

A blue-toned background image showing a complex network of white nodes and connecting lines, resembling a web or data network.

AB*Security Suite Operation Concepts

ActiveBase Server Version 3

email: support@active-base.com

Visit us: www.active-base.com

1 Introduction

Sensitive and Personal Information (SPI) leakage and integrity affect key decision makers and financial reporting. Most organizations have security policies that cannot be enforced on DBA & development tools - unsecured, uncontrolled with unlimited SPI access.

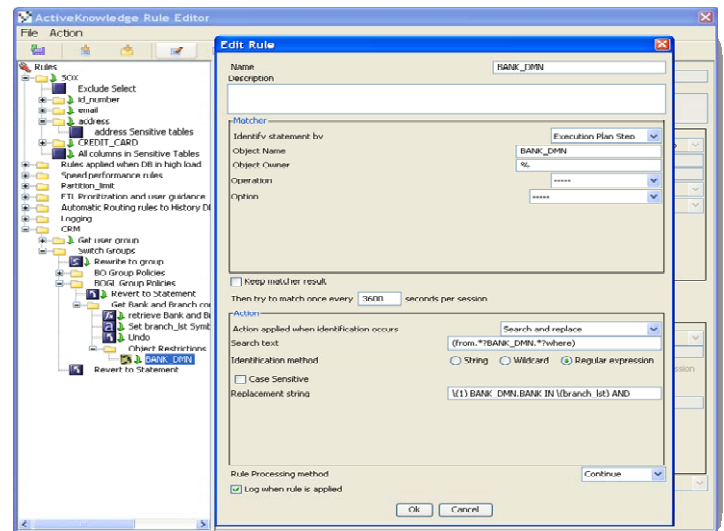
AB*Security Suite enables to enforce security policies and compliance restrictions on IT professionals, outsourced DBA, developers and QA teams.

It provides an automated data security solution for **blocking, scrambling and hiding sensitive and personal information stored in mission critical production systems**, QA and development environments.

AB*Security Suite enables timely legitimate access using time based **security access certificates** issued by the security team.

Automatic learning mode generates detailed user access profiles. User Access Profiles determine user access restrictions, privilege changes and object usage patterns.

SQL injection and Critical Path Update (CPU) prevention rules are applied using frequently updated Knowledge Packs.



Management Console: flexible and powerful Rule Tree

2 Operation Overview

AB*Security Suite is an independent **audit** and **proactive security** solution which enables to implement compliance requirements and powerful real-time security enforcement while preserving application performance.

It intercepts user requests before they reach the database, applying predefined and customized security rules and Knowledge Packs in real-time, scrambling\blocking\restricting\hiding the results returned **without touching the actual data in the database**. AB*Security Suite is easy to deploy and fully transparent to clients, applications and databases.

Example: 1. DBA using Toad (DBA administration tool) enters '**Select ... credit_card ... from customers**

2. The DBA enters and executes the SQL request.
3. Toad opens a connection through AB*Security listener port on to the database listener.
4. The 'Select' request is sent from the Toad client to the database using AB*Security listener port (changing the TNSNAMES.ORA entry).
5. As the request goes through AB*Security listener port, it applies predefined set of rules. These rules identify that a Toad user is submitting a 'Select' request including a personal sensitive column 'credit_card'. The rule applies scrambling.
6. Rule 'Scramble' rewrites the 'Select' request by changing the '**Select...credit_card...**' into '**Select....credit_card_scramble(credit_card)....**', applying a PL/SQL scrambling function on credit_card data column.
7. When the TOAD request is received by the database, it includes the scrambling PL/SQL function on the 'credit_card' column, scrambling the column's sensitive data.
8. The scrambled results are sent from the database to the Toad application.

Predefined Knowledge Pack rules include rule examples and best practice for various leading business applications, with solutions to common Security issues experienced by IT.

Knowledge pack include predefined rules that identify and block SQL injections, Critical Patch Updates (CPU), data masking format library and implementation best practice, restricting specific columns or rows in sensitive tables.

Rules are administered using a single centralized management GUI.

