (credentials) sahip olmak gerekir. DBADM yetkisi verildiğinde aynı veritabanı için otomatik olarak aşağıdaki yetkiler de verilmiş olur.[25]

- BINDADD
- CONNECT
- CREATETAB
- CREATE_EXTERNAL
- ROUTINE, CREATE
- NOT_FENCED_ROUTINE
- IMPLICIT_SCHEMA
- QUIESCE_CONNECT
- LOAD

DBADM yetkisine sahip kullanıcılar diğer kullanıcılara veritabanı üzerinde haklar verebilirler ve geri alabilirler. DBADM veya daha üst bir yetkiye sahip bir kullanıcı veritabanında Log dosyalarını okuma, olay izlemeleri yaratma, aktifleştirme ve silme işlemlerini de gerçekleştirebilir.

DBADM yetkisi, SECADM yetkisinin yapabildiği şeyleri sağlamaz. SECADM yetkisinin yetkilerini de diğerleri sağlayamaz.

4.3.2.1.6. Sistem izleme yetkisi (SYSMON)

SYSMON yetkisi, instance ve instance'da yer alan veritabanı sitemini izlemeyi sağlar. SYSMON yetkisi, sysmon_group konfigürasyon parametresi kullanılarak gruba verilir. SYSMON yetkisiyle aşağıdaki komutları çalıştırma yetkisi verilir:

- GET DATABASE MANAGER MONITOR
- GET MONITOR SWITCHES
- GET SNAPSHOT
- LIST ACTIVE DATABASES
- LIST APPLICATIONS
- LIST DCS APPLICATIONS
- RESET MONITOR
- UPDATE MONITOR SWITCHES

SYSADM, SYSCTRL veya SYSMAINT yetkili kullanıcılar, SYSMON yetkisine de sahip olurlar.

4.3.2.1.7. LOAD yetkisi

LOAD yetkisi veritabanı seviyesinde olup bu yetkiye sahip kullanıcılar, tablo üzerinde INSERT hakkının yanı sıra, tabloya veri yüklemek için LOAD komutunu kullanabilir.

4.3.2.1.8. Veritabanı yetkileri

Her bir veritabanı yetkisi, veritabanı üzerinde işlemleri gerçekleştirmek için bir yetki ID'ye izin verir. Veritabanı yetkileri farklı haklardan oluşur. Yetkiler, bir tablo veya bir index gibi veritabanının belli bir bölümü üzerinde belirli bir işlemin yapılmasına müsaade eder. 10 adet veritabanı yetkisi vardır. [25]

- SECADM: Veritabanı nesnelerinin sahipliğini değiştirmek dâhil, veritabanı güvenliğiyle ilgili çoğu şeyi konfigüre etmek için kullanılır. İnstance için, tüm nesneler, etiket tabanlı kontrolun (LBAC) bir parçası olarak SECADM yetkisiyle yaratılabilir, silinebilir, kullanıcılara yetki verilebilir veya verilen yetki kullanıcılardan geri alınabilir.
- DBADM: Veritabanı yöneticisi gibi hareket eder. SECADM hariç diğer veritabanı yetkilerinin tamamına sahiptir.
- CONNECT: Veritabanına bağlantı kurulmasına müsaade eder.
- BINDADD: Veritabanında yeni paketler yaratmak için kullanılır.
- CREATETAB: Veritabanında yeni tablolar yaratılmasına izin verir.
- CREATE_EXTERNAL_ROUTINE: Veritabanının diğer kullanıcılar ve uygulamalar tarafından kullanılması için procedure yaratmaya yarar.
- CREATE_NOT_FENCED_ROUTINE: Kullanıcı tanımlı fonksiyon veya korunmayan procedureler yaratmak için kullanılır.
- IMPLICIT_SCHEMA: Şema yaratmaya müsaade eder.
- LOAD: Bir tablo içine veri yüklemeye müsaade eder.
- QUIESCE_CONNECT: Veritabanı bekleme durumundayken erişimin sağlanmasına imkân verir.

Yalnızca SYSADM yetkili kullanıcı, SECADM ve DBADM yetkilerini verebilir. Diğer tüm yetkiler SYSADM veya DBADM yetkilerine sahip kişiler tarafından verilebilir.

Bir veritabanı yaratıldığında, aşağıdaki veritabanı yetkileri otomatik olarak PUBLIC'e verilir:

- CREATETAB
- BINDADD
- CONNECT
- IMPLICIT_SCHEMA

PUBLIC'ten herhangi bir yetkiyi geri almak için DBADM veya SYSADM yetkisine sahip kullanıcı olmak gerekir.

4.3.2.1.9. Şema yetkisi (IMPLICIT_SCHEMA)

Bu yetkiye sahip herhangi bir kullanıcı bir, CREATE SCHEMA komutunu kullanarak şema yaratabilir ve şemayı var olmayan bir şema adı kullanarak adlandırabilir. Yeni bir veritabanı yaratıldığı zaman, PUBLIC'e IMPLICIT_SCHEMA veritabanı yetkisi verilir. [25]

4.3.2.1.10. Şema hakları

Şema hakları nesne hakları içinde yer alır. Şema hakları bir veritabanı üzerindeki şemalar üzerindeki işlemleri kapsar. Bir kullanıcıya aşağıdaki haklar verilebilir:

- CREATEIN kullanıcıya şema içinde nesneler yaratma izni verir.
- ALTERIN kullanıcıya şema içindeki nesneleri değiştirme izni verir.
- DROPIN kullanıcıya şema içinden nesneleri silme izni verir.

Şemanın sahibi bu hakları diğerlerine verebilir. Nesneler şema içinde bulunur, bunlar: tablolar, görüntüler, indexler, paketler, veri tipleri, fonksiyonlar, triggerlar, procedureler ve aliaslar.

4.3.2.1.11. Tablespace hakları

Tablespace hakkı bir veritabanındaki tablespaceler üzerindeki işlemleri içerir. Kullanıcıya tablespace içinde tablolar yaratmak için, USE hakkı verilir.

4.3.2.1.12. Tablo ve görüntü hakları

Tablo ve görüntü hakları, bir veritabanındaki tablolar veya görüntüler üzerindeki işlemlerle ilgilidir. Kullanıcı aşağıdaki haklar için veritabanı üzerinde CONNECT yetkisine sahip olmalıdır:

- CONTROL, bir tablo veya görüntüyü silme, tablo haklarını verme, geri alma gibi tüm hakları kullanıcıya sağlar. Tablonun yaratıcısı otomatik olarak tablo üzerinde CONTROL yetkisine sahip olur.
- ALTER, bir tablo üzerinde değişiklikler yapmaya müsaade eder.
- INDEX, bir tablo üzerinde indexler yaratmaya izin verir. İndexi yaratanlar index üzerinde otomatik olarak CONTROL hakkına sahip olurlar.
- INSERT, bir tablo veya görüntü içine kayıt eklemeye ve IMPORT özelliğini kullanmaya müsaade eder.
- REFERENCES, bir tablodaki ilişkileri belirlemeye, foreign key oluşturma ve silme işlemlerine müsaade eder.
- SELECT, bir tablo veya görüntüden kayıtlar seçmeye, tabloda bir görüntü yaratmaya ve EXPORT komutunu çalıştırmaya yarar.
- UPDATE, bir tablo veya görüntüdeki bir veya birden fazla sütundaki kayıdı değiştirmeye müsaade eder.

Bu haklar GRANT üzerindeki WITH GRANT OPTION kullanılarak verilebilir.

4.3.2.1.13. Paket hakları

Paket, bir veritabanı nesnesi olup bir program için, veritabanı yöneticisi yoluyla, en etkili şekilde veriye erişmek için gerekli olan bilgiyi içerir. Paket hakkı, paketleri yaratma ve işletmek için kullanıcıya verilir. Paket için kullanıcıya, CONTROL BIND, EXECUTE hakları verilebilir. [25]

4.3.2.1.14. Index hakları

Br index yaratma imkânı sağlar. ndexin yaratıcısı index üzerinde otomatik olarak CONTROL yetkisine sahip olur.

4.3.2.1.15. Sequence hakları

Sequence'in yaratıcısı Sequence üzerinde otomatik olarak USAGE ve ALTER haklarını elde eder. USAGE, sequence için NEXT VALUE ve PREVIOUS VALUE ifadelerini kullanmak için gerekli bir haktır. Sequence üzerinde ALTER hakkı kullanıcılara sequence'i yeniden başlatmak ve sequence değerlerini değiştirme imkânı sağlar.

4.3.2.1.16. Routine hakları

Bir veritabanı içindeki metotlar, procedüreler, fonksiyonlar gibi tüm routine tipleri üzerinde işlemler içeren haklardır. EXECUTE hakkına sahip olunduğunda, kullanıcı routine'i çağırabilir, fonksiyon yaratabilir.

4.3.3. Veri Şifreleme

DB2 veri şifrelemede, şifreleme ve deşifreleme fonksiyonları kullanılır: ENCRYPT, DECRYPT,_BIN, DECRPYT_CHAR ve GETHINT.

ENCRYPT fonksiyonu, bir parola tabanlı şifreleme metodu kullanarak veri şifrelemedir. Bu fonksiyonlar bir şifre hatırlatması kullanılmasına da imkân tanır. Parola hatırlatması, şifrelenmiş verinin içerisine gömülür. Şifrelenmiş veri yalnızca doğru parola kullanılarak deşifre edilebilir.

ENCRYPT fonksiyonlarının sonucu, VARCHAR FOR BIT DATA(32631'le limitli) veri türü olur. [25]

Yalnızca CHAR, VARCHAR ve FOR BIT DATA veri türü şifrelenebilir.

DECRYPT_BIN ve DECRYPT_CHAR parola bazlı şifreleme kullanarak veriyi deşifreleyen fonksiyonlardır.

DECRYPT_CHAR daima VARCHAR döndürürken, ECRYPT_BIN daima VARCHAR FOR BIT DATA geriye döndürür.

GETHINT fonksiyonu, parola hatırlatma içeren bir bilgi geri döndürür. Bir parola hatırlatma, sahiplerine parolaları hatırlatmak için kullanılan bir bilgidir. Örneğin "Ömer" parolasını hatırlamak için hatırlatma olarak "Oğlum" kullanılabilir.

Veriyi şifrelemek için kullanılan parola için aşağıdaki iki yöntemden birisine karar verilir:

- Parola argümanı. Parola bir string olup açık olarak ENCRYPT fonksiyonu tarafından çağrılır. Veri, verilen parolayla, şifrelenir ve deşifrelenir.
- Şifreleme parolasının özel olarak saklanması. SET ENCRYPTION PASSWORD deyimi parolayı şifreler ve şifrelenmiş parolayı özel bir yerde saklanması için veritabanı yöneticisine gönderir. ENCRYPTION PASWORD özel kaydı yalnızca şifrelenmiş olarak saklanır.

Özel kayıttaki başlangıç ve varsayılan değer boş bir stringtir.

Şifreler için geçerli uzunluklar 6 ile 127 arasıdır. Şifre hatırlatmalar için geçerli uzunluklar 0 ile 32 arasıdır.

Örneğin, bir işçiye ait sosyal güvenlik numarasını şifrelenmiş olarak EMP adında bir tabloda saklayan bir uygulama yapalım. Bu örnekte ENCRYPTION PASSWORD değeri kullanılarak şifreleme yapılmaktadır.

```
CREATE TABLE emp (ssn varchar(124) for bit data);
SET encryption password = 'Ben123';
INSERT INTO emp (ssn) VALUES (encrypt('289-46-8832'));
INSERT INTO emp (ssn) VALUES (encrypt('222-46-1904'));
INSERT INTO emp (ssn) VALUES (encrypt('765-23-3221'));
SELECT decrypt_char(ssn) FROM emp;
```

ENCRYPTION PASSWORD değeri kullanılarak bir görüntü (view) oluşturuluyor.

```
CREATE VIEW clear_ssn (ssn) AS SELECT decrypt_char(ssn) FROM emp;
```

Şimdi de ENCRYPTION PASSWORD değeri 'Ben123' belirlenerek clear_ssn görüntüsündeki kayıtlar listelenmektedir:

```
SET encryption password = 'Ben123';
SELECT ssn FROM clear_ssn;
```

Bir başka örnek vermek gerekirse, bir web uygulamasının müşteri hakkındaki bilgiyi tuttuğu varsayılsın. Bu uygulamada, custname müşteri adı, cardnum kart numarası ve userpswd şifresi için kullanılmaktadır. Bir kayıt ekleme işlemi şu şekilde gerçeklenir:

```
INSERT INTO customer (ccn, name) VALUES (encrypt(:cardnum, :userpswd), :custname)
```

Müşteri tarafından doğru olarak girilen bir müşteri adı ve şifre kombinasyonu sonucu uygulamanın girilen müşteri kredi kart bilgisini göstermesi şu şekilde olur:

```
SELECT decrypt_char(ccn, :userpswd) FROM customer WHERE name = :custname;
```

Aşağıdaki örnek müşterilerin şifrelerini hatırlamalarına yardımcı olacak şifre hatırlatma sözcüğünü kullanmaktadır. Yukarıdaki örnek kullanılırsa, şifre hatırlatma değişkeni olarak pswdhint kullanılsın. 'Ankara' şifresi için 'Başkent' şifre hatırlatma kelimesi kullanılacak varsayılsın.

```
INSERT INTO customer (ccn, name) VALUES (encrypt(:cardnum, :userpswd, :pswdhint),
:custname)
```

Eğer müşteri bir şifre hatırlatma isteğinde bulunursa şöyle bir sorgu kullanılabilir:

```
SELECT gethint(ccn) INTO :pswdhint FROM customer WHERE name = :custname;
```

Sorgu sonucu hatırlatma sözcüğü olarak 'Başkent' değeri dönecektir.

4.3.4. Etiket tabanlı erişim kontrolü (LBAC)

Etiket tabanlı erişim kontrolü (LBAC), veriye kimin erişebileceğinin kontrolünü etkili bir şekilde yapılmasına ve güçlendirilmesine fayda sağlar. LBAC, hangi satırlar ve hangi sütunlar üzerinde kimin yazma erişimi, kimin okuma erişimi olacağının açık olarak belirlenmesine yardım eder.

LBAC özelliği çok iyi ayarlanabilir bir özelliktir ve güvenlik yapısına çok iyi adapte edilebilir. Tüm LBAC konfigürasyonu güvenlik yöneticisi tarafından yapılır.

Bir güvenlik yöneticisi, güvenlik politikaları yaratarak, LBAC sistemini konfigüre eder. Bir güvenlik politikası, hangi veriye kimin erişeceğine karar vermek için kullanılacak kriterleri tanımlar. Bir güvenlik politikası, herhangi bir tabloyu korumak için kullanılabilir. Farklı tablolar farklı güvenlik politikaları tarafından korunabilir.

Bir güvelik politikası yarattıktan sonra, bir güvenlik yöneticisi politikanın parçası olan ve güvenlik etiketleri olarak adlandırılan nesneler yaratır. Açık olarak bir güvenlik etiketinin ne yapacağına güvenlik politikası karar verir ve veri parçaları için kimin erişim hakkı olacağına karar vermeye yarayacak kriterlere göre konfigüre edilir. Bir kişinin şirketteki pozisyonuna ve ne tür projelerde neler yapması gerektiğine karar verilebilirse, ona göre güvenlik etiketi konfigüre edilebilir. LBAC yeterince esnek bir olup, çok karmaşık sistemlerden basit sistemlere kadar yüksek veya düşük güvenlik gerektiren sistemlerde kullanılabilir. [25]

Bir güvenlik etiketi tablo içinde tutulan veriyi korumak için özel sütun ve satırlarla ilişkili olabilir. Bir güvenlik etiketi tarafından korunan veriye korunmuş veri denir. Bir güvenlik yöneticisi kullanıcılara güvenlik etiketleri tarafından korunmuş verilere erişebilmeleri için izinler verebilir. Bir kullanıcı, korunmuş veriye erişmek istediğinde kullanıcının güvenlik etiketi, güvenlik etiketiyle korunan veriyi karşılaştırılır. Koruma etiketi bazı güvenlik etiketlerini engelleyecek ve bazılarını da engellemeyecektir.

Çoklu güvenlik politikaları için, kullanıcı güvenlik etiketleri barındırabilir. Herhangi bir güvenlik politikasında, bir kullanıcı en fazla bir etiket okuma erişimi için bir etiket yazma erişimi için barındırabilir.

