Guardium®

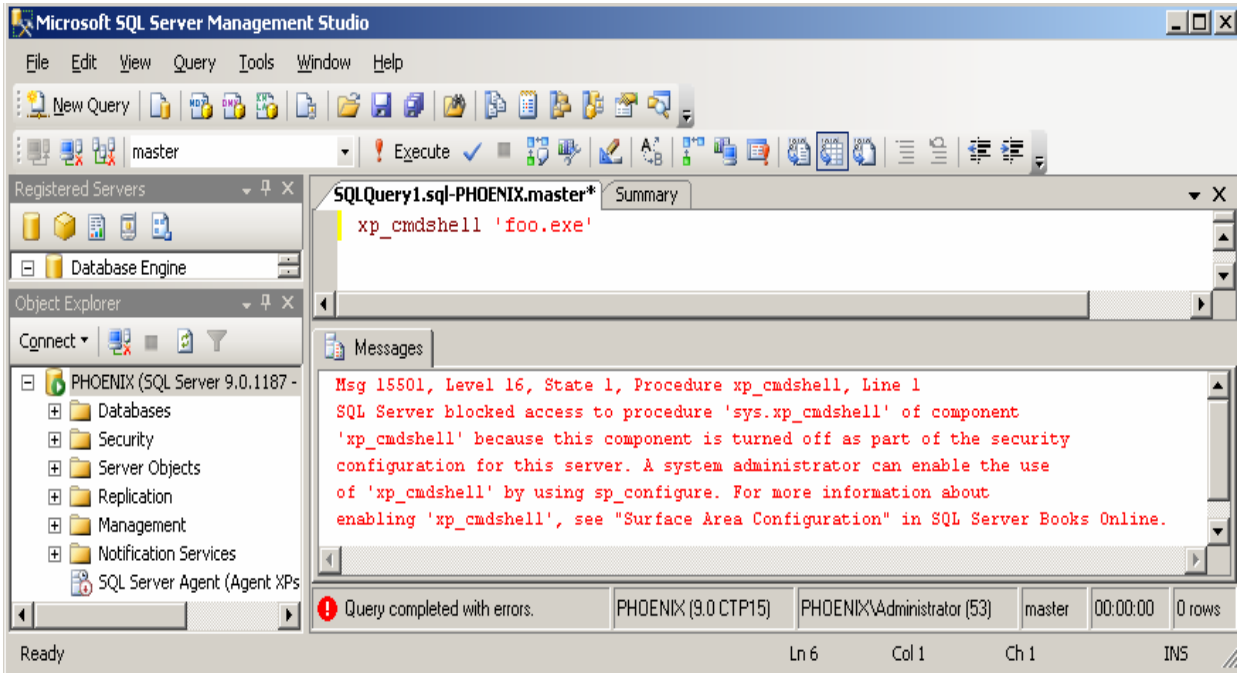
New 2005
Auditing and Security Features

SQL 2005 – New Security Features

- Hardening
  - Assessments
  - Monitoring
  - Auditing
- Enforcement
  - Authentication
  - Authorization
  - Password policies
- Encryption

Hardening

- Trustworthy Computing initiative and SQL 2005:
  - Secure by design – trust Microsoft
  - Secure in deployment - knowledge
  - Secure by default
    - Many features and components are not installed or are disabled by default:
      - Analysis services
      - DTS
      - Replication
      - Full-text search
      - Service broker
      - SQL mail
      - Database mirroring
      - SQL debugging
      - Reporting services
      - SQLiMail
      - Sample databases
      - Xp_web
      - Xp_cmdshell



Authentication

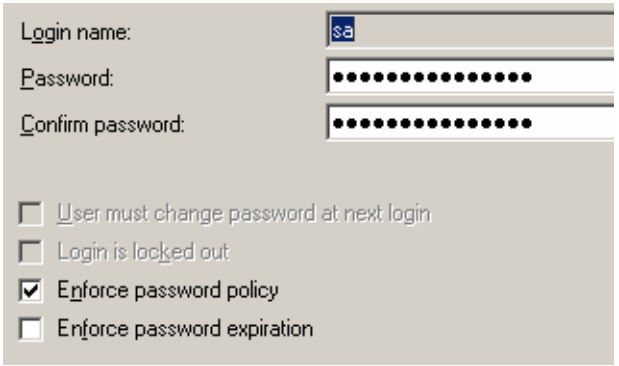
- Mixed authentication is more secure
  - Encrypts with SQL generated certificates rather than hashing
  - Requires new MDAC client based on .NET provider

