Oracle® Reference Architecture

Security

Release 3.1

E15630-03

October 2010

ORACLE®

ORA Security, Release 3.1

E15630-03

Primary Author: Dave Chappelle

Contributing Authors: Patrick McLaughlin, Bob Hensle, Stephen Bennett, Anbu Krishnaswamy, Mark Wilkins

Contributors: Jerome Bugnet, Marc Chanliau, Jason Rees

Contents

2 Security Concepts & Capabilities

3 Common Security Standards

List of Figures

Send Us Your Comments

ORA Security, Release 3.1

E15630-03

Oracle welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most about this document?

If you find any errors or have any other suggestions for improvement, please indicate the title and part number of the documentation and the chapter, section, and page number (if available). You can send comments to us at its_feedback_ww@oracle.com.

Preface

The most current themes in the IT industry involve the need to make information and computing more agile, ubiquitous, expedient, and yet inexpensive. This is reflected in emerging computing strategies such as Service-Oriented Architecture (SOA), Cloud Computing, Business Process Management (BPM), and Enterprise Social Computing. They represent an ongoing shift from "locked-down", "silo'ed", monolithic applications to highly distributed and shared computing environments.

However, all of these trends place an increasing burden on security. Presenting functions and data in a highly distributed and shared manner makes them more exposed and potentially vulnerable. Not only is it a concern when sensitive data needs to traverse security domains, firewalls and public networks, companies also need to consider security risks within their own organizations and data centers. Even employees and contractors with no malicious intent can inadvertently cost the company money and damage its reputation by exposing sensitive data via eavesdropping, lost laptops and backups, easily guessed passwords, and other all-too-common occurrences.

Securing the modern computing environment involves many different tools and strategies, such as:

- Physical security: badges, locks, security personnel, etc. to limit physical access to sensitive areas such as computer rooms
- Network perimeter security: firewalls, denial of service prevention, message parsing and validation, etc.
- Desktop security: user authentication, anti-malware solutions (virus, worms, trojans, etc.)
- Vulnerability management: prompt application of security patches for all affected systems
- Security Assurance: compliance with Secure Development Principles to avoid the introduction of security vulnerabilities such as buffer overruns, SQL injection, cross-site scripting, etc. which may result in successful exploits.
- Platform security: proper configuration of underlying servers and operating systems
- Content management: controlling access to managed content and governing the content lifecycle
- Transport layer security: identity validation and encryption
- Application security: user authentication, authorization, and auditing (AAA)

- Message-level security: message encryption, signing, field-level security and data redaction
- Database security: user and administrative access control, encryption, and masking
- Federation: single sign-on (SSO), identity propagation, credential mapping, and trust between security domains
- Identity and access management: managing and provisioning users, roles, entitlements, and access rights

While some of these strategies are specialized forms of security and topics unto themselves, others are pervasive throughout the IT landscape and inherent in almost all forms of business solutions. Oracle Reference Architecture (ORA) Security will examine the more pervasive security concerns that pertain to business solutions deployed into the Oracle solution environment. It will focus on the last six points listed above, which pertain to identity management, authentication, propagation, and authorization; and data confidentiality, integrity, and federation.

ORA Security is a security reference architecture, intended to be used as a baseline for the creation of an organization's specific security architecture. Customization is expected and encouraged in order to best fulfill the unique needs of the organization.

ORA Security provides a blueprint to address many common security needs. It is not however a method to determine an organization's security needs. For instance, it does not prescribe how to perform risk assessments, create threat profiles, or establish trust models. ORA Security is meant to be versatile enough to be applied in many environments with varying degrees of security.

Document Purpose

This document provides a reference architecture for designing an enterprise security framework. This framework supports the security needs of business solutions and helps to unify the disparate security resources commonly found in IT today. It offers security services that are critical to the integrity of modern distributed and service-oriented solutions, and beneficial to legacy systems as well.

In order accommodate those that do not have a strong security background, this document includes chapters that describe some of the common security concerns and a collection of the most relevant industry standards.

Audience

This document is intended for Security Architects, Solution Architects, Data Architects, and Enterprise Architects. The material is designed for a technical audience that is interested in learning about intricacies of security and how infrastructure can be leveraged to satisfy many of the common security needs. In-depth knowledge or specific expertise in security fundamentals is not required.

Document Structure

This document is organized into chapters that introduce security concepts, standards, and architecture views.

The first three chapters provide a primer on security concepts, capabilities, and common industry security standards. These chapters are intended to give the novice reader an understanding of key concepts related to security architecture.