Bir güvenlik yöneticisi kullanıcılara muafiyetler verebilir. Bir muafiyet, korunmuş veriye erişilmesine izin verir aksi takdirde güvenlik etiketleri erişimi engelleyebilir. Bununla beraber güvenlik etiketleri ve muafiyetler, LBAC güvenlik belgeleri olarak adlandırılır.

Eğer korunmuş bir sütuna erişmeye çalışılınırsa, kullanıcı LBAC güvenlik belgesi erişime izin vermeyecektir ve başarısız olduğuna dair bir hata mesajı alınır.

Eğer korunmuş satırları okuma girişiminde bulunulursa kullanıcının LBAC güvenlik belgesi okumaya müsaade etmeyecektir ve bu durumda DB2 kayıtlar yok gibi rol yapar. Bu satırlar çalıştırılacak SELECT, UPDATE veya DELETE içeren bir SQL cümlesiyle seçilemeyecektir. Toplama fonksiyonları, LBAC güvenlik belgelerinin okumaya müsaade etmediği kayıtları görmezden gelecektir, yok sayacaktır. Örneğin, COUNT(*) fonksiyonu, sadece okuma izni olan kayıtların sayısını geri döndürebilecektir.

LBAC neleri sağlamaz: [25]

- LBAC asla isteğe bağlı erişim denetimi yasaklanmış veriye erişime izin vermez.
- LBAC güvenlik belgesi yalnızca korunmuş veri için erişim kısıtlaması sağlar.
- LBAC güvenlik belgeleri, bir veritabanı veya tabloyu düşürüldüğünde, tablo veya veritabanının korunmuş veri içerip içermediğine bakmaz.
- LBAC güvenlik belgeleri, verinin yedeğini alma durumunda herhangi bir kontrol yapmaz. Eğer bir tablo için yedekleme çalıştırılırsa, veri üzerinde LBAC koruması olsa bile kayıtların yedeği hiçbir kısıtlama olmaksızın alınır. Yedeğin bulunduğu medyada veri LBAC tarafından korunmaz. Veri sadece veritabanında korunur.

4.3.5. Hesap denetimi

Doğrulama, yetkiler ve haklar veriye beklenen erişim veya tanınmayı kontrol etmek için kullanılır, fakat bu metotlar veriye beklenmeyen erişim veya tanınmayan kişiler konusunda yetersiz kalabilir. Bir diğer veri erişim şekli, DB2 veritabanı tarafından sağlanan denetleme özelliğidir. İstenmeyen veri erişiminin izlenmesi ve sonraki analiz, veri erişiminin kontrol altına alınmasını geliştirmeye ve veriye kötü niyetli ve dikkatsiz, yetkisiz erişimi önlemek için önemli bir yer tutar. Sistem yöneticileri tarafından bireysel kullanıcı erişimi ve uygulamayı izleme, veritabanı sisteminizdeki hareketlerin kaydını sağlar.

DB2 veritabanı denetim özelliği, önceden tanımlanmış veritabanı olayları için bilgiler üretir ve denetim işleminin devam etmesi için imkânlar sağlar. Bir denetim log dosyasında tutulan kayıtlar bu özellik sayesinde üretilir. Kayıtların analizi sistemi kötü kullananların kimliğini saptamak için kullanılabilir. Bir kere kimliği saptandığında, sistemi kötü kullananlar sistemden çıkarılabilir veya zararları azaltılabilir.

Denetim servisi, instance seviyesinde görevini yapar, tüm veritabanı ve instance seviyesindeki hareketlerin kaydını tutar. [10]

Birden daha fazla veritabanı bölümünde denetim kayıtları üretilebilir. Her bir denetim kaydı meydana geldiği veritabanı bölümü hakkında ve koordinatör bölüm hakkında bilgi içerir.

Denetim log (db2audit.log) ve denetim konfigürasyon dosyası (db2audit.cfg) instance'ın güvenlik altdizininde bulunur.

Denetim servisi için yetkili kullanıcılar, aşağıdaki hareketleri kontrol edebilirler:

- DB2 veritabanı instance içindeki denetlenebilir olayların kaydını başlatabilirler ve durdurabilirler.
- Denetim servisi özelliğinin konfigüre edilmesi, denetlenebilir olaylardan seçilen kategorileri içeren hareketler kaydedilebilir.
- Denetim kayıtları bir dosyadan veya limitsiz ASCII dosyasından, formatlanarak ve kopyalanarak seçip alınabilir, çıkarılabilir.
- Aktif denetim kayıtlarından kayıtları özetlenebilir.

Denetim kayıtları farklı kategorilerde üretilebilir. Denetleme için mevcut olayların listesi aşağıda yer almaktadır. Her bir kategori için bir anahtar kelime kullanılmaktadır:

- Denetim (AUDIT): Denetleme ayarları değiştirildiğinde veya denetleme log'a erişildiğinde kayıtlar üretir.

- Yetki kontrolü (CHECKING:. DB2 veritabanı nesneleri veya fonksiyonlarına erişmek veya kullanmak için girişimde bulunulduğunda yetki kontrolü esnasında kayıtlar üretir.
- Nesne bakımı (OBJMAINT): Veri nesnelerini yaratıldığında veya düşürüldüğünde kayıtlar üretir.
- Güvenlik bakımı (SECMAINT): Nesne, veritabanı yetkileri veya DBADM yetkisi için, yetki verme ve geri almada kayıtlar üretir.
- Sistem yönetimi (SYSADMIN): SYSADM, SAYMAINT veya SYSCTRL yetkileri işlemler gerçekleştirdiğinde kayıtlar üretir.
- Kullanıcı geçerlilik kontrolü (VALIDATE): Sistem güvenlik bilgisini alma veya kullanıcıları doğrulamada kayıtlar üretir.
- İşlem içeriği (CONTEXT): Bir veritabanı işlemi gerçekleştirildiğinde işlemin içeriğini gösteren kayıtlar üretir.

5. VERİTABANLARINDAKİ RİSKLER

5.1. Güvenlik Boşlukları

Kurumun ağına, ağ kaynaklarına ve verilere yapılabilecek saldırılar aynı zamanda veritabanına da zarar verebilmeleri anlamına gelmektedir. Veritabanı güvenliği dosya, kayıt ve alan (field) gibi farklı seviyelerde ele alınmalıdır. Veritabanlarındaki güvenlik boşluklarından bazıları şunlardır:

- Yazılımdan kaynaklanan riskler: [56]
  - Veritabanı satıcı üretim hataları (bug), karışık yapılandırmalar.
    - Ara bellek taşması (Buffer Overflow).
    - Programlama hataları.
    - Zayıf tasarım (Yetersiz şifreleme şekli).
    - Karışık yapılandırmalar (Ör. Oracle'da init.ora dosyasında "Remote_OS_ Authent=True" parametresinin bu şekilde olması önemli güvenlik sorunlarına yol açmaktadır).

- Yetkisiz sunucu erişimi:
  - Güvenlik duvarındaki veritabanını ilgilendiren portların erişim kontrolünde başarısızlık.
  - Erişim kontrol yöntemlerinden yoksun olmak.
  - Veritabanı sunucusunun fiziki güvenliğinin olmaması,
  - Sunucu üzerinde gereksiz bileşenlerin bulundurulması,

- Sistem yöneticilerinden kaynaklanan riskler:
  - Güvenilir ayarlar olup doğru şekilde kullanılmaması.
  - Kullanıcılara aşırı yetkiler verme.
  - Sistem yapılandırmasında yetkisiz değişiklikler,

- İşletim sistemi yamalarının zamanında yapılmaması,
- Riskli varsayılan ayarlar, varsayılan uygulamalar ve yapılandırmalardan kaynaklanan güvenilir olmayan seçenekler,
- Güvensiz veritabanı yedeklemeleri,

- Kullanıcılardan kaynaklanan riskler:
- Yetersiz şifre kalitesi,
- Erişim yetkisi olmayan personel tarafından saldırılar,
- Bir veritabanı için veritabanı yöneticisi haklarına sahip kullanıcılar,
- Hoşnutsuz, mutsuz veya meraklı personel,

- Çevresel Nedenler:
- Ağ trafiğinin dinlenmesi.
- Güvenli olmayan ağ hatları ve hatalı yapılandırılmış ağ elemanları.
- Önemli verilerin ağ üzerinden açık olarak geçirmek.

5.2. Temel Saldırı Metotları

Yukarıda güvenlik boşluğu olarak görülen durumlar aynı zamanda saldırıcılar için veritabanındaki veriye izinsiz erişim, dinleme, kopyalama, zarar verme, bilginin kaybolması, ulaşılamaz veya kullanılamaz duruma gelmesi amacıyla da kullanılabilir. En yaygın saldırı yöntemleri şunlardır:

1. Ağ trafiğinin gizlice dinlenmesi: Hassas veri ağ dinleyicilerinden korunmalıdır.
2. Yetkisiz sunucu erişimi: Veritabanına direkt erişimde yetkisiz olan sunucu erişimini önlemek için istemcilere kısıtlama olmalıdır. Aksi takdirde veritabanı sunucusunu yetkisiz erişimlerine karşı elverişli yapar.
3. Direkt bağlantı saldırıları: Direkt bağlantı saldırıları, doğrulaması yapılan kullanıcılar ve kullanıcı adı şifresi olmayan kullanıcıların her ikisini de kapsamaktadır. Veritabanı sunucusuna direkt bir bağlantı sağlayan kullanıcı değişik komutları çalıştırabilir. Örn. Query Analyzer (lsqlw.exe). Sunucu bilgisi vb. bilgiler saldırıcı tarafından açığa çıkarılabilir, portlar dinlenebilir.

4. Şifre kırma: Saldırıların diğre bir saldırı yöntemi de hesap adlarını bildikleri kişilerin şifrelerini kırmaya çalışmaktır. SQL de "sa" ve Oracle da "Sys" veya "System" gibi. Bazı şifre kırma yöntemleri, zayıf veya boş şifreler, şifrede basit kelime, sözcük kullanımı olarak söylenebilir.

   Temel şifre kırma saldırıları: Sözlük saldırısı, şifre zorlama yoluyla saldırı, elle şifre tahmini gibi yöntemleri örnek olarak gösterebiliriz. Bu saldırılara karşı koymak için, sunucu login işleminde organizasyonun güvenlik politikasının gerektirdiği karmaşıklığa sahip bir şifre üretilmeli ve şifreleri oluşturmada temel sözlük kelimelerinden sakınılmalıdır.

   Ayrıca, çalıntı hesapla veritabanına yazma yetkisini ele geçirmekte kötü sonuçlar doğurabilir.
5. SQL İnjeksiyon saldırısıyla uygulamaların sahiplerine ait erişim haklarından yararlanılması.

5.2.1. Ara bellek taşması (Buffer overflow)

Statik değişkenler, program hafızaya yüklenirken, programın veri kısmına yerleştirilirler. Dinamik değişkenler ise program çalışırken, dinamik olarak stack (yığın) denen hafızada program için hazırlanmış özel bölümde yaratılıp, yok edilirler. Ara bellek taşması da bu dinamik değişkenlerin taşıyabilecekleri veri miktarından fazlasını yükleyerek değişkenin sınırlarını aşması olayıdır. Örneğin, 5 byte veri taşıyabilecek bir diziye 10 byte veri kopyalamak, bu belleğin taşmasına neden olmaktadır.

Örnek:

```
main(){
char array[8];
Strcpy (array,"bu string uzun olduğundan ara bellek taşmasına yol açacaktır");
return 0;}
```

Bu durumda programın çalışması sırasında ayrılan hafıza boyutu kadar veri işlem görür, diğerleri ise yasak bölge diye nitelenen yığın bölgesine yerleşmek zorunda kalır. Bu işlem bazı durumlarda programın çökmesine ya da çalışmanın durdurulmasına neden olur ve geçersiz işlem yürütüldü uyarısı verilmesine neden olarak açık oluşmasına yol açar. Bazı programlarda ise herhangi bir değişim

olmaksızın çalışmaya devam edilir. Yapılan bu işlemlerde en önemlisi de, yığın denen hafıza alanına taşan veri kötü amaçlar için yapılandırılırsa bu durumda saldırganın sistem üzerinde istediği işlemi yapmasını sağlayabilir. Bu işlemler sonucunda bu alanda virüs işlemleri bile oluşturulabilir. Aslında ağ güvenliği dışında başka bir açıdan bu olaya bakılınırsa bu tip yazılım hatalarına bug denilmektedir. [1]

Ara bellek taşmasının başlıca nedenleri arasında önde geleni programlama dillerinin bellek ve dizi ya da veri boyutunu kontrol etmemesinden kaynaklanır. Örneğin, C ve C++ hiçbir sınır kontrolüne sahip değildir. Sınır kontrolü bunlar gibi diğer bazı dillerde de programcılara bırakılmıştır. Fakat programcılarda bu gibi detaylara fazla dikkat göstermeyip, uygulamaların kullanımında ara bellek taşması açığına meydan bırakmaktadır. C'de kullanılan Sprintf() fonksiyonu bu sınır kontrolü yaparak bu tür açıklara engel olurken daha çok kullanılan Strcpy gibi fonksiyonlar ise tam tersi durum olarak bu tür ara bellek taşması açıklarına davetiye çıkarmaktadır. Bunlar ağ üzerinde fazlaca trafik oluşturarak, iletimin çok ağırlaşmasına neden olmaktadır. [11]

Ara bellek taşmasının nedeni, sınır kontrolünün yapılmamasıdır. Bunu önleme yöntemi de sınırları kontrol eden güvenli kodlama yapmaktır. Sınır kontrolü belleğe erişen fonksiyonlarda ve kodu değiştirebilen derleyicilerle sağlanabilir. Sınır kontrolü derleme anında olursa, program olası zayıflıklardan haberdar olur ve kodu bu yönde değiştirebilir. Bu durumda bellek taşması gibi durumlara sebebiyet bırakılmaz. [50]

```
strcpy(smallbuff, ptr); // kontrolsüz kodlama
strncpy(smallbuff, sizeof(smallbuff), ptr); // güvenli kodlama
```

Modern programlama dilleri bu probleme karşı önlemlerini almıştır. Örneğin Java dilinde dizilerin boyutlanması otomatik olmaktadır. C ve C++ programlama dili bu tür problemlere karşı oldukça zayıftır.

5.2.2. SQL enjeksiyonu

Veritabanında sorgulama yapmasına ve veritabanına veri girişine izin verilen herhangi bir kullanıcı sistemin içinde kabul edilir. Veritabanı uygulamalarında

veritabanı genelde sunucu bilgisayar üzerinde bulunmakta olup, istemci bilgisayarlardan veri girişi veya sorgulaması yapılabilmektedir.

İnternet kullanıcılarına hizmet vermek üzere kullanılan sunucu tabanlı uygulamalardan ASP (Active Server Page) ve PhP en yaygın olanlarıdır. ASP ve PHP sunucu tabanlı uygulamalarda veri saklama ve sorgulaması için SQL (Structured Query Language) kullanılmaktadır. Microsoft Internet Explorer, Opera, Netscape vb. bir tarayıcı aracılığıyla veritabanlarına veri girilebilir veya veri tabanlarından veri sorgulaması yapılabilir.

ASP ve PHP gibi sunucu tabanlı uygulamalar ile veritabanlarına veri girişi ve veri sorgulaması yapılması kolaylıklar sağlarken bazı riskleri de beraberinde getirir. Tasarım aşamasında uygulamada oluşan güvenlik açıkları, uygulamanın kullanılması esnasında önemli sorunlara yol açabilmektedir. Bu açıklar kredi kartı bilgileri, kişisel bilgiler gibi önemli ve hassas bilgilerin çalınmasına neden olmaktadır.

Uygulamanın çalışması esnasında kullanıcıdan veri sorgulama veya girişi için alınan parametre değerlerinin yerine, bazı SQL deyimlerinin yazılmasıyla ortaya çıkan bu açıklarla veritabanı sunucusu ve veritabanı üzerindeki bütün bilgilere erişim ve değişiklik yapma hakkı elde edilebilir. Bu problemden korunmak için girilen parametre değerlerini özel bazı karakterlere karşı bir fonksiyon veya benzer bir şekilde kontrol edilmesi gibi güvenlik denetimiyle olabilir. Şimdi bazı örneklerle SQL enjeksiyonunun nasıl yapıldığını görelim.

Örneklerde hedef veritabanı olarak Microsoft SQL Server'a Microsoft Internet Information Server üzerinden ASP tabanlı bir kullanıcı olarak bağlanılsın. Uygulamaya ait siteye giriş yapılmasına olanak sağlayan 2 adet form olsun.

giris.asp : Kullanıcı adı (Username) ve şifre (password) parametrelerini alan form.
girisKontrol.asp : Veritabanı ile bağlantıya geçen ve girilen değerlerin doğruluğunu kontrol eden ASP sayfası.