AB*Security Suite provides powerful masking algorithms for securing sensitive data:

- Reusing existing PL/SQL scrambling functions used on QA/Dev environments on selective user access to production sensitive and personal data
- Data substitution, replacing a value in the column with fictionalized data
- Hiding/nullifying, which replaces column values with NULL or '****'
- Randomization, replacing the value with random data
- Skewing, which alters the numeric data by a random variance
- Applying mathematical function making substitutions based on custom functions, or other algorithms
- Character substring masking, which replaces a particular substring with a custom mask

AB*Security Suite provides User Access Profiles. When a user access is not compliant with the profile, the possible actions can be applied:

- **Apply on-line masking/scrambling** to personal and sensitive data
- **Blocking** the request and send alert and/or a user notification
- **Quarantine** - block all session requests and new connections from the same machine or user for X minutes
- **Apply delays** between each request
- **Kill session**

3 How it works

- AB*Security suite includes a powerful software module that acts as a SQL*Net proxy. The AB*Security proxy works on the SQL*Net protocol layer, identifying SQL statement protocol packets flowing from the clients\applications to the databases through AB*Security, applying predefined rules in real-time.
- It is implemented by simply redirecting selectively clients\applications to connect to an ActiveBase Listener ports (by changing TNSNAME.ORA or JDBC property file). Using the ActiveBase Listener port enables administrators to apply SQL statement optimizations transparently, without touching applications or databases.
- AB*Security suite connection switching functions, routes only selected applications/modules connection requests through ActiveBase Listener, where the rest of the applications completely bypass ActiveBase proxy.
- It applies predefined rules on incoming SQL statements.
- Rules apply actions based on various **identification** criteria, including:
 1. Identifying SQL patterns (syntax matching and regular expression),
 2. 'From' clause objects included in the SQL
 3. 'Where' clause conditions – identifying number of days requested on a column in transaction table (for determining if Nested loop hint will be applied or an index range scan hint)
 4. Partial explain plan pattern matching or single execution plan step,
 5. Number of partitions to be scanned,
 6. Time of day
 7. Oracle cost.

- AB*Security suite rule **actions** on incoming SQL requests include:
 1. SQL scrambling – rewriting ‘Select list’ sensitive (SPI) columns
 2. Blocking with an option to return a customized message to the user
 3. Search-and-replace
 4. Define symbols

4 ActiveBase SQL traffic routing scalability

AB*Security includes several features that guarantee the support for OLTP applications with throughputs of as high as 50,000 SQL statements per second through the ActiveBase Listener.

1. Selective routing based on application/module/application server/client host name, OS user or program name list
2. ActiveBase server can be installed on the database server to minimize network hops (Unix/Linux/Windows OS)
3. ActiveBase is built on a multi-threaded server software, with a powerful configuration, managing both the number of threads created upon service initiation, and the number of sessions opened per thread to minimize any possible session delays.
4. ActiveBase supports multiple listener port configurations, to divide incoming application traffic to several different listener ports and to several different ActiveBase servers.
5. Low latency – Packet average transit time in ActiveBase Security is 100 microseconds.
6. Fast Rule engine – several patented innovations have been included in the rule engine in order to be able to support both high-load OLTPs, as well as large scale BI applications. Rules falls into one of three categories:
 - a. Text based, which takes ~20 Microsecond per parsed statement (20/1,000,000 of a second).
 - b. Parser based, which takes ~1 millisecond per parsed statement
 - c. Database based (explain plan or PL/SQL), which takes ~50-100 milliseconds per parsed statement. While usually not harmful on BI environments these should be used more carefully in heavy OLTP environments.

- Rule tree folders minimize rule overhead on incoming traffic by first applying quick text based identification (e.g., identifying SQL statements with large tables and problematic 'Where' clause condition), followed by parser or database based rules, matched to a small percentage of overall SQL statements.
7. A rule tree has been included, that enables ActiveBase to bypass all fast and efficient SQL statements with no delay, while only the long and inefficient SQL statement are caught and manipulated by the rules – accelerating them by x10-1000 times.
 8. ActiveBase software resource consumption is around 1% of the server CPU, with no I/O overhead.

5 Hardware Requirements

- Processor speed: 1GHz or higher (Either as single processor or combined multi processor speed)
- 2 GB of available hard-disk space (600M are required for statement logging purposes)
- Minimum of 1 GB RAM

6 Supported Operating Systems

ActiveBase server can be installed on the following operating systems (32 or 64 bit):

- Sun Solaris 8 or higher
- HP-UX 11 or higher
- AIX 5.2 and higher
- Linux Redhat 7.2 or higher

The AB*Security suite GUI Management Console (for administrating the AB*Priority) can be installed on Microsoft Windows NT/2000 (server or professional)

ActiveBase software is built on Java technology, and such, is installed with a dedicated Java Virtual Machine (JVM), and does not depend on pre-existing Java installations on the server. ActiveBase is released with JVM 1.5.0_04.