

The ALLIANCE Security Broker Framework & Demo

© SAP CEC - 2002

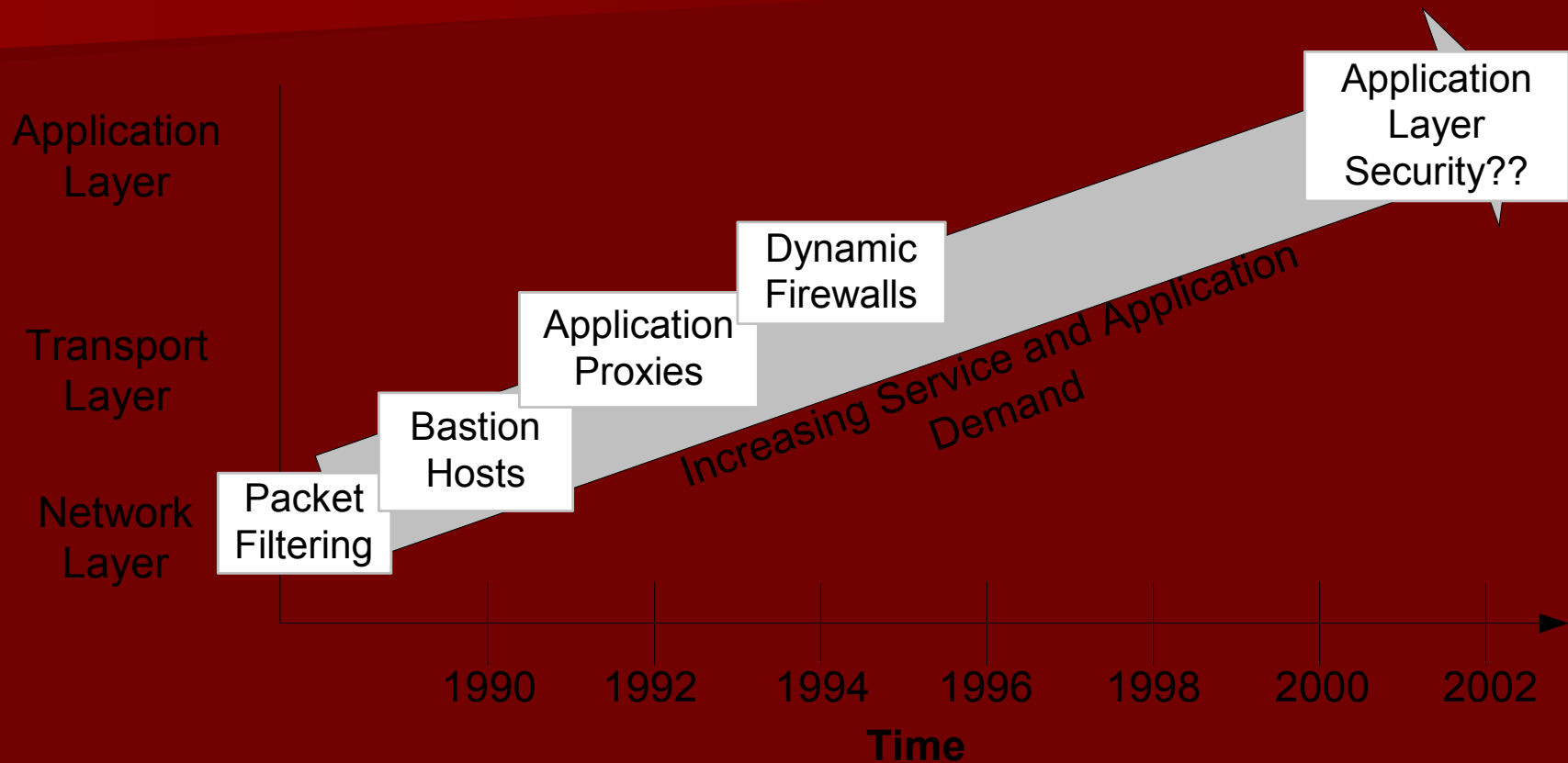
Contents

- Firewalls & Problems
- Overview of the proposed Architecture
- Our Implementation
- Further Work
- Demo
- Conclusion

Firewalls

- barricade for protection of valuable resources from destructive events
- History
 - “firewall-like” mechanisms
 - Application Level Firewalls
 - Dynamic Firewalls
 - ...