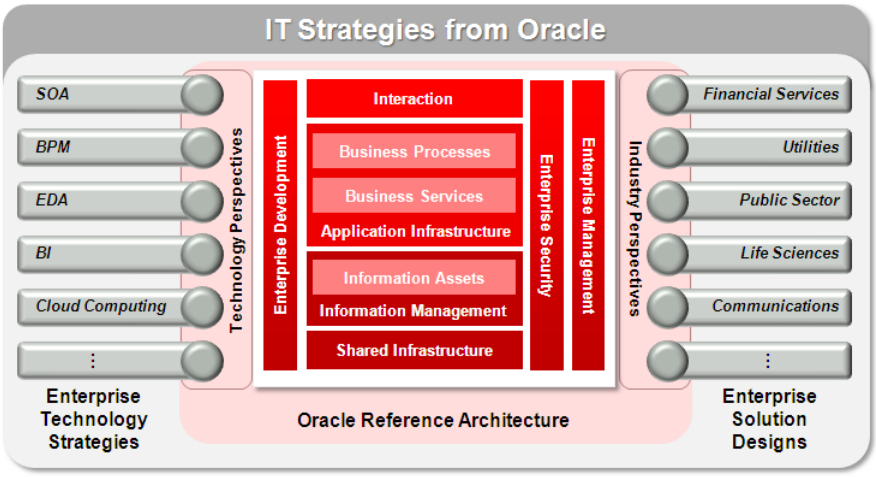
The remaining chapters describe a reference architecture for security that includes secure computing and data platforms, an enterprise security framework, and a set of security services. The architecture is presented using a set of common viewpoints which include conceptual, logical, and deployment views. The architecture is also mapped to Oracle products.

How to Use This Document

This document is designed to be read from beginning to end. Those that are already familiar with security concepts and standards may wish to skip the first three chapters and proceed with the reference architecture definition that begins in chapter 4. Others may read all chapters or skim the first three chapters based on their present understanding of concepts and their interest in security standards.

Related Documents

IT Strategies from Oracle (ITSO) is a series of documentation and supporting collateral designed to enable organizations to develop an architecture-centric approach to enterprise-class IT initiatives. ITSO presents successful technology strategies and solution designs by defining universally adopted architecture concepts, principles, guidelines, standards, and patterns.



ITSO is made up of three primary elements:

- **Oracle Reference Architecture (ORA)** defines a detailed and consistent architecture for developing and integrating solutions based on Oracle technologies. The reference architecture offers architecture principles and guidance based on recommendations from technical experts across Oracle. It covers a broad spectrum of concerns pertaining to technology architecture, including middleware, database, hardware, processes, and services.
- **Enterprise Technology Strategies (ETS)** offer valuable guidance on the adoption of horizontal technologies for the enterprise. They explain how to successfully execute on a strategy by addressing concerns pertaining to architecture, technology, engineering, strategy, and governance. An organization can use this material to measure their maturity, develop their strategy, and achieve greater levels of success and adoption. In addition, each ETS extends the Oracle Reference Architecture by adding the unique capabilities and components provided by that particular technology. It offers a horizontal technology-based perspective of ORA.

- **Enterprise Solution Designs (ESD)** are industry specific solution perspectives based on ORA. They define the high level business processes and functions, and the software capabilities in an underlying technology infrastructure that are required to build enterprise-wide industry solutions. ESDs also map the relevant application and technology products against solutions to illustrate how capabilities in Oracle's complete integrated stack can best meet the business, technical and quality of service requirements within a particular industry.

ORA Security is one of the series of documents that comprise Oracle Reference Architecture. ORA Security describes important aspects of the enterprise security layer including identity, role, and entitlement management, authentication, authorization, and auditing (AAA), and transport, message, and data security.

Please consult the ITSO web site for a complete listing of ORA documents as well as other materials in the ITSO series.

Conventions

The following typeface conventions are used in this document:

Convention	Meaning
boldface text	Boldface type in text indicates a term defined in the text, the ***ORA Master Glossary***, or in both locations.
italic text	Italics type in text indicates the name of a document or external reference.
underline text	Underline text indicates a hypertext link.

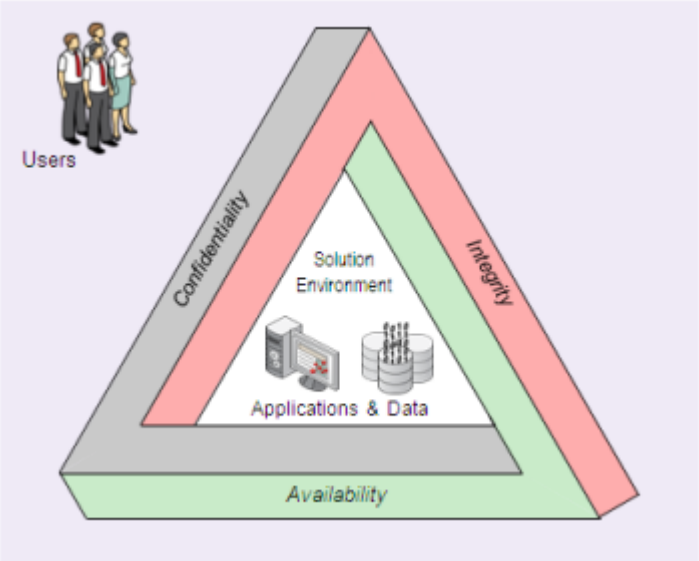
1

Introduction

As organizations evolve their IT architectures, embracing new technologies and computing strategies, they must be diligent in their efforts to ensure security. As Internet commerce and web-based computing emerged, new types of attacks and threats emerged as well. Along with this came government regulations around protecting personal information, payment data, financial records, archives, audit data, etc.