Kötü amaçlı (‘) imleçlerin yardımıyla izinsiz giriş sağlama:

Kullanıcı “Username” ve “Password” verilerini giris.asp formuna girip gönderdikten sonra girisKontrol.asp sayfası veritabanı ile bağlantı kurup ilgili tabloda Username ve Password sütunlarında gönderilen kullanıcı adı ve şifresi ikilisinin örtüşüp örtüşmediğinin kontrolünü yapar. Bu işlem sonucunda eğer sonuç olumluysa kullanıcıya; “Giriş Başarılı”, olumsuzsa; “Giriş Başarısız” mesajı verilecektir.

Username: Ömer
Password : 2007

şeklinde kullanıcı tarafından girilen bilgi veritabanında aşağıdaki SQL sorgusunu oluşturacaktır;

```
SELECT Count(*) FROM Users WHERE Username = 'Ömer' AND Password ='2007'
```

Sade ve hatasız bir SQL cümlesine benzese de saldırgan aynı SQL sorgusuna bilgileri şu şekilde gönderirse;

Username: Ömer
Password : ‘ OR 1=1--

Bu durumda oluşacak SQL cümlesi ise;

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

olur. Bu durumda girişin sağlanması için şart “Ömer” kullanıcı adına ait şifrenin hiçbir şey olması veya 1=1 eşitliğinin sağlanması yeterli olacaktır.
Sonuç: Sorgu çalıştığında 1=1 eşitliği sağlandığı için saldırı başarıyla sonuçlanacak ve “Giriş Başarılı” mesajı verilecektir.

Microsoft SQL Server “--“ imlecinden sonra gelen yersiz kullanılmış tırnak işaretlerini önemsememektedir. Basit bir yöntem olarak görülen ve sadece SQL Server’a ait olan bu özellik saldırganlara önemli avantajlar sağlamaktadır.

Uzaktan çalıştırılabilir yordamlar (prosedür):

SQL Server’a varsayılan sistem hesabından yapılan bağlantı SQL enjeksiyon saldırısında muhtemel saldırgana sunucuda saklanan yordamları (prosedür)

çalıştırabilmesi için gerekli hakları tanıyacaktır. Saldırganın kullanabileceği yordamlardan (prosedür) bir tanesi; "master..xp_cmdshell" dir.

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

giriş değerleriyle meydana gelecek SQL Cümlesi;

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

Sonuç: SQL Server, username ve password bulunduran sütunları arayacaktır bulamadığı için "Hatalı Username & Password" mesajını verecektir fakat bu arada arka planda "dir c:" komutunu çalıştıracak ve saldırgan C sürücüsünün içeriğine ulaşmış olacaktır.

SQL Server hedef alınarak yapılan saldırılar:

Yönetici haklarına sahip saldırgan DELETE, INSERT, UPDATE 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 cümlesi;

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

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

ODBC açıklarını kullanarak yapılan saldırılar:

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

Saldırılan site; http://WebSitem/Anasayfa.asp?id=53 şeklinde ürün ID leri ile çalışan ASP tabanlı bir web sitesi olsun. Saldırı, SQL Server'ın integer ve string cinsinden verileri birlikte gönderememesinden faydalanılarak yapılır. Gönderilen '53' sayısına herhangi bir SQL stringi eklenir.

http://WebSitem/Anasayfa.asp?id=53 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 cümlesi;

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. /Anasayfa.asp, line 5

Hata, saldırgana "Table1" olarak bulduğu cevabı integer'a çeviremediğini (dolayısıyla veritabanındaki ilk tablo adının "Table1" olduğunu) belirtmektedir. Saldırgan diğer tabloların adını da bu şekilde kolaylıkla öğrenebilir. Tablo isimlerini öğrendikten sonra tablo içindeki bilgilere örneğin kullanıcı adı şifre bilgilerine erişmesi de yine aynı yöntemleri kullanarak çok kolaydır.

Veritabanına ekleme yapma veya veri düzenleme:

Kullanıcı adı ve şifre bilgisine ulaşan saldırgan benzer yöntemleri ve UPDATE, INSERT komutlarını kullanarak şifreyi değiştirebilir veya daha ötesi başka bir kullanıcı hesabı açabilir.

http://WebSitem/Anasayfa.asp?id=10; UPDATE Admin_Login SET Password = 'NewPwd' WHERE Username='Ömer--

5.2.2.1. SQL enjeksiyonundan korunma yolları

Gelebilecek SQL enjeksiyon saldırılarından korunabilmek için alınan önlemlerde temel alınması gereken nokta "Aksi doğrulanıncaya kadar tüm kullanıcı girdileri kötüdür!" düşüncesine göre hareket etmektir.

a) Kullanıcı haklarının sınırlandırılması:

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

b) Girdilerde tırnak imleçlerinin (') 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(strKelimeler)
    ReplaceQuotes = Replace(strKelimeler,"'","''")
    End Function %>
```

c) 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.

d) Girdi uzunluğunun sınırlandırılması:

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

e) Girdi cinsinin kontrol edilmesi:

Forma girilen verinin istenilen türden bir veri olup olmadığını kontrol eden bir fonksiyon kötü amaçlı kullanımlarda saldırganın kullanabileceği harf/sayı seçeneğini kısıtlayacaktır.

f) Kullanılan veri gönderme metodu:

Form aracılığı ile toplanılan verileri yollarken "POST" metodunu kullanmak gönderilen verinin adres çubuğunda görülmemesini sağlar.

g) Dinamik SQL ifadeleri kullanmaktan kaçınmak:

SQL enjeksiyon sorunları dinamik SQL ifadelerinin kullanılmasından yani kullanıcıdan gelen veriler kullanılarak oluşturulan SQL ifadelerinin çalıştırılmasından kaynaklanmaktadır. Kayıtlı yordam (Stored procedure) veya parameterized SQL kullanımı SQL injection konusunda çok ciddi bir önlem olacaktır. Parameterized SQL doğru bir önlem olacaktır ancak mümkün olan her durumda kayıtlı yordamlar (Stored procedure) en doğru seçim olacaktır.

h) Hatalar (Exceptionlar) mümkün olduğunca az bilgi içermeli:

Tahmin edilebilen veya edilemeyen tüm hatalar için önceden uyarı mesajları veya yapılacak işlemler belirlenmelidir. Kullanıcıya gösterilecek hata mesajı mümkün olduğunca az bilgi içermelidir.

6. DAHA İYİ BİR VERİTABANI GÜVENLİĞİ İÇİN NELER YAPILMALI

Her bir veritabanı güvenliği kendine özgü yapılara sahiptir. Veritabanlarında ortak olarak dikkat edilmesi gereken özellikler şunlardır.

- İyi bir doğrulama yöntemi seçilmelidir.
- Veritabanlarının bulunduğu sunucu üzerinde fiziki güvenlik sağlanmalıdır.
- Veritabanı kurulumundan sonra veri dosyaları, sistem dosyaları ve dizinler için gerekli izinlerin (Unix'te 0750 gibi) yapılması gerekir. Unix ortamında umask değerini 022 ayarlamak gerekir. Bu durumda kullanıcılardan yazma izinleri alınmış olur, sadece okuma iznine sahip olurlar.
- Kurulum dosyaları silinmelidir. Kurulum dosyaları konfigürasyon bilgileri içerebilir.
- Veritabanı ve sunucu seviyesinde, kullanıcılara verilen rollerin nasıl verildiğini inceleyip, hakların gerekli olan minumum seviyede olmasına dikkat etmek gerekir. Kullanıcılara işlerini yapabilecek minumum haklar verilmelidir. Haklar izlenmeli ve sürekli kontrol edilmelidir. ANY içeren haklar sınırlandırılmalı veya tamamen kaldırılmalıdır. [27]
- Veritabanı yöneticisi (DBA) rolüne sahip kullanıcılar ve WITH ADMIN haklarına sahip kullanıcılar sınırlandırılmalıdır.
- PUBLIC için kullanılabilir olan yordamları (procedure) sınırlandırmak ve haklarını kontrol etmek gerekir.
- Uygulama kodu içerisinden veritabanına bağlantı sırasında yönetici yetkisi veya gelişmiş haklara sahip bir yetki kullanılmamalıdır. Uygulamanın veri tabanına bağlanması için minimum yetkilere sahip bir kullanıcı hesabı veya rol oluşturulmalıdır.
- Kurulumdan sonra, varsa, örnek veritabanları ve örnek kullanıcıları kaldırmak gerekir.

- Tüm şifreleri inceleyip basit ve boş şifrelerin meydana getirebileceği güvenlik zayıflığına karşı önlem almak gerekir. Sa, SYS, SYSADM ve SYSTEM vb. kullanıcıların şifreleri kontrol edilmelidir. Şifre politikaları kullanılmalıdır.
- Kullanılmayan network kütüphaneleri kaldırılmalıdır. Çoğu ortamlar TCP/IP üzerinde olup diğerleri kullanılmıyorsa kaldırılmalıdır. [5]
- Gerekli olmayan portları kapatılmalıdır.
- Varsa, kullanılmayan default hesaplar kaldırılmalıdır.
- Denetim özelliği iyi bir şekilde yapılandırıp kullanıcıların hareketleri izlenmeli bilhassa başarısız giriş denemeleri dikkatlice incelenmelidir.
- Sunucu erişiminde kullanılmak üzere her bir veritabanı yöneticisi için bir hesap yaratılmalıdır. Tüm veritabanı yöneticileri sunucuya aynı kullanıcı adını kullanarak bağlanmamalıdır. [42]
- Veritabanını kurarken, sisteme kurulan paketleri ve özellikleri anlamak gerekir. Sadece gerekli olan bileşenleri kurmak gerekir.
- İyi bir veritabanı yedekleme planı uygulanmalıdır.
- Ağ güvenliği sağlanmalıdır. Güvenlik duvarı, saldırı tespit sistemleri (IDS) ve tuzak sistemler oluşturulmalıdır.
- İşletim sistemi güvenliğinin arttırılması gerekir. Gerekli yamalar ve önerilen güvenlik konfigürasyonları yapılmalıdır. Gereksiz bileşenler kaldırılmalı, anti-virus yazılımları kurulmalı, çeşitli erişim kısıtlamaları getirilmelidir.
- Uygulama tasarımı ve programlanması sırasında güvenlik ilkeleri ve oluşabilecek problemler dikkate alınmalıdır (Girilen veri, veri tipine uyumluluk gösteriyor mu? Girilen veri limitler dâhilinde mi? Gerekli hata denetimi fonksiyonlarıyla program çıktısı kontrol ediliyor mu? vb.).
- Bir antivirüs yazılımı kurulup sürekli güncel tutulmalıdır.

6.1. SQL Server Veritabanı İçin Yapılması Gerekenler

- SQL Server da her iki doğrulama tipi de güvenilir olmasına rağmen karışık doğrulama tipine oranla Windows doğrulama tipini kullanmak daha iyi güvenlik sağlamaktadır.
- Windows kayıtlı yordamlarının (Windows extended stored procedure) sınırlandırılması veya kaldırılması gerekir.

- Kullanıcı tarafından oluşturulmuş yordamlar (extended procedure) kurulmamalıdır.
- MS DTC (Microsoft Distributed Transaction Coordinator) bir işlem ile birden fazla farklı kaynağın üzerinde işlem yapılmasını sağlar. Mesela bir işlem ile tüm sunucular üzerinde kalıcı bir güncelleme işlemi yapabilir veya yapılmış bu işlemi hepsinden geri alabilir. Eğer uygulama gerektirmiyorsa MTDC'yi etkisiz hale getirmek gerekir. [23]

6.2. Oracle Veritabanı İçin Yapılması Gerekenler

- UNIX ortamında, Oracle root olarak kurulmamalıdır.
- init.ora için izinler sınırlandırılmalıdır. init.ora, Oracle için gerekli parametrelerin başlatılması için gerek duyulan dosyadır.
- Sunuculardaki iSQL*Plus özelliğini etkisiz hale getirmek gerekir. iSQL*Plus, SQL*Plus'ın web arayüzden çalışan şeklidir.
- ALTER SESSION, ALTER SYSTEM ve BECOME USER için haklara sınırlama koymalı veya izin verilmemelidir. [26]
- Tüm kullanıcılardan CONNECT ve RESOURCE rollerini kaldırmak gerekir. RESOURCE rolü, CREATE CLUSTER, CREATE INDEXTYPE, CREATE OPERATOR, CREATE PROCEDURE, CREATE SEQUENCE, CREATE TABLE, CREATE TRIGGER, CREATE TYPE haklarını barındırır.
- Listener için bir şifre belirlenmelidir. Listener, istemci ve sunucu arasındaki bağlantıyı kuran bir proxy'dir. Yerelden ve uzaktan yönetilebilir. Oracle'ın 8i ve önceki versiyonlarında listener şifresi, açık veya şifreli olarak listener.ora dosyasında saklanabiliyordu. Listener şifresi için şifre politikası uygulama zorunluluğu yoktu.

6.3. DB2 Veritabanı İçin Yapılması Gerekenler

- CLIENT doğrulama hiçbir zaman kullanılmamalıdır. SERVER_ENCRYPT, DATA_ENCRYPT veya KRB_SERVER_ENCRYPT doğrulamadan birisinin kullanılması gerekir.
- Sistem katalogları üzerindeki haklar geri alınmalıdır.

- Eğer Windows ortamında çalışılıyorsa, tüm normal kullanıcılar DB2USERS grubuna, tüm yöneticiler DB2ADMINS grubuna eklenmelidir. [9]

6.4. Veritabanları İçin Diğer Yapılması Gerekenler

6.4.1. Veritabanı yamalarını yükleme

Yamaları takip etme ve zamanında yükleme, güvenliği garanti etmese de güvenliğin en önemli ve gerekli olan işlemlerinden birisidir. Yama işlemi, zor ve zaman gerektiren bir durumdur. Satıcılar da yamaları çok hızlı yayımlamamaktadır. Örneğin, IBM DB2 7.2 versiyonunda LOAD ve INVOKE komutlarından kaynaklanan bir tampon bellek taşması vardı. Bu açıklar, IBM tarafından 2002 Kasımda kabul edilmişti. Onarılması 2003 Eylülde 10 ay sonra olmuştu. Yazılımları onarmak, test etmek ve yeniden yayımlamak zaman almaktadır. [35]

6.4.2. Güvenlik bildirilerini izlemek

Veritabanı ortamındaki zayıf noktalar ve hangi yamaların mevcut olduğunu bilmek ve bir an öce güvenlik problemlerini gidermek çok önemlidir. Her zaman güvenlik haberlerinin farkında olmak ve zayıflıklar olduğu zaman veritabanı ortamı için gerekli önlemleri almak gerekir.

Çeşitli web siteleri, veritabanı ortamları için güvenlik açıkları, uyarılar, yol göstermeler içermektedir. Büyük veritabanı satıcıları da güvenlik uyarılarını müşterilerine bildirmektedir. Bu sitelerden bazıları:

- www.cert.org
- cve.mitre.org
- www.securityfocus.com
- www.securitytracker.com/search/search.html

Büyük veritabanı satıcılarının da kendi güvenlik duyuru sayfaları bulunmaktadır:

- Oracle Güvenlik uyarıları sayfası,
  http://www.oracle.com/technology/deploy/security/alerts.htm

- SQL Server: SQL Server Güvenlik Merkezi, http://www.microsoft.com/technet/security/prodtech/sQLserver.mspx
- DB2: DB2 destek sayfası, http://www-306.ibm.com/software/data/

Güvenlik uyarıları için kullanıcıların katkıda bulunulabileceği adresler:

- Oracle: http://www.oracle.com/technology/deploy/security/securityemail.html
- SQL Server: http://www.microsoft.com/technet/security/bulletin/notify.mspx
- DB2: http://www-1.ibm.com/support/mysupport/us/en/

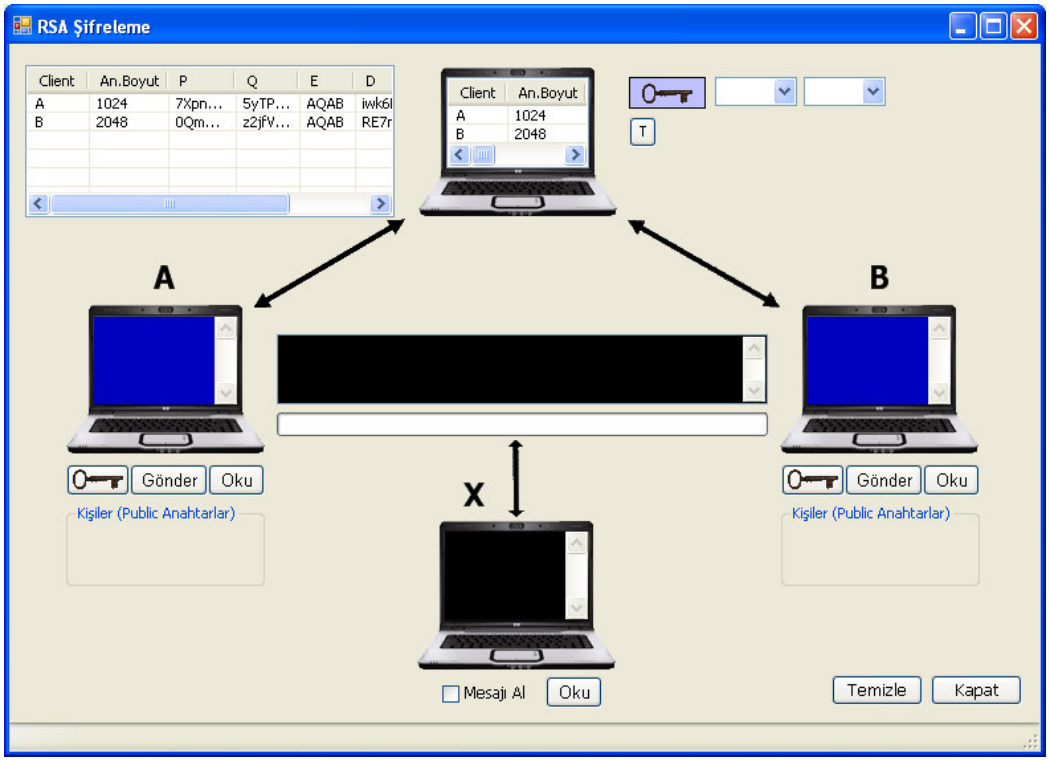
Veritabanlarında keşfedilecek zayıf noktalar aşağıdaki yollarla satıcılara iletilebilir.

- Oracle: secalert_us@oracle.com
- SQL Server: http://www.microsoft.com/technet/security/bulletin/alertus.aspx
- DB2: http://www-306.ibm.com/software/support/probsub.html

7. TEZ KAPSAMINDA YAPILAN YAZILIM UYGULAMALARI

7.1. RSA Şifreleme Uygulaması

RSA veri şifreleme yöntemi yazılımsal olarak gerçekleştirilmiştir. Yazılım uygulamasında C# dili kullanılmıştır. Uygulama, kullanıcının gireceği anahtar boyutuna göre istemcilere açık ve gizli anahtarlar üretmektedir. Bu üretilen açık ve gizli anahtarlar ayrı dosyalarda saklanmaktadır. Program rasgele (random) olarak asal P ve Q değerleri belirleyip daha sonra yapılacak işlemlerde kullanılmak üzere kullanıcıya ait gizli anahtar (E,N) ve açık anahtar (D,N) üretilmektedir. Söz konusu anahtar bilgileri mevcutsa, program çalışmaya başladığında otomatik olarak yüklenmektedir. Şekil 7.1'de A ve B istemcilerinin anahtarları daha önce üretilmiş olduğundan, program çalışmaya başladığında anahtar bilgileri ana sunucuya yüklenmektedir. Her kullanıcının anahtar bilgisi kendisinde ve ana sunucuda yer almaktadır. Başka bir kullanıcıya şifreli mesaj gönderecek olan kişi öncelikle o kişinin açık anahtarını ya direkt olarak o kişiden ya da ana sunucudan edinmesi gerekmektedir. Karşı tarafın açık anahtarını direkt ilgili kişiden almaktansa ana sunucudan almak daha doğru bir hareket olacaktır. Çünkü karşı tarafın yerine sahte kimlikli bir kişi geçebilir ve yanlış açık anahtarla şifrelenen metin yine yanlış gizli anahtarla açılabileceğinden, mesaj gerçek kişi yerine yanlış kişiye gitmiş olacaktır.

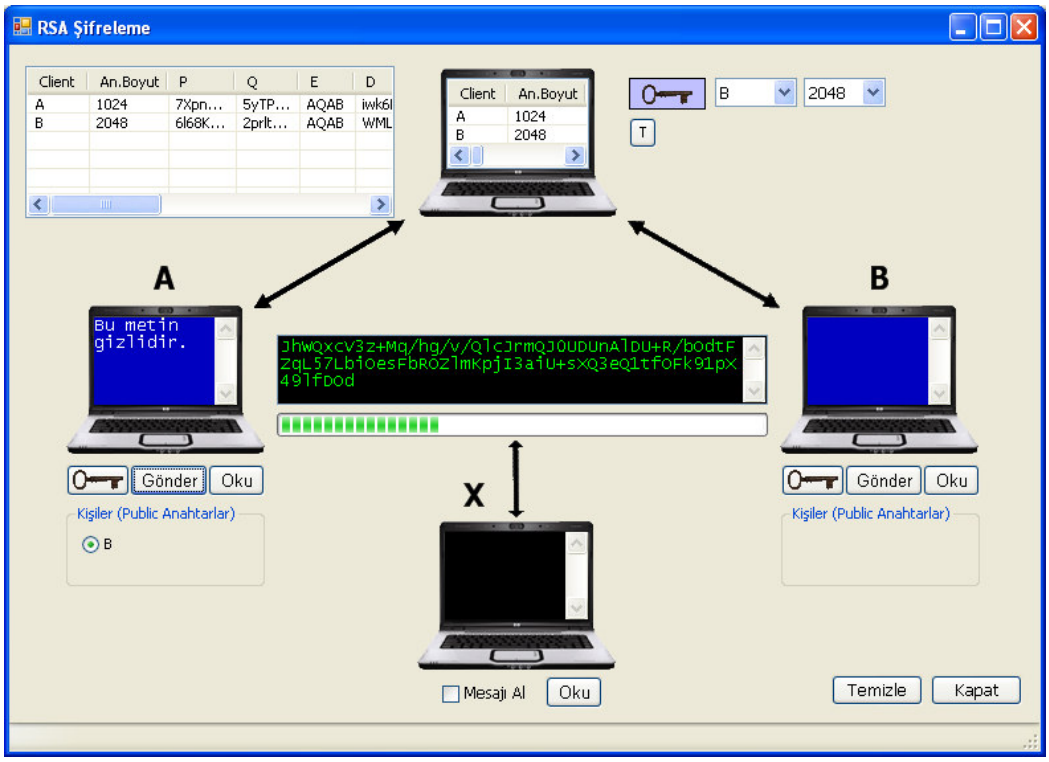


Şekil 7.1: RSA programının çalışma şekli

Mesajın gönderilmek istenen kişinin açık anahtarı edinildikten sonra (örneğimizde B kullanıcısı), mesaj o kişinin açık anahtarıyla şifrelenerek karşı tarafa gönderilir. Şifreleme işlemini gerçekleştiren program kodu:

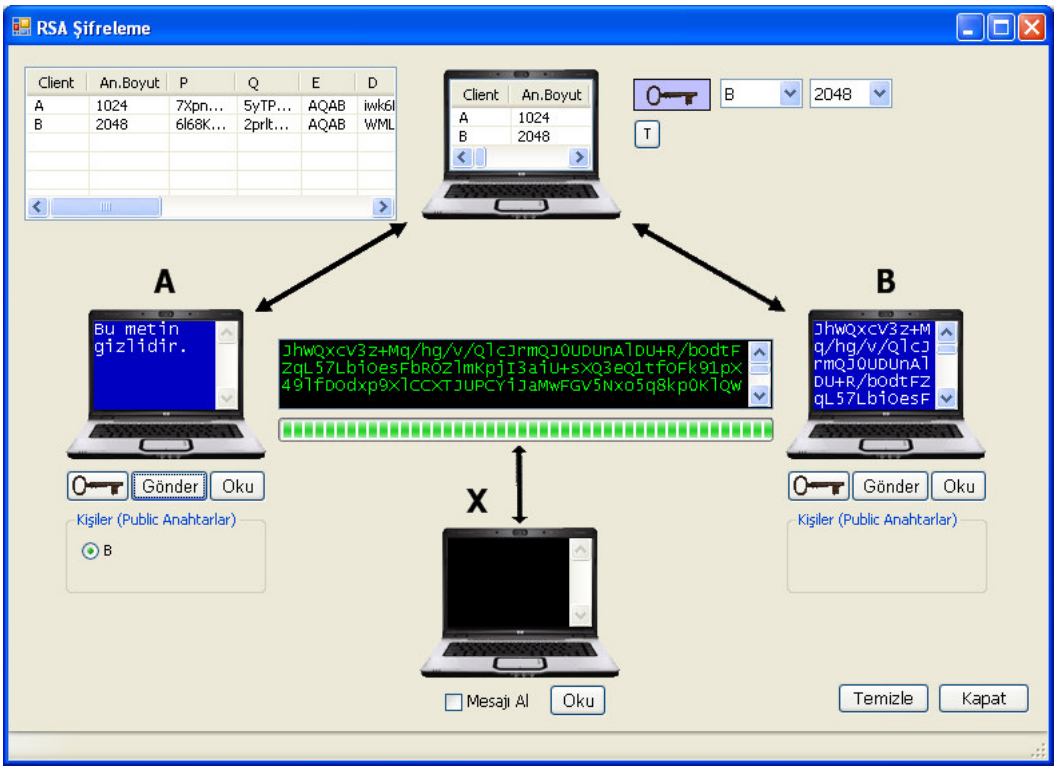
```
public string RSASifrele(string girisString, int dwAnahtarBoyu, string xmlString)
        {
            RSACryptoServiceProvider rsaCryptoServiceProvider = new
RSACryptoServiceProvider(dwAnahtarBoyu);
            rsaCryptoServiceProvider.FromXmlString(xmlString);
            int anahtarBoyu = dwAnahtarBoyu / 8;
            byte[] bytes = Encoding.UTF32.GetBytes(girisString);
            int maxUzunluk = anahtarBoyu - 42;
            int veriUzunluk = bytes.Length;
            int adimSayisi = veriUzunluk / maxUzunluk;
            StringBuilder stringBuilder = new StringBuilder();
            for (int i = 0; i <= adimSayisi; i++)
            {
```