Progressive trend of Firewalls



Application Layer Security

- comprises of methodologies and utilities:
 - Specification
 - Design
 - Implementation
 - Execution
 - Management of Security policies

Architecture

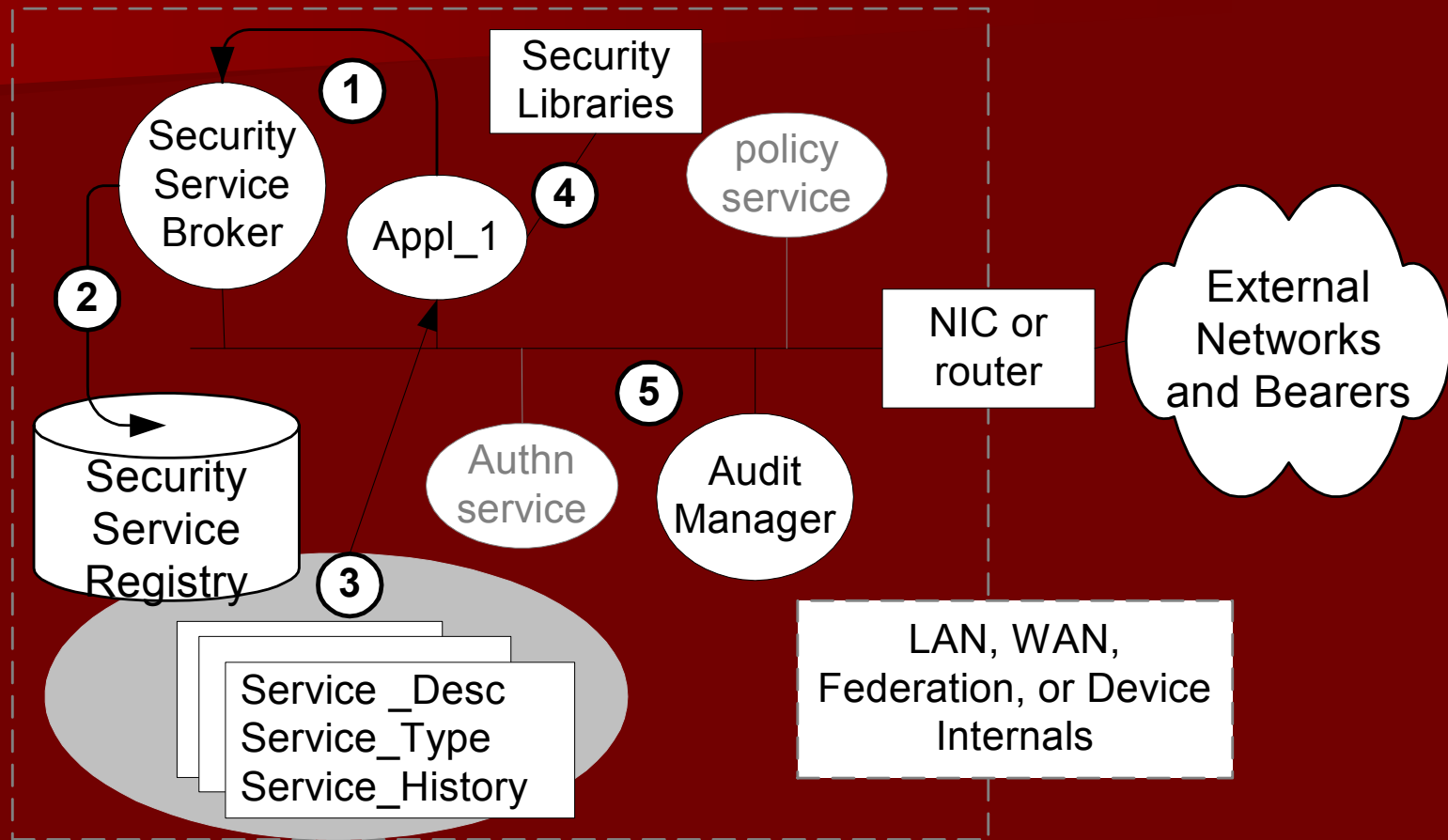


Fig.1. Proposed architecture for Application Layer Security

Why do we need this architecture?