Passwords and Password Policies

- Manage SQL Server account password and lockout properties
  - Relies on Windows 2003 functionality
    - I.e. – adopts from definitions done at the Windows 2003 level
      - Password complexity
      - Password expiration
    - Uses NetValidatePasswordPolicy() API available in Windows 2003
  - Options in CREATE LOGIN T-SQL statement:

```
< option_list2 >::=
    …
    | CHECK_EXPIRATION = { ON | OFF}
    | CHECK_POLICY = { ON | OFF}
    …
```

  - You cannot set CHECK_POLICY to OFF and CHECK_EXPIRATION to ON
  - `is_policy_checked` and `is_expiration_checked` added to `sys.sql_logins` and in the login properties within management studio:
  - `Lockout policies:`
    - Account lockout duration
    - Account lockout threshold
    - Reset account lockout counter after



Changes in Authorization/Permissions

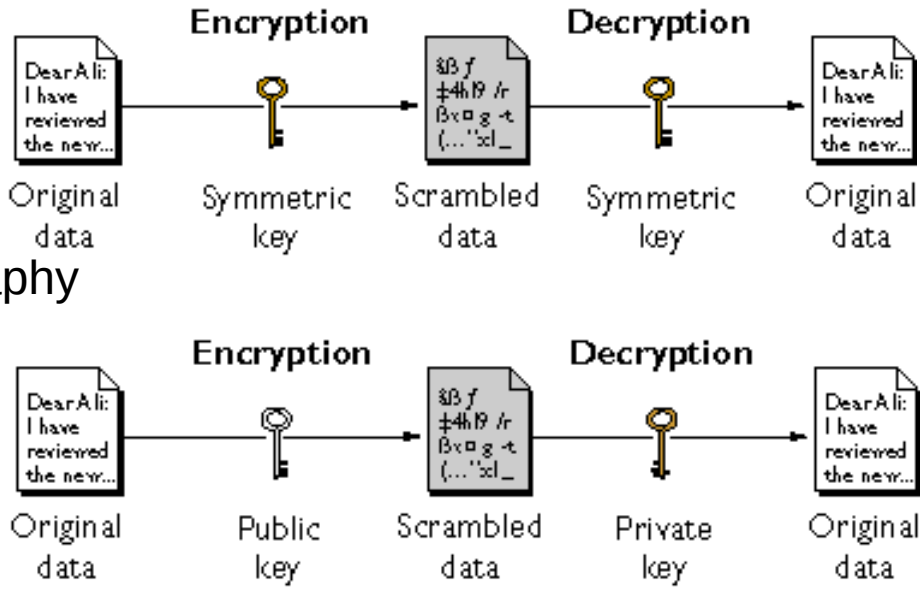
- Schemas:
  - SQL 2000: schema == owner
  - SQL 2005: schema concept conforms to ANSI SQL
    - Object – serverName.databaseName.schemaName.objectName
      - Each user has a default schema for name resolution
    - No need to drop objects or reassign ownership when deleting a user
    - Schemas can be owned by users, database roles or Windows groups
      - Each schema has an owning principal
    - Predefined schemas – sys.schemas
    - Ownership chaining is still based on owners and not on the schema
  - Example usage:
    - Create a schema with procedures that encapsulate access to base objects
    - Do not give access to the base objects
    - GRANT EXECUTE at the schema level
- EXECUTE AS – changing ownership chaining for SPs, functions, triggers, …
  - EXECUTE AS CALLER – standard SQL 2000 ownership chaining
    - Use when caller's permissions need to be checked
  - EXECUTE AS USER={user name|login name}
    - Potentially VERY dangerous in terms of security - impersonation
  - EXECUTE AS SELF – The user who issues the EXECUTE AS SELF clause
  - EXECUTE AS OWNER
- NOTE: Prefer code signing!