```
            byte[] gecBytes = new byte[(veriUzunluk - maxUzunluk * i > maxUzunluk) ?
maxUzunluk : veriUzunluk - maxUzunluk * i];
            Buffer.BlockCopy(bytes, maxUzunluk * i, gecBytes, 0, gecBytes.Length);
            byte[] encryptedBytes = rsaCryptoServiceProvider.Encrypt(gecBytes, true);
            Array.Reverse(encryptedBytes);
            stringBuilder.Append(Convert.ToBase64String(encryptedBytes));
        }
        return stringBuilder.ToString();
    }
```



Şekil 7.2: RSA programında şifreleme.

Şifrelenen metin iletim hattı üzerinden karşı kullanıcıya şifreli olarak iletilir. Şekil 7.2' de mesajın şifreli olarak B kullanıcısına iletilmesi, Şekil 7.3'te ise iletilmiş hali görülmektedir.

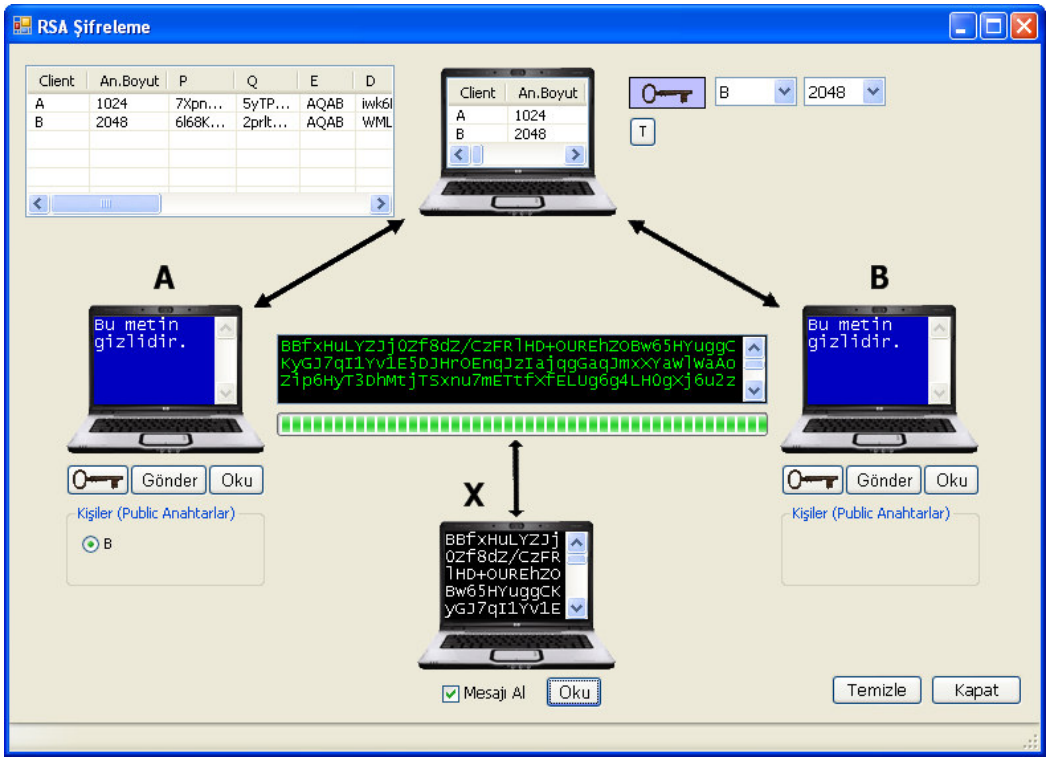


Şekil 7.3: RSA programında şifrelenmiş mesajın iletilmesi.

Şifreli mesajın gerçek sahibi olan B kullanıcısı kendi açık anahtarıyla şifrelenen metni, yine kendi gizli anahtarını kullanarak çözebilir ve içeriğini görüntüleyebilir. Mesajı deşifrelemek için programda kullanılan kod şu şekildedir:

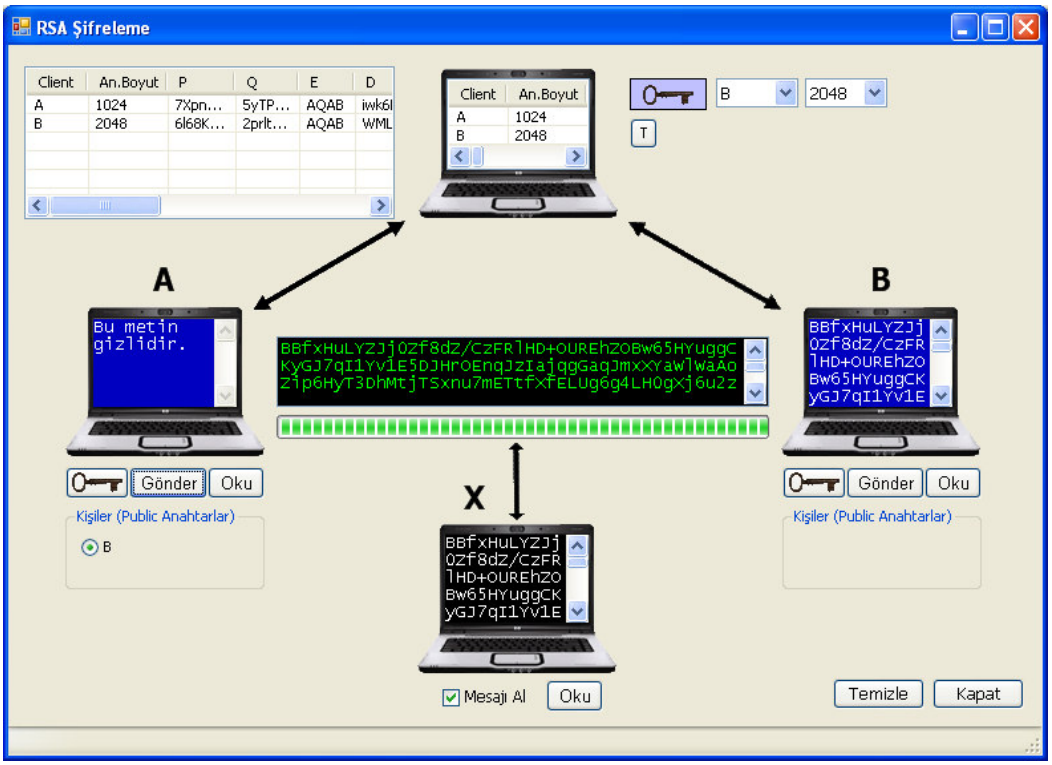
```
public string RSADesifrele(string girisString, int dwAnahtarBoyu, string xmlString)
        {
            RSACryptoServiceProvider rsaCryptoServiceProvider = new
RSACryptoServiceProvider(dwAnahtarBoyu);
            rsaCryptoServiceProvider.FromXmlString(xmlString);
            int base64BlokBoyu = ((dwAnahtarBoyu / 8) % 3 != 0) ? (((dwAnahtarBoyu / 8) / 3)
* 4) + 4 : ((dwAnahtarBoyu / 8) / 3) * 4;
            int adimSayisi = girisString.Length / base64BlokBoyu;
            ArrayList arrayList = new ArrayList();
            for (int i = 0; i < adimSayisi; i++)
            {
               byte[] encryptedBytes =
Convert.FromBase64String(girisString.Substring(base64BlokBoyu * i, base64BlokBoyu));
               Array.Reverse(encryptedBytes);
```