- Inflexibility of Transport Layer Security
- Integrating security into applications from design phases is difficult
- Network Security Administration requires tools to maintain consistency
- Distance between business specification and technical specification of security leads to clashes and misinterpretations

Steps for Application-Broker interactions

■ Application

- announcing its pre-defined security requirements

■ Broker

- consults the S.S.Registry
- searches for registered services that meet the specifications of the application tender
- returns the contract

Steps for Application-Provider interactions

■ Application

- contacts with the Provider specified in the contract

■ Provider

- implements the service
- returns the result back to Application

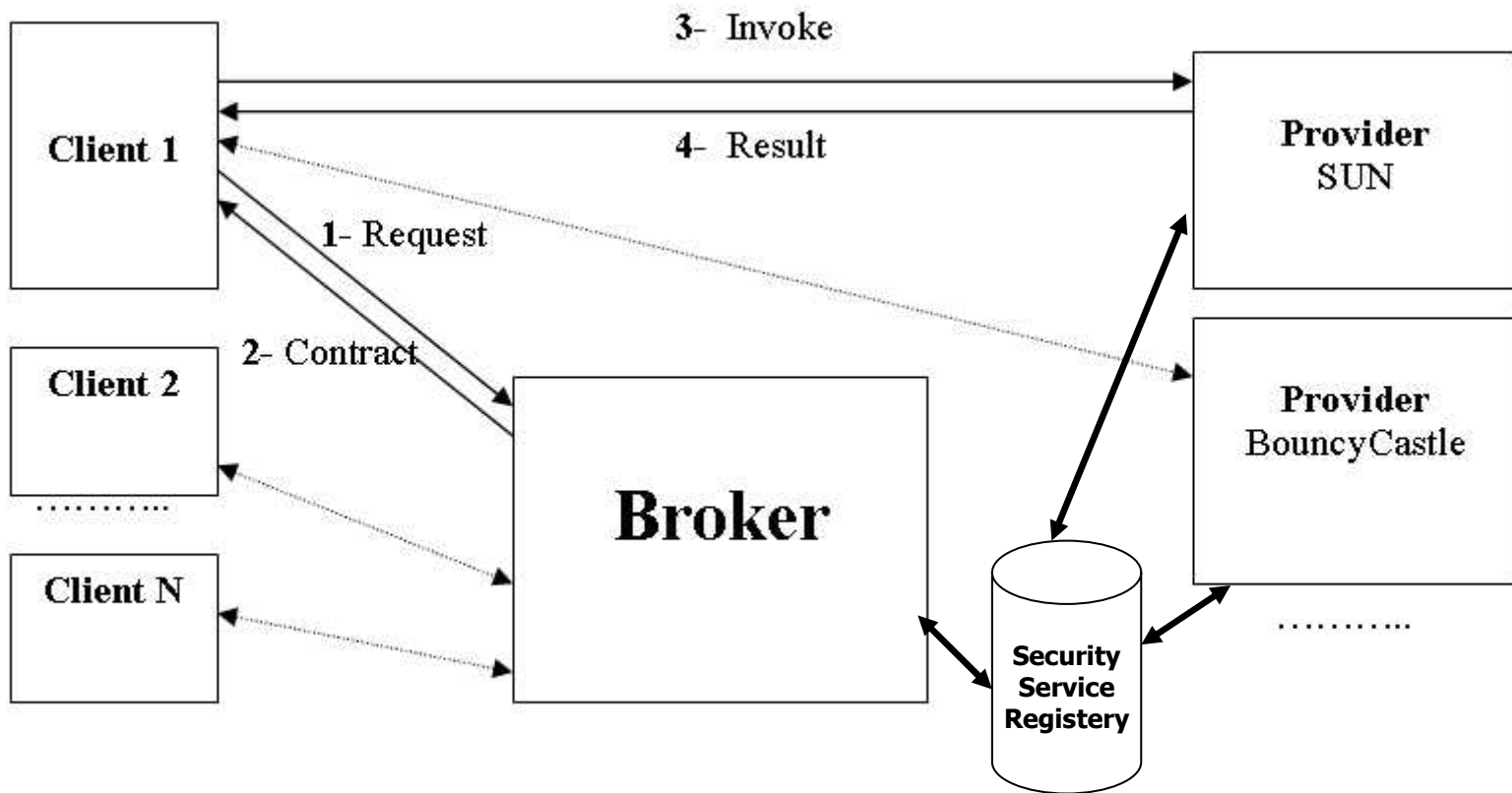
Other Features

- Security Application Libraries included for completion of application's security specifications
- Libraries are available for dynamic binding, whereas services from the providers are enabled by messaging
- Each application responsible for its auditing mechanism
- Auditing requires a standard interface between the application-specific audit processes, and the centralized audit manager

Our implementation

- Implementation in Java
- Multi-Provider, Multi-Client, One-Broker
- Communication over JMS
 - Provider, Server → asynchronous
 - Clients → synchronous
- Repositories as XML data
- Security Providers : SUN, Bouncy Castle

Sequence of Interactions



Request (Client → Broker)

Application

Request

Service Name	Method Name	Method Algorithm	Provider	Location	Availability
Integrity	MessageDigest	MD5	SUN	SUNQueue	80