Changes in Authorization/Permissions

- Improved Catalog security:
  - Don't use direct access to system tables – use catalog views (sys schema).
    - E.g.
      - Sys.server_permissions
      - Sys.database_permissions
    - Can also still use INFORMATION_SCHEMA views
    - The VIEW DEFINITION permission allows access to the security metadata without providing any other access to the secured resource
  - Catalog view permissions are set at row level
    - E.g. each row has class (database, object, column, ..), permission type (almost a 100 types), and grant/deny/revoke (for granular definition through multiple levels)
      - Deny always takes precedence
  - Minimal access granted to PUBLIC role.

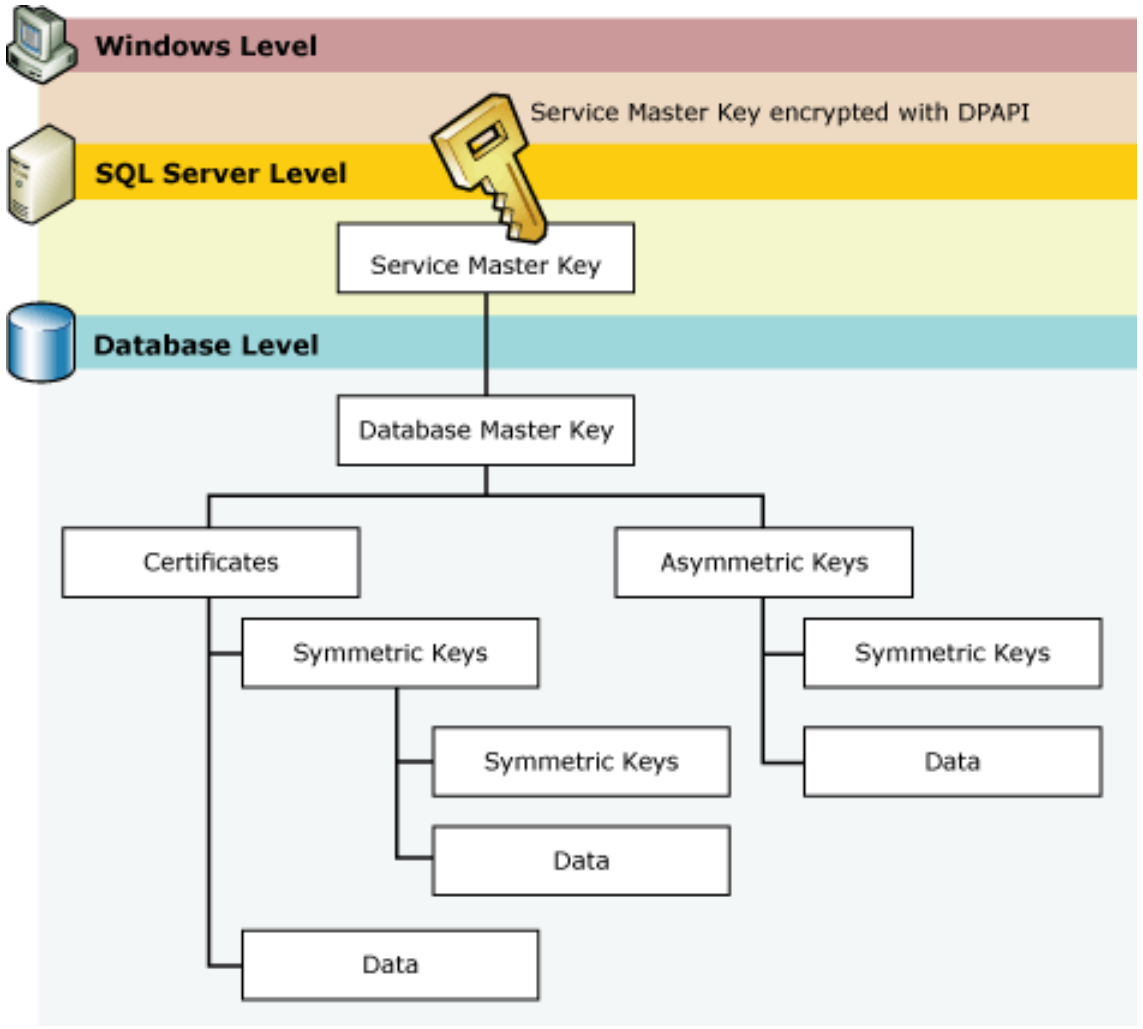
Encryption

- Finally.. - native encryption in SQL Server
  - Encryption of data at rest
  - Comprehensive management of keys
- Symmetric key cryptography
- Public/Private (asymmetric) key cryptography
  - 1000-10000 times slower than symmetric
- Certificates:
  - A public key (or pair of keys) signed using a CA's private key
- Real life:
  - Asymmetric keys are used to generate a per-session symmetric key



Key Management

- SQL 2000 – certificates installed manually on the hosting Windows
  - "Weird" side-effects
- SQL 2005
  - Each instance has a Service Master Key (SMK)
    - Created automatically during setup
    - Encrypted using the credentials of the SQL Server service account
  - SMK is used for connections strings, linked server settings and other instance-level settings
  - SMK is used to encrypt the Database Master Keys (DMK) (in addition to a password used when creating the DMK).
  - DMK is used to create certificates, symmetric and public/private key pairs
    - The DMK itself (different per database) is a symmetric key