```
            arrayList.AddRange(rsaCryptoServiceProvider.Decrypt(encryptedBytes, true));
        }
        return
Encoding.UTF32.GetString(arrayList.ToArray(Type.GetType("System.Byte")) as byte[]);
    }
```



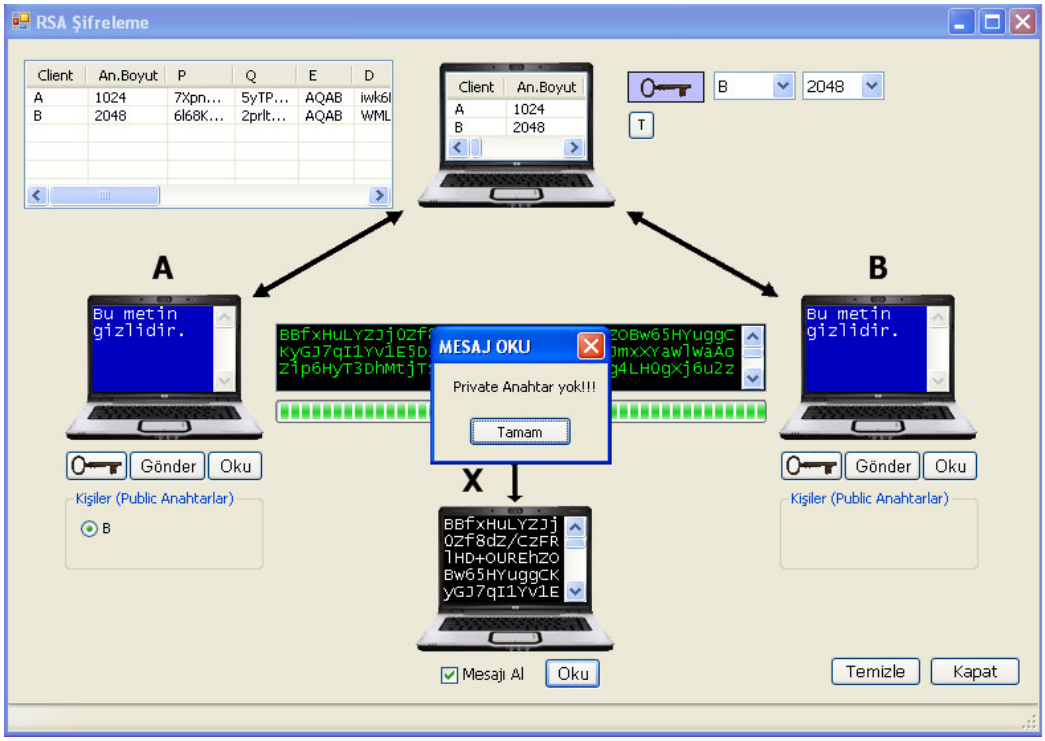
Şekil 7.4: RSA şifrelemede mesajın sahibi tarafından deşifre edilmesi.

Aksi bir durum ise iletim hattı üzerinden araya üçüncü bir kişinin girmesi durumudur. Ara giren kişi şifreli metni ele geçirebilir. Şekil 7.5'te şifreli mesajın üçüncü bir kişi tarafından ele geçirilme durumu görüntülenmektedir.



Şekil 7.5: RSA şifrelemede mesajın farklı kullanıcılarca ele geçirilmesi.

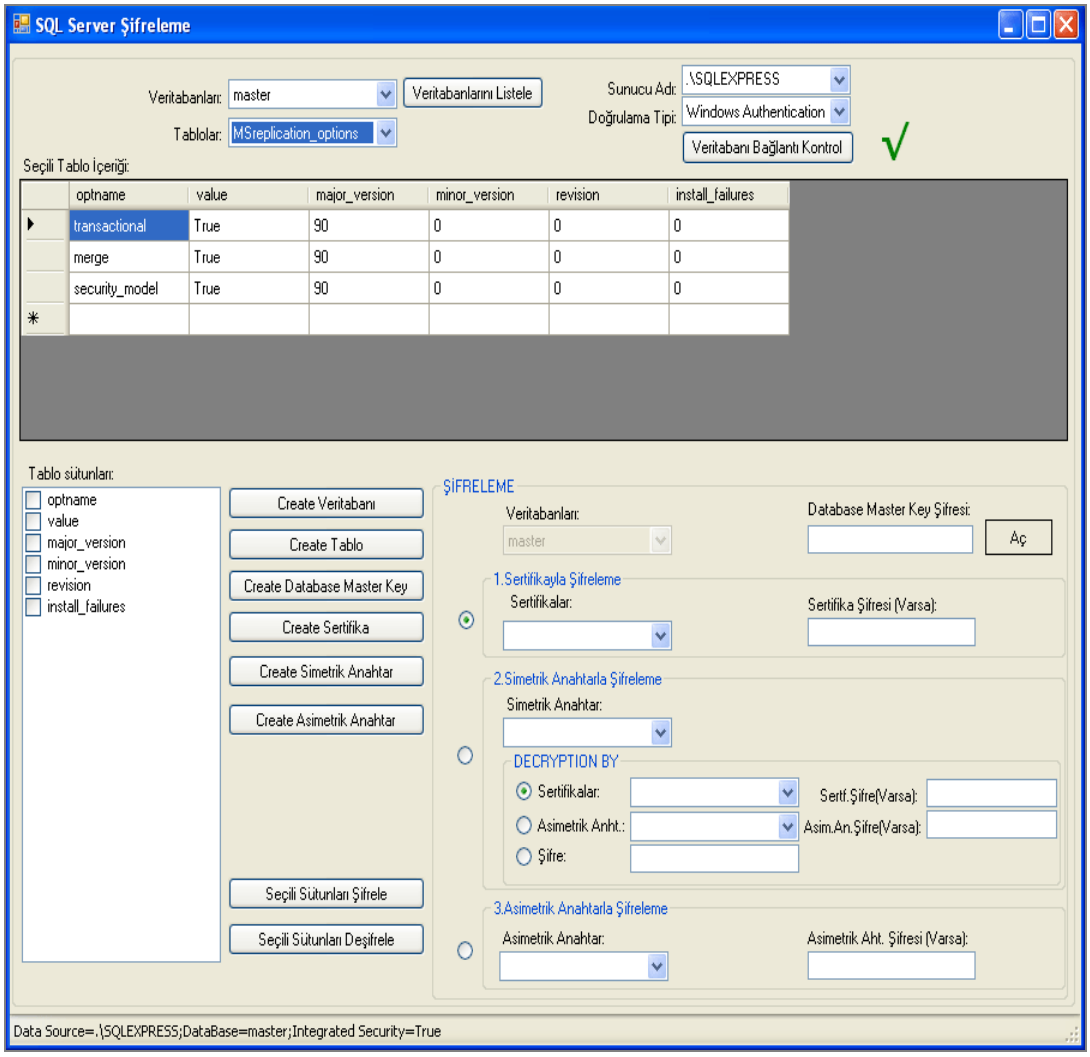
Mesaj, araya giren kişinin gizli anahtarıyla şifrelenmediğinden dolayı, mesajı ele geçiren üçüncü bir kişi (örneğimizde X kullanıcısı) mesajı deşifre edemeyecektir.



Şekil 7.6: RSA şifrelemede mesajın farklı kullanıcılarca deşifre edilmesi.

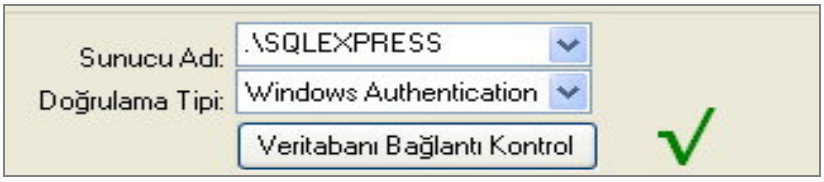
7.2. SQL Server Veritabanı Şifreleme Uygulaması

Yazılım uygulamasında C# dili kullanılmış olup, SQL Server veritabanı sisteminde istenen herhangi bir veritabanına ait tablolar üzerinde veri şifreleme ve deşifreleme işlemlerini gerçekleştirebilmektedir. Şifreleme işleminde sütun bazlı şifreleme yapılabilmektedir. Aynı anda bir veya birden fazla sütunun şifrelenmesine veya deşifrelenmesine imkân tanımaktadır. Programın ana ekran görüntüsü Şekil 7.7 deki gibidir.



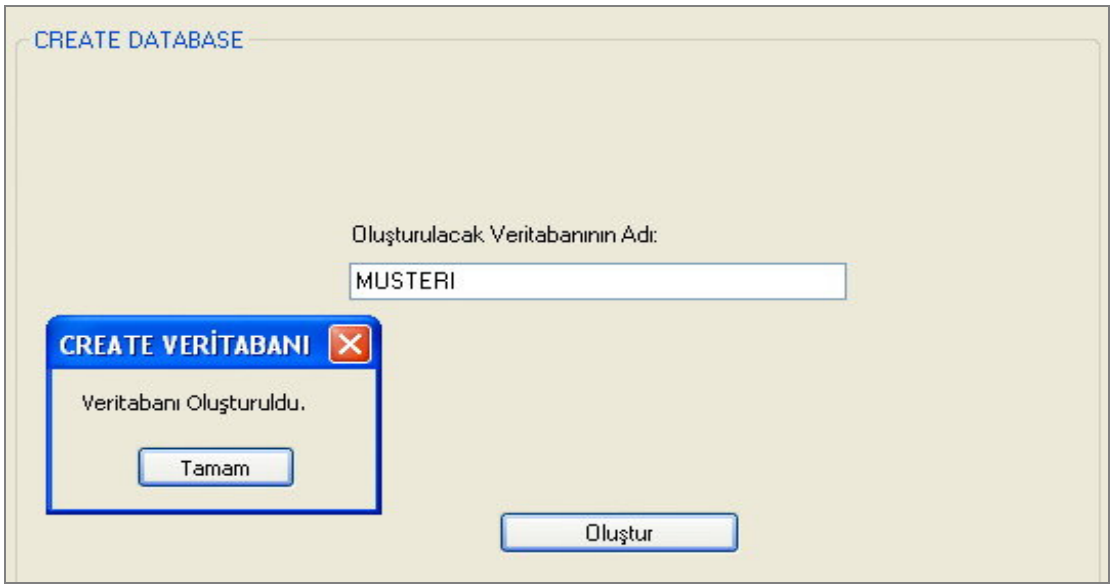
Şekil 7.7: SQL Server Veritabanı Şifreleme Programı.

Program çalıştığında, bağlanılmak istenen veritabanı sunucusu ve kullanılacak doğrulama tipi istenmektedir.



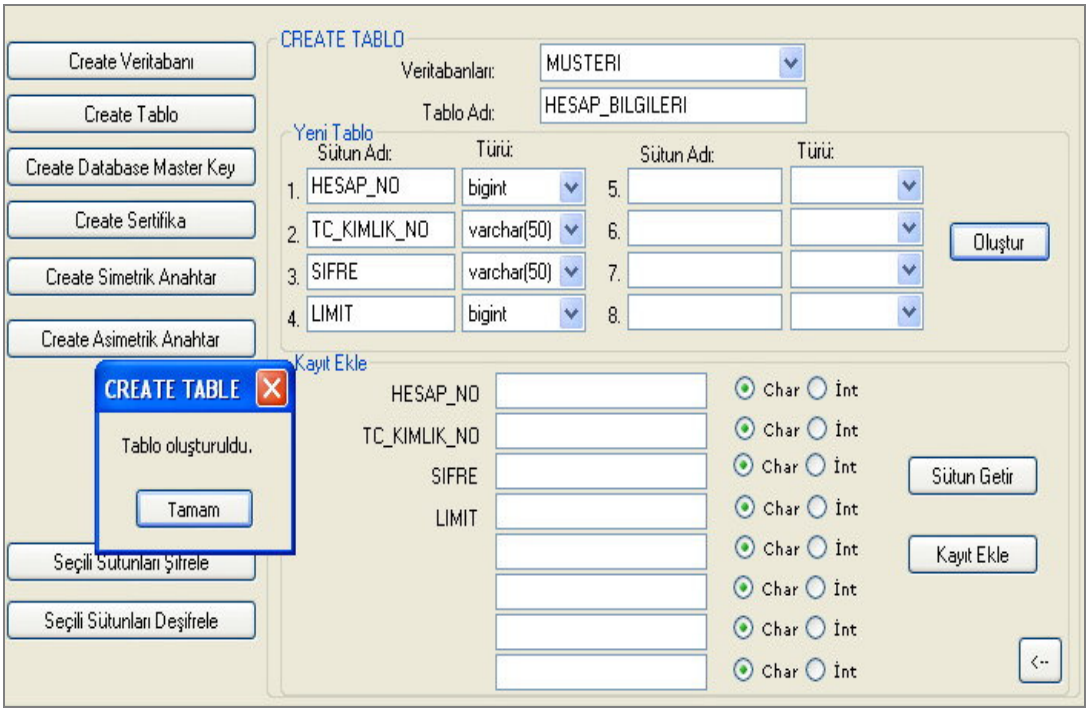
Şekil 7.8: SQL Server Veritabanı doğrulama yöntemi.

Veritabanı sunucusu için gerekli bilgiler seçilip bağlantı sağlandıktan sonra, programda o sunucuda yer alan veritabanları listelenebilir. Program ayrıca, mevcut veritabanları üzerinde değil de yeni bir veritabanı üzerinde işlem yapmak için yeni bir veritabanı ve içerisinde tablolar oluşturma olanağı sağlamaktadır. Create Veritabanı butonuna tıklanarak yeni veritabanı oluşturma ekranı açılır. Yeni veritabanı için daha önce kullanılmamış yani o sunucuda yer almayan bir veritabanı adı belirlenir. Örneğimizde MUSTERI adında yeni bir veritabanı oluşturulmaktadır.



Şekil 7.9: SQL Server veritabanında yeni bir veritabanı oluşturma.

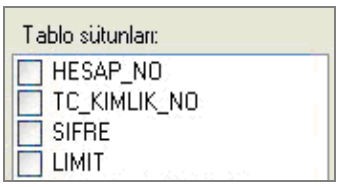
Yeni oluşturulan veritabanı içinde hiçbir tablo veya nesne bulunmaz. Veritabanımızda HESAP_BILGILERI adlı bir tablo oluşturmak için Create Tablo sekmesine tıklayıp tablo adını girelim. Ayrıca tabloda bulunacak sütun adlarını ve veri türlerini belirlemek gerekmektedir. Oluştur butonuna tıklanarak tablo oluşturulur.



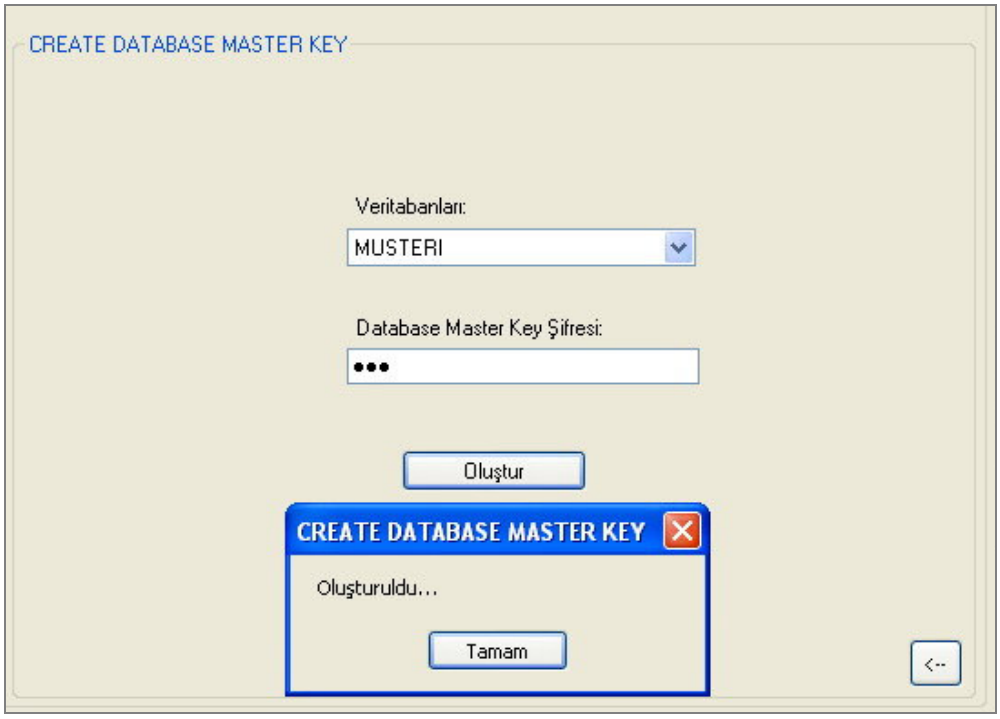
Şekil 7.10: SQL Server da veritabanı içerisinde yeni bir tablo oluşturma.

Bu aşamadan sonra veritabanı başına bir adet olabilen Database Master Key oluşturmak gerekir. Programın Create Database Master Key butonuna tıklanarak açılan ekrandan Musteri veritabanı seçilerek kullanıcı tarafından Database Master Key için bir şifre belirlenir ve Oluştur butonuna tıklanılır.

Tablolar listesinden seçilen tabloya ait bilgiler ekrana yansır. Ayrıca oluşturulan tablo otomatik olarak aktif yani seçili tablo durumuna geleceğinden tablo sütunları da Şekil 7.11'deki gibi listelenir.

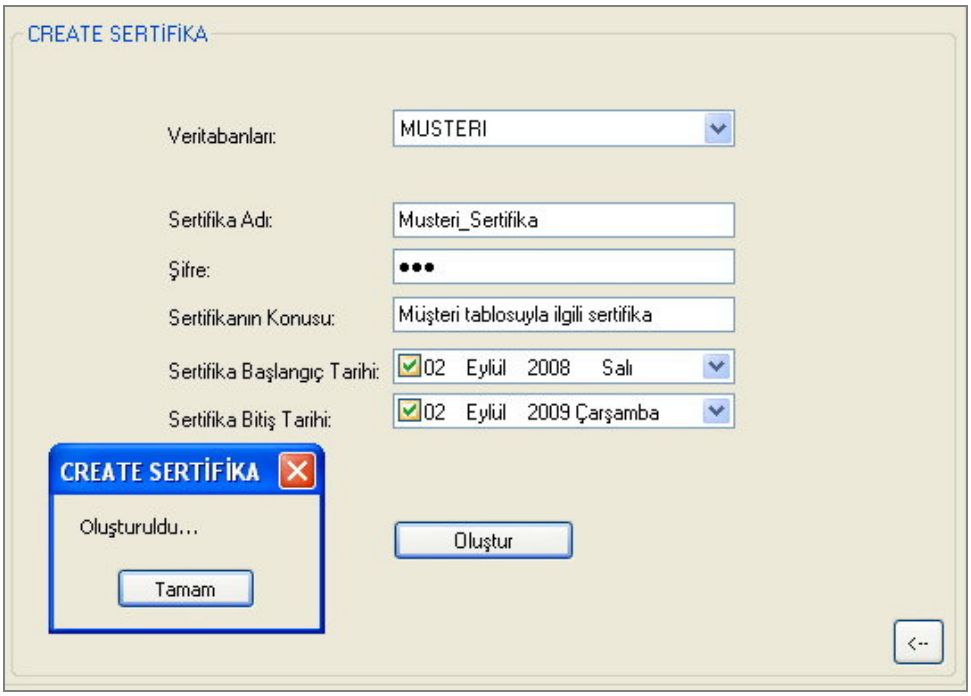


Şekil 7.11: Veritabanında seçili (aktif) tabloya ait sütunlar.



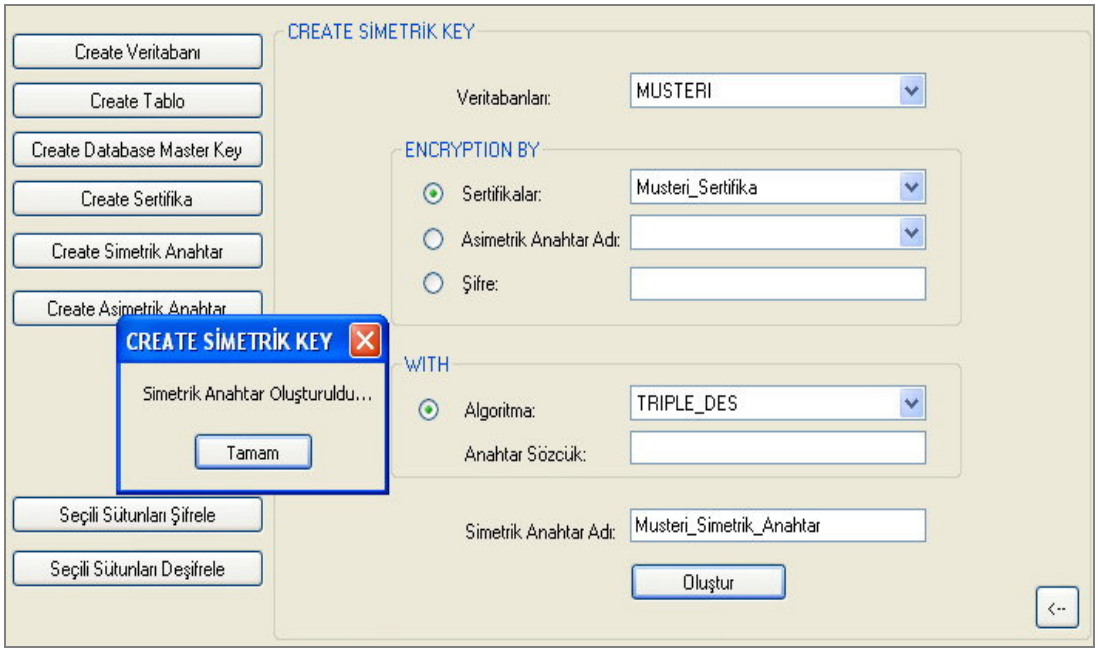
Şekil 7.12: SQL Server da veritabanı master anahtarı oluşturma.

Şifreleme yapmak veya simetrik, asimetrik anahtar oluşturmak için var olan bir sertifikayı kullanmak veya yeni bir sertifika oluşturmak gerekir. Burada yeni bir sertifika oluşturulacaktır. Yeni sertifika için konu, başlangıç ve bitiş tarihi girilmelidir.



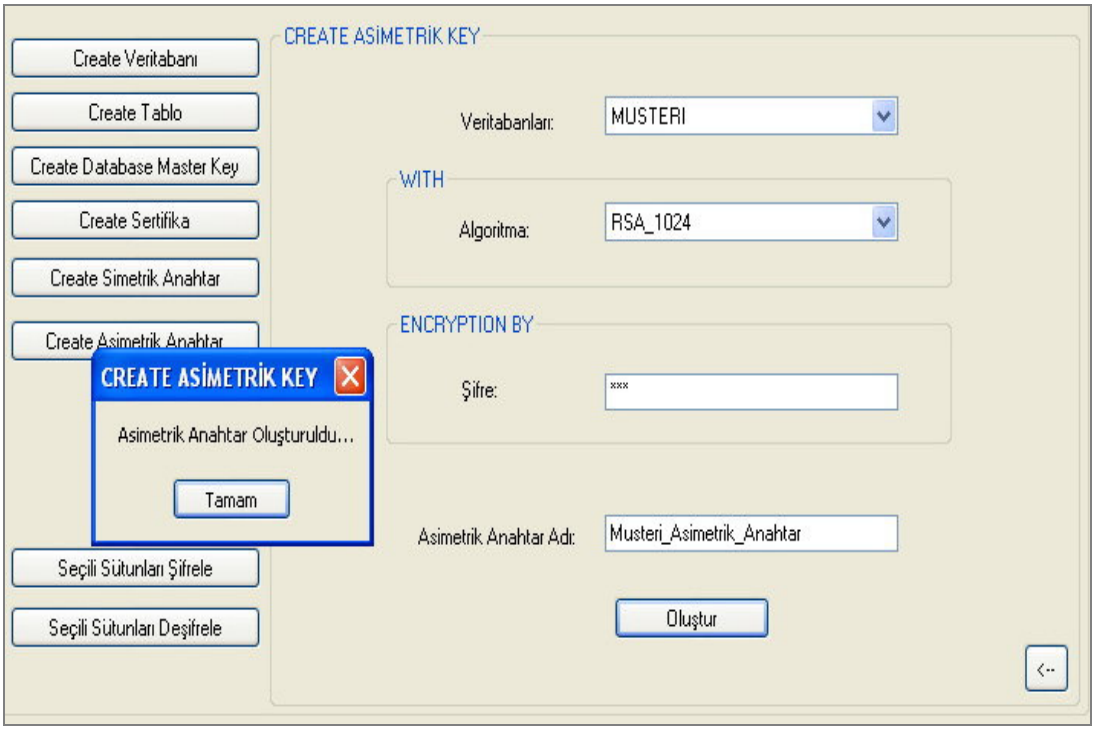
Şekil 7.13: SQL Server da şifrelemede kullanılmak üzere sertifika oluşturma.

Eğer şifrelemede simetrik anahtar kullanılacaksa Create Simetrik Key butonu kullanılarak yeni bir simetrik anahtar üretilmelidir. Simetrik anahtarın bir sertifika veya asimetrik anahtarla mı şifreleneceği belirlenir. Sertifika adı veya asimetrik anahtar adı listeden seçilir. Simetrik anahtar için belirlenecek şifreleme algoritması ise DES, TRIPLE_DES, RC2, RC4, RC4_128, DESX, AES_128, AES_192, AES_256 seçeneklerinden birisi olabilir.



Şekil 7.14: SQL Server simetrik anahtar oluşturma.

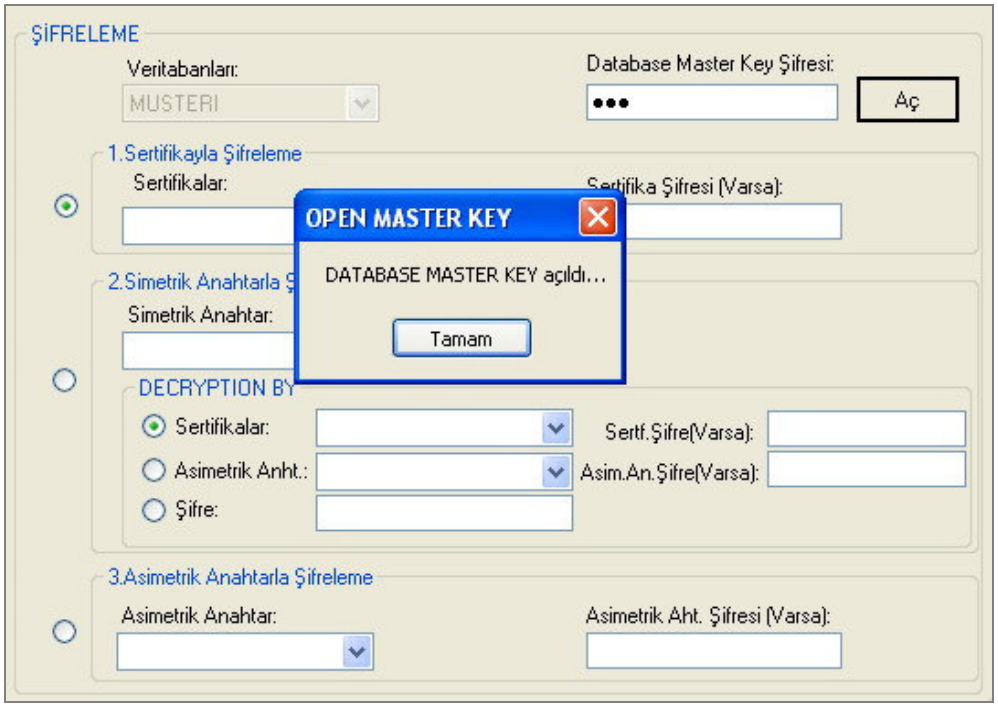
Şifreleme asimetrik anahtar kullanılarak da yapılabilir. Create Asimetrik Key butonu yardımıyla asimetrik anahtar oluşturma ekranı açılıp oluşturulacak asimetrik anahtar için bir algoritma seçilir. Bu algoritma RSA_512, RSA_1024, RSA_2048 seçeneklerinden birisi olabilir.



Şekil 7.15: SQL Server da asimetrik anahtar oluşturma.

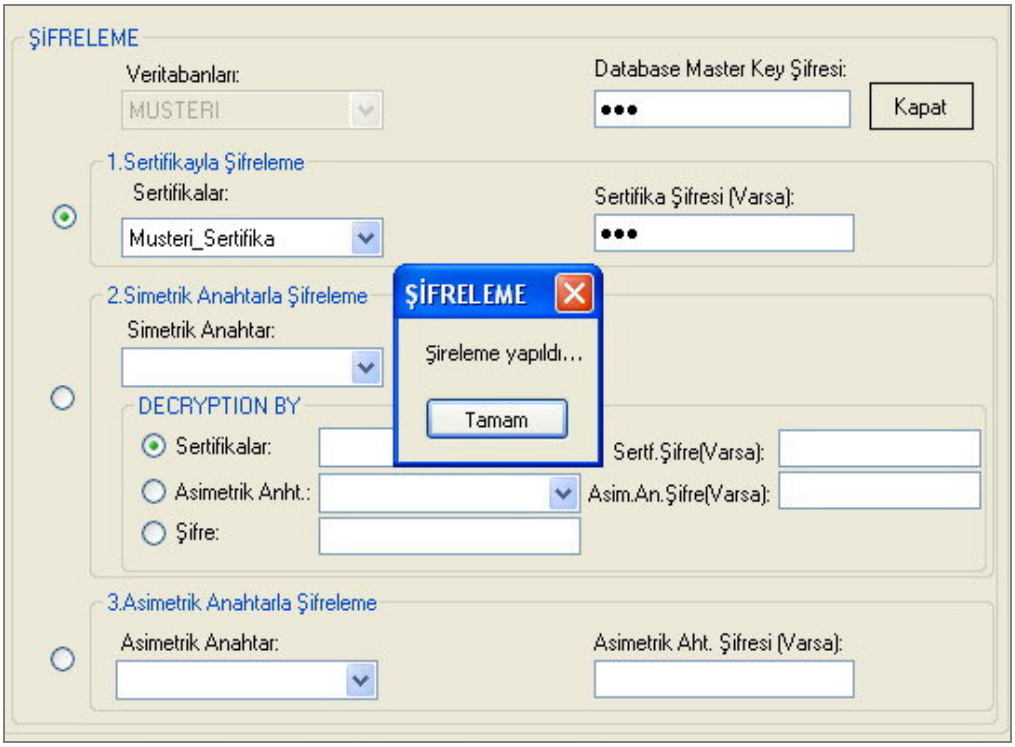
Veritabanı şifreleme için gerekli olan Database Master Key, Sertifika, Simetrik anahtar ve Asimetrik anahtar oluşturulduktan sonra şifreleme işlemi yapılabilir.

Şifreleme işlemi için ilk önce Database Master Key açılmalıdır. Database Master Key için bir şifre belirlenmişse, söz konusu şifre girilip Aç butonuna tıklamak yeterlidir.



Şekil 7.16: SQL Server da veritabanı master anahtarının açılması.

Eğer şifrelemede sertifika kullanılmak isteniyorsa Sertifikalar listesinden sertifika adı ve tablodaki sütun adlarının bulunduğu listeden şifrelenmek istenen sütunların yanındaki onay kutuları seçilir, Seçili Sütunları Şifreleme butonuna tıklanarak şifreleme işlemi gerçekleştirilir.



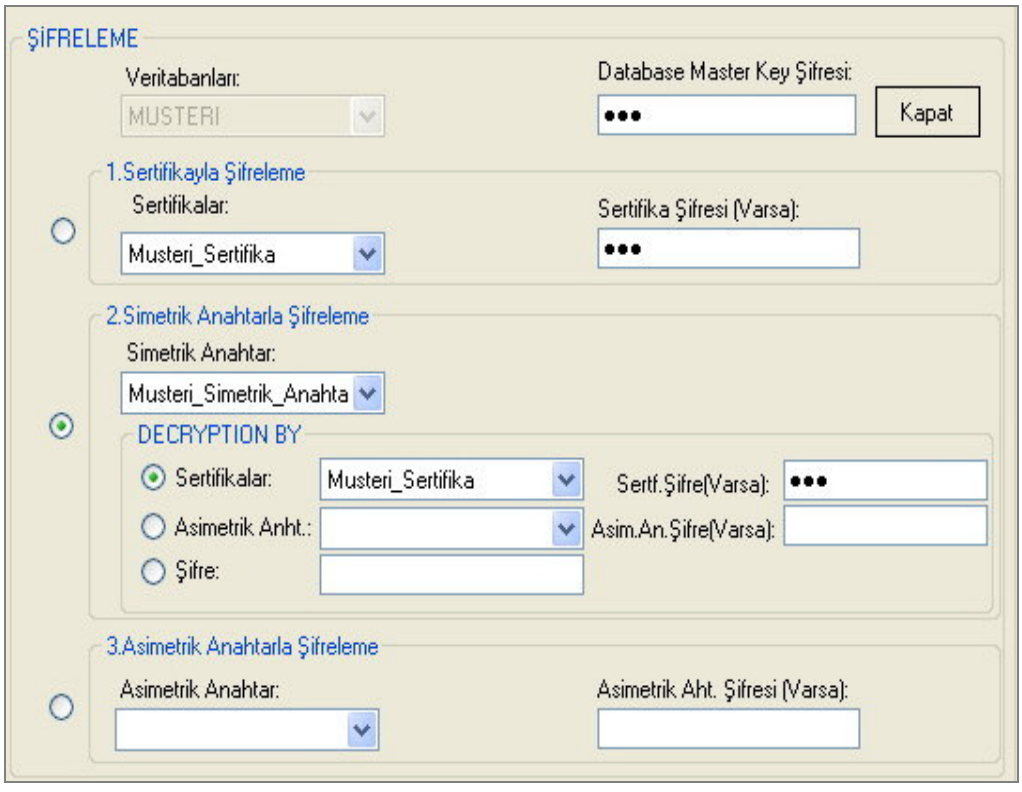
Şekil 7.17: SQL Server veritabanında sertifika kullanılarak şifreleme.

Şifrelemesi yapılan sütunun şifrelenmiş hali yeni bir sütun oluşturularak Şekil 7.18'de olduğu gibi tabloda gösterilir.



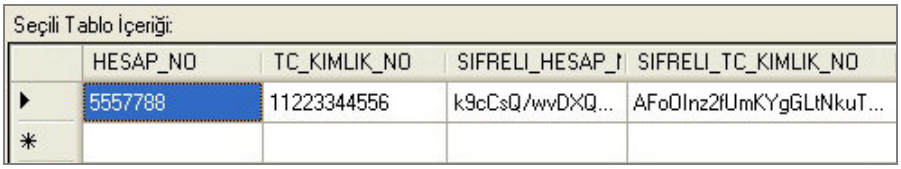
Şekil 7.18: SQL Server veritabanında sertifikayla şifreleme çıktısı.

Şifreleme işlemi için simetrik anahtar kullanılmak isteniyorsa Simetrik Anahtarla Şifreleme alanındaki simetrik anahtarlar listesinden bir simetrik anahtar ve bu simetrik anahtarın kullandığı şifreleme yöntemi seçilir, varsa şifresi girilir. Daha sonra şifrelenmek istenen sütunlar işaretlenir ve Seçili Sütunları Şifreleme butonuyla şifreleme işlemi gerçekleştirilir.



Şekil 7.19: SQL Server veritabanında simetrik anahtar kullanılarak şifreleme.

Simetrik anahtarla şifrelenmiş sütun, kullanıcıya bilgi vermesi amacıyla yeni bir sütun olarak tabloya eklenmektedir.



Şekil 7.20: SQL Server veritabanında simetrik anahtar şifreleme çıktısı.

Şifreleme işlemi için asimetrik anahtar kullanılmak isteniyorsa Asimetrik Anahtarla Şifreleme alanındaki asimetrik anahtarlar listesinden bir simetrik anahtar seçilir ve varsa şifresi girilir. Daha sonra şifrelenmek istenen sütunlar işaretlenir ve Seçili Sütunları Şifreleme butonuyla şifreleme işlemi gerçekleştirilir.



Şekil 7.21: SQL Server veritabanında asimetrik anahtar kullanılarak şifreleme.

Asimetrik anahtarla şifrelenmiş sütun, yeni sütun olarak Şekil 7.22'deki tabloda görülebilmektedir.



Şekil 7.22: SQL Server veritabanında asimetrik anahtar şifreleme çıktısı.

Şifrelenmiş verinin deşifrelenmesinde ise, deşifrelemesi yapılacak sütun için hangi yöntemle şifrelenmişse aynı yöntemle deşifrelenmesi gerekmektedir. Deşifrelenecek sütun işaretlenir, sertifika, simetrik anahtar veya asimetrik anahtarla şifreleme yöntemlerinden hangisi kullanılmışsa o yöntem seçilir ve Seçili Sütunları Deşifrele butonuna tıklanarak deşifreleme işlemi yapılır. Deşifrelenmiş sütuna ait bilgi, karşılaştırma yapılabilmesi amacıyla yeni bir sütun olarak tabloya eklenir.