Add new Method... **Send Request**

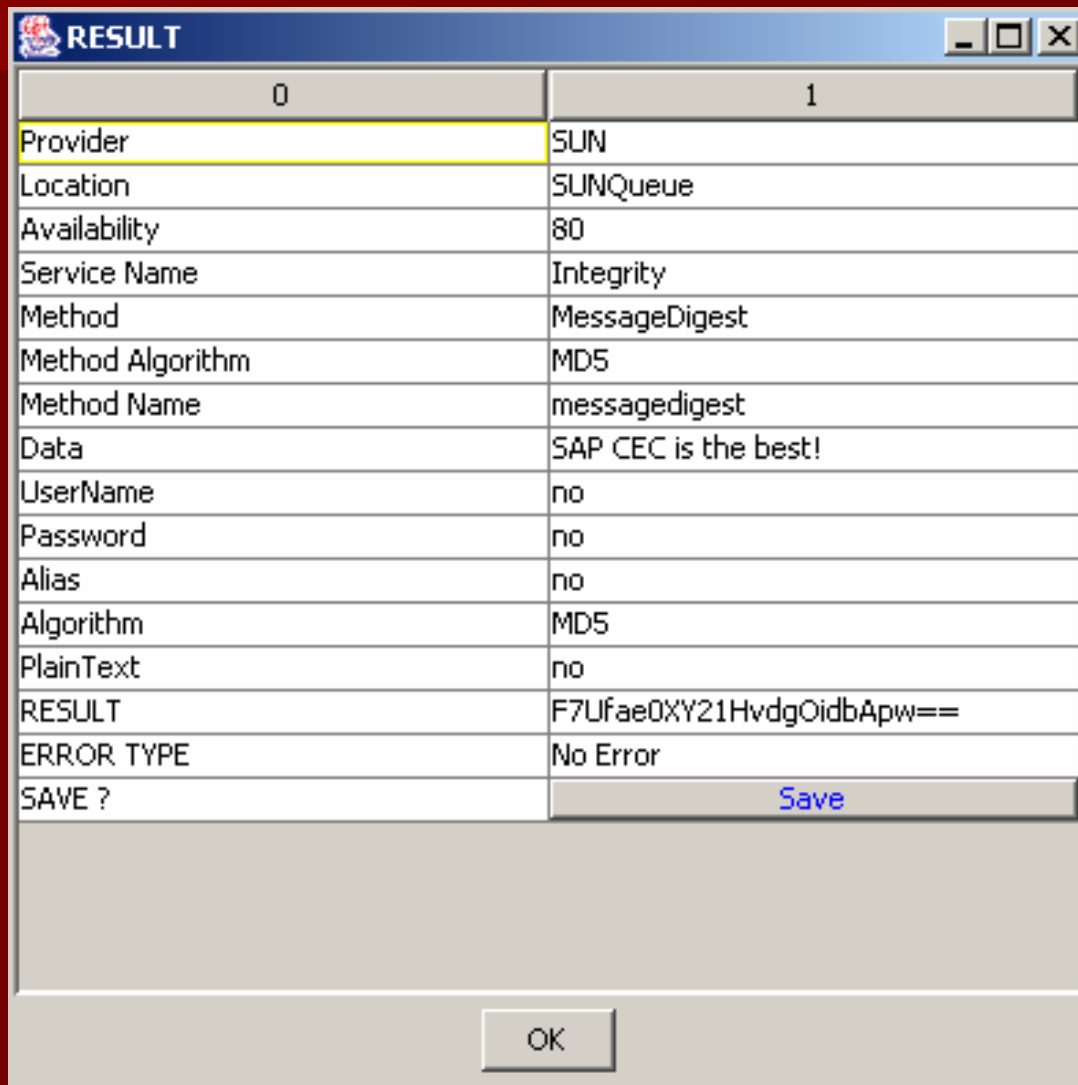
Contract (Broker→Client)

The screenshot shows a dialog box titled "CONTRACT - Ok" with a standard Windows window border. It contains a table with two columns, labeled "0" and "1". The table lists various contract parameters. A checkmark is visible in the "1" column for the first row. Below the table is a large empty text area and an "implement" button.