    - Certificates and keys can be protected by the DMK or an additional password(s)


Windows Level
Service Master Key encrypted with DPAPI
SQL Server Level
Service Master Key
Database Level
Database Master Key
Certificates
Asymmetric Keys
Symmetric Keys
Symmetric Keys
Symmetric Keys
Data
Data
Data

Encryption API

- CREATE CERTIFICATE Foo WITH …
- EncryptByCert(Cert_ID('Foo'), 'clear text goes here')
- DecryptByCert(Cert_ID('Foo'), @cypherTextVar)

- CREATE ASYMMETRIC KEY Bar WITH ALGORITHM=RSA_2048
- EncryptByAsymKey(AsymKey_ID('Bar'), 'Clear text goes here')
- DecryptByAsymKey(AsymKey_ID('Bar'), @cypherTextVar)

- CREATE SYMMETRIC KEY Bar WITH ALGORITHM AES_256 ENCRYPTION BY CERTIFICATE Foo
  - Or – you can use a pass phrase instead
- OPEN SYMMETRIC KEY Bar CERTIFICATE Foo
- EncryptByKey('Clear text goes here')
- DecryptByKey(@ cypherTextVar)
- CLOSE SYMMETRIC KEY Bar

Code and Module Signing

- Infrastructure used for encryption is always also used for digital signatures
- Used to ensure that a table or view can be accessed only via a certain module (e.g. SP)
  - More flexible than EXECUTE AS or permissions
  - Can accomplish the same things without worrying about ownership chaining or changes to the execution context
- Supports cross-database permissions easily and cleanly
  - Export and import the certificate
  - FAR SAFER!

New Risks

- Servicing HTTP SOAP Requests:
  - Execute stored procs, UDFs, T-SQL batches
  - SQL 2005 running on Windows 2003 (using HTTP.SYS rather than IIS as an intermediary):
- Endpoints – flexible points of entry into SQL 2005
  - Transports
    - TCP/IP, HTTP, named pipes, LPC, …
    - Including encryption – e.g. when setting MIXED authentication for an HTTP endpoint endpoints must run on SSL
  - Payload
    - SOAP, TDS, Database mirroring
  - Listening ports
  - Authentication mode
  - Permissions