Şekil 7.23: SQL Server veritabanında deşifrelenmiş sütun.

Program, tüm SQL Server veritabanlarındaki tablolar üzerinde mümkün olan ve SQL Server şifreleme tarafından desteklenen tüm simetrik, asimetrik şifreleme algoritmalarını ve sertifikaları alternatifleri kullanarak şifreleme ve deşifreleme işlemlerini gerçekleştirebilmektedir.

8. SONUÇ

Tez çalışması kapsamında veri güvenliği, veri güvenlik prensipleri, veri güvenlik yönetim modelleri, veritabanlarının temel güvenlik araçları olan kullanıcı tanımlama, doğrulama ve yetkilendirme konuları, veri şifreleme yöntemleri, sertifikalar, uygulama, işletim sistemi, ağ ve internet güvenliği, güvenlik duvarları, Kerberos ve SSL protokolleri incelenmiştir. SQL Server, Oracle ve DB2 veritabanlarının güvenlik yapıları araştırılmıştır. Veritabanı sistemlerinin veri şifreleme teknolojileri, veritabanlarındaki riskler, güvenlik boşlukları incelenmiş ve temel saldırı yöntemlerinden SQL enjeksiyonu örneklerle anlatılmıştır. Daha iyi bir veritabanı güvenliği için SQL Server, Oracle ve DB2 veritabanlarında yapılması gerekenler ortaya konulmuştur. Açık anahtarlı sistemde ve sertifikalarda yaygın olarak kullanılan günümüzün popüler şifreleme algoritması RSA ile ilgili bir uygulama yapılmıştır. Ayrıca çok geniş bir şifreleme teknolojisine sahip olan SQL Server veritabanı ile ilgili her tür sunucuda çalışabilen bir veritabanı şifreleme yazılımı gerçekleştirilmiştir.

Yapılan RSA yazılım programıyla, büyük anahtar değerleri kullanılarak gizli ve açık anahtarlar üretmek, verileri şifrelemek ve açık anahtarların ağ üzerinde paylaşılması konuları bir uygulamayla örneklenmiştir.

SQL Server veritabanında veri şifreleme yöntemleri, veritabanı ve tablolar oluşturma, sertifika, simetrik ve asimetrik anahtarlar üretme ve tüm SQL Server veritabanı sunucularında çalışabilen esnek, etkili, saydam şifreleme sağlayan bir yazılım programıyla da SQL Server veritabanı şifreleme mimarisi örneklenmiştir.

Bu tez çalışması sonucunda, veritabanı güvenliğinin sağlanması için başta veritabanlarının temel güvenlik özellikleri olan kullanıcı doğrulama, yetkilendirme ve kullanıcı denetiminin etkili bir şekilde yapılması, verinin iletim hattı üzerinden şifreli olarak taşınması ve veritabanı içinde şifreli olarak saklanması, verileri

kullanacak uygulamalarda yazılımdan kaynaklanan açıklarının olmaması durumlarının sağlanması gerektiği ortaya çıkmaktadır. Ayrıca, sistemin içerden ve dışarıdan insanlar veya virüsler tarafından gelebilecek saldırılara karşı savunma mekanizmalarına sahip olabilmesi, sistemin fiziki güvenliğinin sağlanması, düzenli yedeğinin alınması, veritabanı yazılımının güncellenmesi, veritabanı sağlayıcılarına ait güvenlik uyarılarının takip edilmesi ve veritabanı sistemi kullanıcılarının bilgilendirilmesi gerekmektedir.

Bu tezde incelenen büyük veritabanlarının güvenlik özelliklerinin karşılaştırmasında Tablo 8.1'deki sonuçlar ortaya çıkmıştır.

Güvenlik Özelliği	**SQL Server**	**Oracle Enterprise (Advanced Security)**	**IBM DB2**
İşletim Sistemi Yoluyla Doğrulama	Var	Var	Var
Veritabanı Doğrulaması	Var	Var	Yok
Single Sign-On	Var	Var	Var
Ağ Paket Şifreleme	Var	Var	Yok
Veri Şifreleme	Var	Var	Var
Açık Anahtar Altyapısı (PKI)	Var	Var	Var
Kerberos Desteği	Var	Var	Var
Şemalar	Var	Var	Var
Veritabanı ve Sunucu Rolleri	Var	Var	Var
Hesap Denetimi (Auditing)	Var	Var	Var
Profiller/ Politikalar	Var	Var	Var
Sertifika Servisleri	Var	Var	Var
En az haklarla çalışma	Var	Var	Var

Tablo 8.1: Veritabanları güvenlik özellikleri karşılaştırması

SQL Server veritabanı kullanıcı doğrulamasını Windows yoluyla ve karışık (mixed) doğrulama yöntemleriyle yapabilmektedir. Oracle veritabanı kullanıcı doğrulaması konusunda en çok yönteme sahip olan veritabanıdır. İşletim sistemi, veritabanı, PKI, Kerberos, RADIUS ve dizin tabanlı servisler kullanılarak doğrulama yapabilmektedir. DB2 veritabanı ise Microsoft SQL Server ve Oracle'ın aksine kendi veritabanı üzerinde doğrulama yapmamaktadır. Doğrulamayı işletim sistemi aracılığıyla yapmaktadır. Aslında sınırlı bir durum gibi görülse de Microsoft ve Oracle tarafından da işletim sistemi tabanlı kullanıcı doğrulamanın kullanılması önerildiğinden, güvenlik açısından sorun teşkil etmemektedir. Çünkü işletim sistemi tabanlı kullanıcı doğrulaması güçlü bir doğrulama sağlar. DB2 doğrulama seçeneklerinden birisi olan CLIENT seçeneği veritabanı güvenliğini tehlikeye atması nedeniyle kullanılmaması gereken zayıf bir doğrulama seçeneğidir.

SQL Server, Oracle ve IBM DB2'nin her üçü de işletim sistemi üzerinden, erişim için kullanıcı kimlik doğrulaması gerektiren birden fazla yapıya, sisteme tek bir giriş noktasından giriş yapma olanağı sağlayan Single Sign On (SSO) özelliğine sahiptir.

Ağ üzerinden verilerin şifreli olarak gönderilmesi SQL Server ve Oracle tarafından SSL yoluyla sağlanmaktadır. IBM DB2 de ise veri paketlerinin şifreli gönderimi mevcut değildir. Ancak, IBM'nin Tivoli yazılımı kullanılarak ağ üzerinden verilerin şifreli olarak gönderilmesi sağlanabilmektedir.

SQL Server şifrelemede RSA_512, RSA_1024, RSA_2048, DES, TRIPLE_DES, RC2, RC4, RC4_128, DESX, AES_128, AES_192, AES_256 gibi şifreleme algoritmaları kullanılabiliyorken, Oracle veritabanı şifrelemede ise AES192, AES128 ve AES256 ve 3DES168 algoritmaları kullanılabilir. DB2 verileri şifrelemede ise DES, TRIPLE_DES AES_128, AES_192 ve AES_256 algoritmalarını kullanmaktadır. Oracle ve DB2 simetrik şifreleme yöntemlerini kullanırken, SQL Server simetrik ve asimetrik şifreleme algoritmalarını kullanarak daha çeşitli bir şifreleme sağlamaktadır. SQL Server şifrelemede kullanılabilen tüm simetrik ve asimetrik şifreleme algoritmaları tez kapsamında yapılan yazılım uygulamasında mevcut olup şifreleme olanağı sağlamaktadır.

SQL Server, Oracle ve IBM DB2, PKI açık anahtarları ve sertifikaları kullanarak açık anahtar altyapısının kullanılmasını sağlayarak doğrulama işlemini gerçekleştirebilmektedirler.

Kerberos teknolojisi günümüz işletim sistemleri tarafından yaygın olarak kullanılmakta olup, Kerberos doğrulamayla kullanıcı doğrulama işlemi SQL Server, Oracle ve IBM DB2 tarafından da sağlanmakta ve desteklenmektedir.

Şemalar, ilişkili veritabanı nesnelerini bir arada tutarak veritabanlarının yönetilmesinde ve güvenliğinin sağlanmasında önemli araçlar olarak SQL Server, Oracle ve IBM DB2 tarafından kullanılmaktadır.

Roller, Oracle ve DB2 veritabanlarında "haklar", SQL Server veritabanında ise "izinler"in gruplanması yoluyla kullanılmaktadır. Oracle, SQL Server ve DB2'den farklı olarak, kullanıcı sistem kaynaklarını sınırlandırmak için profilleri kullanır.

SQL Server, Oracle ve IBM DB2'nın her üçü de etkili bir hesap denetimi sağlamaktadır. Ayrıca, Oracle'ın sahip olduğu, Fine-Grained hesap denetimi koşullu kayıt özelliği, trail tablosunun şişmesini engellediği gibi, sadece isteğe bağlı kayıt tabanlı kayıt oluşturulmasına da olanak sağlamaktadır.

Parola yönetimi için SQL Server, Oracle ve IBM DB2 şifre politikası belirlenmesine imkân tanımaktadır. Bu şekilde kullanıcıların çok daha kaliteli bir parola seçimi ve kullanılması zorunlu kılınmaktadır.

Kullanıcılara, işleriyle ilgili görevlerini yerine getirebilmeleri için gerekli olan minimum hakkın verilmesi olanağı SQL Server, Oracle ve IBM DB2 tarafından sağlanabilmektedir.