0	1
	<input checked="" type="checkbox"/>
Provider	SUN
Location	SUNQueue
Availability	80
Service Name	Integrity
Method	MessageDigest
Method Algorithm	MD5
Method Name	messagedigest
Data	yes
UserName	no
Password	no
Alias	no
Algorithm	MD5
PlainText	no

implement

Result (Provider→Client)



0	1
Provider	SUN
Location	SUNQueue
Availability	80
Service Name	Integrity
Method	MessageDigest
Method Algorithm	MD5
Method Name	messagedigest
Data	SAP CEC is the best!
UserName	no
Password	no
Alias	no
Algorithm	MD5
PlainText	no
RESULT	F7Ufae0XY21HvdgOidbApw==
ERROR TYPE	No Error
SAVE ?	Save

OK

Services

- Authentication
- Integrity
- Confidentiality
- Non-Repudiation

Authentication

- Username - password

Integrity

- Message Digest
- MAC (Message Authentication Code)

Confidentiality

- Symmetric Encryption
- Symmetric Decryption
- Asymmetric Encryption
- Asymmetric Decryption

Non-Repudiation

- Create Digital Signature
- Verify Digital Signature

J2EE Technologies

Technologies and APIs for building and deploying enterprise-class server applications.

- Enterprise Javabeans
- Java Server Pages
- Java Servlet
- J2EE Connector
- JNDI
- IDL (for interoperability with CORBA)
- **Java Message Service**
 - Also a new Enterprise bean type
- JavaMail
- Transactions (JTA, JTS)
- JDBC

JMS (Java Messaging System)

- A common Java API for creating, sending, receiving and reading messages
- Enables communication that is
 - Reliable (over SSL)
 - Asynchronous
 - Synchronous
- Designed by Sun and partners

XML repositories

■ Service repositories

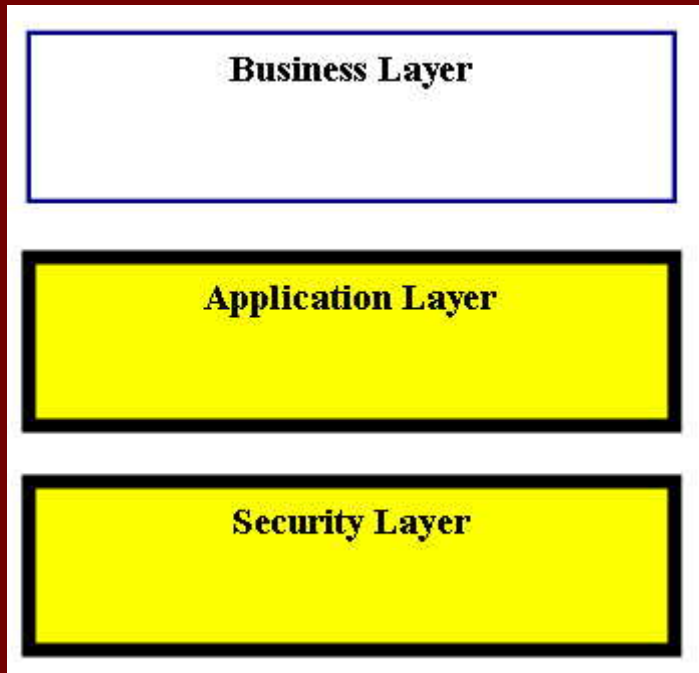
```
<?xml version="1.0" encoding="UTF-8" ?>
- <Provider name="SUN" location="SUNQueue" availability="80">
  - <Services name="Authentication">
    - <Parameters name="usernamepassword">
      <Algorithm name="MD5" methodname="usernamepassword" Param_data="no" Param_username="yes" Param_password="yes"
        Param_alias="no" Param_algorithm="yes" plaintext="no" />
      <Algorithm name="SHA" methodname="usernamepassword" Param_data="no" Param_username="yes" Param_password="yes"
        Param_alias="no" Param_algorithm="yes" plaintext="no" />
    </Parameters>
  </Services>
```

■ Keys repositories

```
<?xml version="1.0" encoding="UTF-8" ?>
- <Keys>
  <Key alias="app1" password="app1" />
  <Key alias="app1_DES" password="app1" />
  <Key alias="app1_DESede" password="app1" />
  <Key alias="app1_Blowfish" password="app1" />
  <Key alias="app2" password="app2" />
  <Key alias="app2_DES" password="app2" />
  <Key alias="app2_DESede" password="app2" />
  <Key alias="app2_Blowfish" password="app2" />
</Keys>
```

Future Work

■ Adding Business Layer



- Restrictions to using services
- User levels/rights

Future Work

- Auditing
 - Each Application will have its own auditing mechanism
 - Central Auditing Manager

Future Work

- Multi Broker
 - Application multicasts & finds out Brokers
 - Application chooses a suitable Broker
- Interface between Broker - Providers
 - Updating service repositories

DEMO

(Digital Signature & Verification)

Conclusion

- Firewalls will become less flexible
- advantages for centralized policy management and auditing
- Application Layer Security: more consistency and flexibility

The ALLIANCE Security Broker Framework & Demo

© SAP CEC - 2002