SOAP Endpoints

- Authentication mode/security principals:
  - LOGIN_TYPE - WINDOWS or MIXED
  - Windows – no additional SOAP authentication headers are required
  - Mixed – client application must implement WS-Security headers (used to submit the SQL Server login information)
    - <SOAP-ENV:Header>
      <wsse:Security xmlns:wsse= "…">
      <wsse:UsernameToken>
      <wsse:Username>JohnDoe</wsse:Username>
      <wsse:Password Type= "…">pass-word1</wsse:Password>
      </wsse:UsernameToken>
      </wsse:Security>
      </SOAP-ENV:Header>
- Authentication type/mechanism:
  - KERBEROS
  - INTEGRATED – Windows-based Kerberos or NTLM authentication between client making the HTTP request and the server
  - NTLM
  - DIGEST – MD5 hashing (one-way hashing) is applied to the user's Windows credentials on the client and server side
  - BASIC – BASE-64 encoding and easily reversed – you cannot use this without SSL or some other form of encryption

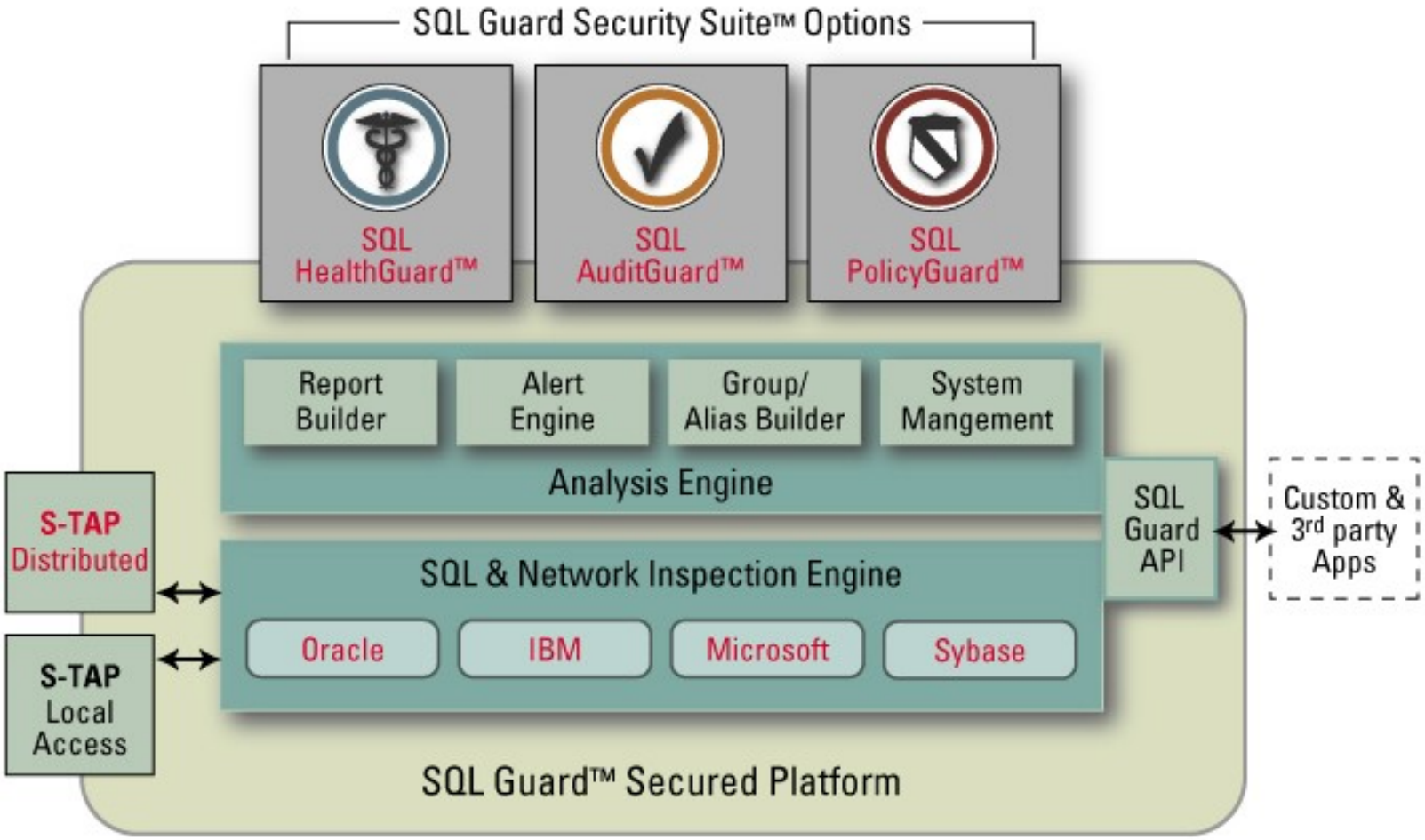
CREATE ENDPOINT *endPointName*
STATE = { STARTED | STOPPED | DISABLED }
AS HTTP (
PATH = '*url*' ,
AUTHENTICATION = (
{ BASIC | DIGEST | INTEGRATED | NTLM | KERBEROS } [,...*n*]) ,
PORTS = ({ CLEAR | SSL} [,... *n*])
[SITE = {'*' | '+' | '*webSite*' },]
[, CLEAR_PORT = *clearPort*]
[, SSL_PORT = *SSLPort*]
[, AUTH_REALM = { '*realm*' | NONE }]
[, DEFAULT_LOGON_DOMAIN = { '*domain*' | NONE }]
[, COMPRESSION = { ENABLED | DISABLED }])
FOR SOAP (
[{ WEBMETHOD ['*namespace*' .] '*method_alias*' **(SP)** } [,...n]]
[BATCHES = { ENABLED | DISABLED }]
[, WSDL = { NONE | DEFAULT | 'sp_name' }]
[, SESSIONS = { ENABLED | DISABLED }]
[, LOGIN_TYPE = { MIXED | WINDOWS }]
[, SESSION_TIMEOUT = timeoutInterval | NEVER]
[, DATABASE = { '*database_name*' | DEFAULT }
[, NAMESPACE = { '*namespace*' | DEFAULT }]
[, SCHEMA = { NONE | STANDARD }]
[, CHARACTER_SET = { SQL | XML }]
[, HEADER_LIMIT = *int*])