Tez çalışması çerçevesinde, veritabanı güvenliğini sağlamak için çeşitli yöntemler incelenmiştir. Bu incelemeler sonunda şunların yapılması gerektiği ortaya çıkmaktadır:

- Sunucu üzerinde çalışan bir doğrulama yöntemi veya veritabanının kendi doğrulama yöntemi seçilmelidir.
- Veritabanlarının bulunduğu sunucu üzerindeki işletim sistemi güvenliği ve fiziki güvenlik sağlanmalıdır.
- İşletim sistemi üzerindeki dosyalar ve dizinler için gerekli izinlerin yapılması gerekir.
- Veritabanı ve sunucu seviyesinde, kullanıcılara verilen rollerin ve hakların gerekli olan minimum seviyede olmasına dikkat etmek gerekir.
- Şifre politikalarını sağlayan parola sisteminin kullanılması gereklidir.
- Denetim özelliği aktif olarak kullanılıp kullanıcıların hareketlerinin izlenmesi gerekir.
- Veritabanı sunucusu kurulurken sadece gerekli olan bileşenleri kurmak gerekir.
- İyi bir veritabanı yedekleme planı uygulanmalıdır.
- Ağ güvenliği için güvenlik duvarı, saldırı tespit sistemleri vb. araçlar kullanılmalıdır.
- İşletim sistemi güvenliğinin arttırılması gerekir. Gerekli yamalar ve önerilen güvenlik konfigürasyonları yapılmalıdır. Gereksiz bileşenler kaldırılmalı, anti-virüs yazılımları kurulmalı, çeşitli erişim kısıtlamaları getirilmelidir.

Tez çalışması çerçevesinde yapılan çalışma sonucunda SQL Server veritabanı için ise;

- Windows doğrulama tipini kullanmak daha iyi güvenlik sağlar.
- Windows kayıtlı yordamlarının (Windows extended stored procedure) sınırlandırılması veya kaldırılması gerekir.
- Kullanıcı tarafından oluşturulmuş yordamlar (extended procedure) kurulmamalıdır.

Tez çalışması çerçevesinde yapılan çalışma sonucunda Oracle veritabanı için ise;

- Oracle için gerekli parametrelerin başlatılması için kullanılan init.ora dosyası için izinler sınırlandırılmalıdır.

- SQL*Plus'ın web ara yüzden çalışan şekli olan iSQL*Plus özelliğini etkisiz hale getirmek gerekir.
- İstemci ve sunucu arasındaki bağlantıyı kuran Listener için bir şifre belirlenmelidir.

Tez çalışması çerçevesinde yapılan çalışma sonucunda DB2 veritabanı için ise;

- Sunucu üzerinde çalışan bir doğrulama yöntemi seçilmelidir. CLIENT doğrulama hiçbir zaman kullanılmamalıdır.
- Eğer sistem katalogları üzerinde haklar verilmişse bu haklar geri alınmalıdır.
- Windows ortamında çalışılıyorsa, tüm normal kullanıcılar DB2USERS grubuna, tüm yöneticiler DB2ADMINS grubuna eklenmelidir.

Ayrıca, veritabanı güvenliğinin sağlanabilmesi için iyi bir planlama ve güvenlik alt yapısı kurulmalıdır. Zayıf ve açık noktaların neler olabileceği sürekli araştırılmalı ve giderilmelidir. Veritabanı satıcısı hataları, zayıf mimari, eksik hatalı konfigürasyon gibi açık noktalar için test ve araştırmalar yapılmalıdır. Açık noktaların kapatılması ve onarılmasına öncelik verilmelidir. Veritabanı satıcısı tarafından yayımlanan yamalar bir an önce kurulmalıdır. Sistemlerin sürekli izlenmesi ve analizi yapılmalıdır. Müşteriler, satıcılar ve tüm kullanıcılara güvenmek ama kontrol etmek, izlemek doğru bir davranış olacaktır. Veritabanı sunucusu ne kadar karışık ve zor tahmin edilen bir parola sistemi kullanıyor olsa bile dikkatsiz bir kullanıcının şifresini bir kâğıda yazıp açıkta bırakması tüm güvenliği etkisiz kılacaktır. İyi bir güvenlik politikası takip edilmezse ne kadar gelişmiş ve büyük bir veritabanı kullanılırsa kullanılsın başarısızlıkla sonuçlanacaktır.

Bir başka durum ise, insanlar arasındaki iletişimdeki ve insan davranışındaki açıklıkları tanıyıp, bunlardan faydalanarak güvenlik süreçlerini atlatma yöntemine dayanan müdahalelere verilen yöntem olan Sosyal Mühendisliğin önemli bir tehdit kaynağı olarak ortaya çıkmasıdır. Direkt ulaşılabilir personel, önemli personel (yöneticiler, gizli bilgiye erişim hakkı olan personel), sempati sahibi personel, destek ihtiyacındaki son kullanıcılar, kandırılmış, aldatılmış ya da ikna edilmiş personel tarafından otoriter yaklaşım, yardım önermek, benzerlik ve ortak noktalar bulmak,

mukabele etmek, bağlılık ve dürüstlüğü suiistimal etmek, düşük bağlılıktan yararlanmak şeklinde olabilmektedir. Sahte senaryolar uydurmak, güvenilir bir kaynak olduğuna ikna etmek, güvenilir bilgi karşılığında yardım, para, eşantiyon, hediye önermek, güven kazanarak bilgi edinmek, çöpe atılmış disket, cd, post-it, not kâğıdı gibi, hassas bilgi içerebilecek eşyaları incelemek, eski donanımları kurcalamak gibi yöntemleri kullanabilirler.

Hiç bir sistem insandan bağımsız değildir. Bilgisayar sistemleri, insanlar tarafından tasarlanır, bakımı ve işletimi insanlar tarafından yapılır ve sistemden faydalanan ve sistemi kullananlar da insandır. İnsan bileşeni aynı zamanda bir veritabanı güvenlik sisteminin en zayıf halkasıdır. Bundan dolayı insan faktörünün istismarına dayanan sosyal mühendislik saldırılarının gerçekleşme olasılığının her zaman olduğu ve göz ardı edilemeyeceği açıktır. Sosyal mühendislik saldırılarının etkisini en aza indirgemenin yolu güvenlik politikalarının güncel tutulmasından, personelin eğitiminden ve bilgilendirilmesinden geçer.

Gelecekte, veritabanı sağlayıcılarının veri güvenliğini sağlamak ve artırmak için kullanıcı inisiyatifini mümkün olduğunca azaltarak, veritabanı içindeki veriyi şifreleme, iletim esnasında ağ üzerinde şifreleme, yedekleme, kullanıcı doğrulaması gibi konularda standart güvenliğin sağlanmasını zorunlu kılan bir yapının tesis edilmesi kaçınılmaz olacaktır. Şifre güvenliği için kaliteli şifre seçme zorunluluğu, veritabanı şifrelemenin uygulamalardan bağımsız olması, verinin ağ üzerinde şifrelenmesi gibi yöntem ve özellikler günümüzde bazı büyük veritabanı sistemleri tarafından kullanılmaya başlanmıştır. İnternet ortamı gibi saldırıya açık ortamlarda sistem güvenliğini artırıcı yeni yöntemler geliştirmek de mecburiyet olup önemli ve sürekli bir amaç olacaktır.

Unutulmaması gereken veritabanı güvenliğinin bir son durum, ulaşılacak bir nokta değil, hiç bitmeyen bir süreç olduğudur. Veritabanı güvenliğinin sağlanmasına yönelik çabalar bitmeyen ve devamlı iyileştirmeler ile güncel tutulması gereken bir faaliyet olmalıdır.

KAYNAKLAR

[1] Akgün F., Buluş E., "Yazılım Mühendisliği Açısından Uygulamalardaki Ara Bellek Taşması Zafiyetinin İncelenmesi", ***Trakya Üniversitesi***, Edirne, (2005),

[2] Avolio, F.M., "A Multidimensional Approach to Internet Security", ***ACM Networker Magazine***, 2 (2), pp. 15-22, (1998).

[3] Baskerville, R., "Information Systems Security Design Methods: Implications for Information Systems Development", ***ACM Computing Surveys***, 25 (4), pp. 375- 414, (1993).

[4] Biham E., Shamir A., "Differential Cryptanalysis of the full 16-round DES, Advances in Cryptology: Proceedings of CRYPTO'92", ***Springer-Verlag,*** Berlin , 487-496, (1993).

[5] BrainTree., *Client/Server Database Security* [online], http://www.bti.com/ Whitepapers/3-Tier_Data_Security/WP_Download/bti_down2/Cswp.pdf, **(Ziyaret tarihi: 21 Kasım 2007).**

[6] Brown, L. P., "Analysis of DES and the Design of the LOKI Encryption Scheme, for Doctor of Philosophy", Department of Computer Science, ***University College, University of new South Wales,*** Australia, (1991).

[7] Buluş E., Sakallı M.T. "AES S Kutusuna Benzer S Kutuları Üreten Simulatör", ***Trakya Üniversitesi***, (2003)

[8] Chong R. F., *Leverage your Microsoft SQL Server 2000 skills to learn DB2 UDB V8* [online], http://www.ibm.com/developerworks/db2/library/techarticle/0307 chong2/0307chong2.html, (2003) **(Ziyaret tarihi: 16 Mayıs 2008)**.