More…

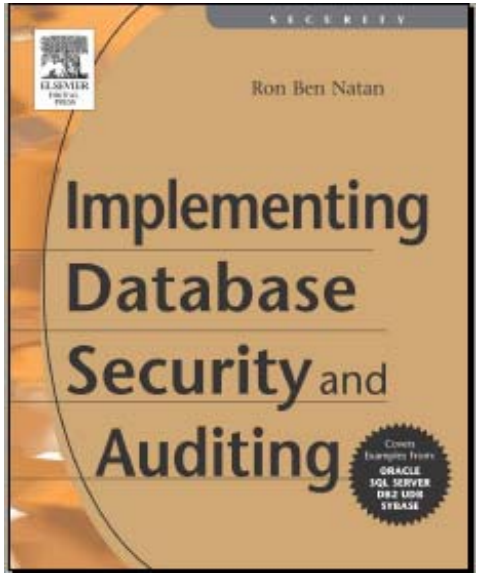
- Auditing enhancements:
  - Run SQL Profiler without granting membership in the SysAdmin fixed server role
  - Covers SQL Server 2005 Analysis Services
  - Triggers for DDL

- Security in SQL Server 2005 Integration Services is very different from security in DTS
  - Encryption
  - Sensitive data protection
  - Package-protection options
  - Roles
  - Digital signing
  - …

SQL Guard


SQL Guard Security Suite™ Options
SQL HealthGuard™
SQL AuditGuard™
SQL PolicyGuard™
Report Builder
Alert Engine
Group/ Alias Builder
System Mangement
Analysis Engine
SQL & Network Inspection Engine
Oracle
IBM
Microsoft
Sybase
S-TAP Distributed
S-TAP Local Access
SQL Guard API
Custom & 3rd party Apps
SQL Guard™ Secured Platform

DATABASE SECURITY STARTS WITH KNOWLEDGE™



SQL Guard™

Ron Bennatan
Guardium, Inc.
230 Third Avenue
Waltham, MA 02451

phone 781-684-6282
fax 781-684-6299

ron_bennatan@guardium.com