[9] *Common Vulnerabilities in Database Security* [online], Pentasafe Security Technologies Inc., http://www.pentasafe.com/whitepapers/pentaSafe_wp_common_ vulnerabilities.pdf, **(Ziyaret tarihi: 24 Aralık 2007).**

[10] *Creation of a Database Security Concept* [online], http://www.bsi. bund.de/english/gshb/manual/s/s02126.htm, **(Ziyaret tarihi: 22 Şubat 2008).**

[11] Cryptographic Toolkit [online], http://www.nist.gov/CryptoToolkit, **(Ziyaret tarihi: 11 Ocak 2008)**.

[12] Dayıoğlu, B., "Ağ ve İşletim Sistemi Güvenliği", TBD BIMY-9 Bildiriler Kitabı, ***Türkiye Bilişimciler Derneği***, (2002).

[13] Dayıoğlu B. ve Özgit A., İnternet Saldırı Tespiti Teknolojileri, ***İletişim Teknolojileri 1. Ulusal Sempozyumu ve Fuarı, Ankara, Türkiye***, (2001).

[14] Daemen J., Rijmen V., "The Design of Rijndael AES-The Advanced Encryption Standard Series: Information Security and Cryptography", ***Springer Verlag***, (2002).

[15] Deborah A. F., Ming-Yuh H., , "Recent Advances in Intrusion Detection Systems", ***Computer Networks,*** 541-545, (2000)

[16] *Definition of Application Level Firewall* [online], http://mtechit.com/ concepts/application_level_firewall.html **(Ziyaret tarihi: 10 Mayıs 2008)**.

[17] *Dirk L., 2006, What is ID?* [online], http://www.sans.org/resources/idfaq/ what_is_id.php **(Ziyaret tarihi: 17 Şubat 2008)**.

[18] Dougherty M., *Securing Your Database* [online], http://www.databasesecurity .com, (2004) **(Ziyaret tarihi: 19 Mayıs 2008)**.

[19] Dougherty M., *Securing Your Database* [online], http://www.db2mag. com/shared/printableArticle.jhtml?articleID=17602334, **(Ziyaret tarihi: 2 Şubat 2008).**

[20] Duane W., *Making Your Network Safe for Databases* [online], http://www.governmentsecurity.org/articles/MakingYourNetworkSafeforDatabases.p hp, **(Ziyaret tarihi: 7 Ocak 2008).**

[21] FIPS 46-3, "Data Encryption Standard", Federal Information Processing Standart (FIPS), ***U.S. Department of Commerce, Washington D.C.,*** (1999)

[22] Godlwasser, S. and Bellare, M., "Lecture N Cryptography", ***M.I.T. Laboratory for Computer Science,*** (1999).

[23] Gornshtein D.,Tamarkin B.,"Features, strengths and weaknesses comparison between MS SQL 2005 (Yukon) and Oracle 10g databases", ***WisdomForce Technologies Inc.,***(2004)

[24] Heys H., S. E. Tavares, "Substitution-Permutation Networks Resistant to Differential and Linear Cryptanalysis", ***Journal of Cryptology***, 1-19, (1996).

[25] *IBM DB2 database* [online] http://www-306.ibm.com/software/data/db2/ **(Ziyaret tarihi: 25 Nisan 2008)**

[26] Jacqueline B., Guido B. ,"DBMS Comparison", ***BeKS Business eKnowledge Solutions***, Bad Homburg, Germany, (2004)

[27] Joaquin A. Trinanes., *Database Security in High Risk Environments* [online], http://www.governmentsecurity.org/articles/DatabaseSecurityinHighRisk Environments. php , **(Ziyaret tarihi: 17 Ocak 2008).**

[28] Keliher L., "Linear Cryptanalysis of Substitution-Permutation Networks", PHd Thesis, ***Queen's University***, Canada, (2003).

[29] Kodaz H., "RSA Şifreleme Algoritmasının Uygulaması", ***Selçuk Üniv.,*** (2003).

[30] Lau, O., "The Ten Commandments of Security", Computers & Security 17 (2), pp. 119-123, ***McGraw-Hill/Osborne***, (1998).

[31] Loney, Kevin, Bryla, Bob. "Oracle Database 10g DBA Handbook" Emeryville, ***California: McGraw-Hill/Osborne***, (2005).

[32] Mezenes, A., J., Van Oorschot, P., C. and Vanstone, S., A., "Handbook of Applied Crptography", ***CRC Press***, (1996).

[33] *Microsoft SQL Server* [online], http://www.microsoft.com/sql/default.mspx **(Ziyaret tarihi: 15 Şubat 2008)**

[34] *Microsoft SQL Server Security* [online], http://www.sqlsecurity.com **(Ziyaret tarihi: 19 Şubat 2008)**

[35] Natan R.B., "Implementing Database Security and Auditing", ***Elsevier Inc.*** USA, 38,39,42-57,97 (2005)

[36] Newman A.,"Protecting Databases", ***Application Security İnc***., United States, (2006).

[37] Newman A. C., "Protecting Databases from Attacks",***West The Mirage***, Las Vegas, NV, May 25, (2005).

[38] NIST Special Publication 800-12, "An Introduction to Computer Security: The NIST Handbook", ***NIST*** , (1995)

[39] NIST Special Publication 800-18, "Guide for Developing Security Plans for Information Technology Systems", ***NIST,*** (1998).

[40] *Oracle Database* [online], http://www.oracle.com **(Ziyaret tarihi: 1 Mart 2008)**

[41] Özgit, A., "Bilişim Güvenliğinden Ne Anlıyoruz?", TBD BIMY-10 Bildiriler Kitabi, ***Türkiye Bilişimciler Derneği,*** (2003).

[42] Ramakrishnan R. and Gehrke J.,. "Database Management Systems" , ***McGraw-Hill***, (2000).

[43] Roeckl C. ,"Stateful Inspection Firewalls"***, Juniper Networks***, USA, (2004)

[44] Salcan O., "Bilgi Güvenliği", ***Silahlı Kuvvetler Dergisi***, Sayı 370, sf: 5467, (2001).

[45] Schneider B. ”Applied Cryptography Second Edition”, Second Edition ,***John Wiley & Sons, Inc., New York***,(1996).

[46] Schneier B., “Applied Cryptography: Protocols, Algorithms, and Source Code in C”, ***John Wiley & Sons Inc.,*** (1994).

[47] *Securing Database Servers*, [online], http://documents.iss.net/whitepapers/ securedbs.pdf, **(Ziyaret tarihi: 29 Aralık 2007).**

[48] *Security Issues* [online], http://www.cyber.ust.hk/handbook3/03_hb3.html **(Ziyaret tarihi: 21 Nisan 2008)**.

[49] Sevgi L., “11 Eylül - Değişen Dünyada Elektronik Savaşlar, Bilgi Güvencesi ve Ulusal Savunma”, EMO Merkez Özel Sayısı, ***EMO***, (2001).

[50] Shao Z., Zhuge Q., HE Y., Defending Embedded Systems Against Buffer Overflow Via Hardware/Software, ***University of Texas at Dallas***, (2005)

[51] Stallings, W., “Cryptography and Network Security Principles and Practices”, ***Prentice Hall***, ***USA***, 4th ed., , 30-35, 110, 181-187, 259-268, (2005).

[52] Şahin A., Buluş E., Sakallı M.T., “Modern Blok Şifreleme Algoritmalarının Gücünün İncelenmesi”, ***II.Müh.Bil.Genç Araşt.K .-MBGAK'2005***, İstanbul, (2005).

[53] Tunçkanat M., Sağıroğlu Ş., “Güvenli Internet Haberleşmesi İçin Bir Yazılım: TurkSteg”, ***EMO***, (2002).

[54] Turner, Tom., *Securing the Database: What are the Issues?* [online], http://www.itaudit.org/forum/security/f218se.htm, **(Ziyaret tarihi: 2 Aralık 2007).**

[55] Xiao, Y., Sun, B., Chen, H.H., Guizani, S., Wang, R., “Performance Analysis of Advanced Encryption Standard (AES)”, ***IEEE GLOBECOM 2006 - 2006 Global Telecommunications Conference,*** San Francisco, United States, 1-5, (2006).

[56] Walia A., Sharma P., “Database Server Security”, ***Department o Information Technology Ministry of Communications&ITE Niketan, 6 C.G.O.Complex****,* (2003).

[57] Walia A., 2004, Sharma P., *Database Security Tools* [online], http://www.cert-in.org.in , **(Ziyaret tarihi: 11 Nisan 2008)**.

[58] Wenke L. and Salvatore J. S., Data Mining Approaches for Intrusion Detection, ***Computer Science Department Columbia University***,(2005).

ÖZGEÇMİŞ

1977 yılında Rize'de doğdu. İlk, orta ve lise öğrenimini Rize'de tamamladı. 1996 yılında girdiği Kocaeli Üniversitesi Bilgisayar Bilimleri Mühendisliğinden 2002 yılında Bilgisayar Müdendisi olarak mezun oldu. 2002 yılından itibaren bir kamu kurumunda Bilgisayar Mühendisi olarak çalışmaktadır. 2005 yılında başladığı Kocaeli Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'ndaki Yüksek Lisans eğitimine devam etmektedir.