To keep up, an enterprise-level security architecture must be versatile and expandable. It must be able to grow and evolve with the organization, support new business initiatives and technology strategies, and handle the types of threats that may emerge. The architecture must be able to accommodate many different types of information, applications, and users.

Figure 1–1 The CIA Triad



In terms of security, the three common goals are confidentiality, integrity, and availability (CIA). Confidentiality is the ability to restrict access to only authorized parties. Integrity asserts that information changes (additions, updates, and deletions) are only performed by authorized parties. And availability requires systems and information to be available when needed.

The intersection between users, applications and data is a fundamental concept in security architecture. It involves classification of users and resources in order to properly grant access rights, which in turn help provide confidentiality and integrity.

This supports the principle of least privilege, where users have access to what they need in order to perform their job, but nothing more. It helps solve many problems associated with the "all-or-nothing" security approach.

Organizations are advised to examine, and continually re-examine their security goals and architecture. They must define security policies and mechanisms to ensure that sensitive resources are only made available to the right people, and that the organization can determine, at any time, "who has access to what".

This chapter delves deeper into the aspects of applications, data, and users. It also introduces a few common security strategies. It is followed by a chapter that discusses many of the capabilities required to provide confidentiality, integrity, and availability. The remaining chapters define a reference architecture designed to address these goals.

Note: Availability in the sense of fault tolerance is not covered within ORA Security - it is discussed in other ORA core documents pertaining to application infrastructure and data management.

1.1 Application Security

There are many things to consider when defining a security architecture for applications. In addition to the most universal security capabilities, such as user authentication, authorization, and auditing, one must address concerns associated with the type of application being protected, the types of threats that the application is most susceptible to, and other concerns that may be unique to the application itself.

An organization can analyze their applications and produce a classification of how sensitive they are. By doing so, they will indirectly be starting to classify their data. Whether they classify applications or not they must classify data in order to find the subset that needs additional security mechanisms applied to it. Classification of data will identify structured data in databases that must be protected, wherever and however it is stored. A similar classification exercise can be used to classify content.

The following sections describe aspects of application security that should be factored into a security architecture and a classification scheme.

1.1.1 Types of Applications

With regards to security, applications are not all created equal. Some are inherently more prone to attacks than others. Some are more critical to protect than others. And, some are easier to manage and maintain than others.

The following list identifies some of the security characteristics of different types of applications:

- **Network-based applications (vs. non-network).** Not long ago, access to applications (by users) was via hard wired terminal connections. However, since Internet Protocol (IP) evolved into the communications standard of choice, most applications built today use this standard for interaction with users, other applications, and peripheral devices. The ubiquity of IP networks makes them convenient and cost-effective, both to legitimate users as well as hackers.

  The number of physical locations from which a network-based application can be accessed, and potentially compromised, is usually far greater than that of an application that is not connected to a network. This is especially true for network-based applications exposed over wireless networks and/or the Internet.

- **Internal (intranet-based), Public (Internet), and Partner (VPN).** For network-based applications the scope of the network is an important factor for security. Internal applications, particularly those that are not accessible outside the company's office space, and very importantly - those that are not accessible via wireless network, have a security advantage in that physical building security offers a first line of defense. One must gain access to resources within one of the buildings where the network is deployed (office or data center) in order to begin an attack on the system. (This, of course, assumes that the networks themselves are secure, e.g. firewalls, switches, and routers cannot be hacked or bypassed.)

  Applications exposed to partners over a private network or VPN have a greater level of exposure than internal applications since they rely on another company's security measures and trust relationships. It is important to isolate these systems from internal applications by using separate networks and authentication schemes. This will help to guard against attacks on internal applications in the event a partner-facing application is compromised. If the partner application is used for electronic commerce, then non-repudiation and secure auditing are very important considerations.

  Public-facing applications generally have the greatest amount of risk. One must assume they are under constant attack from professional hackers using the most sophisticated techniques. These applications should also be isolated from internal applications, and should be protected as much as possible. Protection may include network-based solutions such as firewalls, network address translation (NAT), reverse proxies; O/S and code level hardening; as well as the many application and data security techniques discussed in this document.

- **Web 1.0, Web 2.0.** First generation web applications, (i.e. Web 1.0), use HTML for information presentation, and occasionally use Java applets or ActiveX for specialized user interaction. Over the past decade many web server vulnerabilities have been discovered and exploited, creating a challenge for application owners and vendors to keep up with security patches.

  Web 2.0 introduces even more client-side code to enrich the user experience (i.e. Rich Internet Applications). This can be done in a standardized way, such as the Ajax framework. New message-injection attacks are possible in areas such as cross-site scripting and cross-site-forgery due to poor protocol implementations and poor message parsing. An example exploit is the SAMY worm created on MySpace.

  In the case of both Web 1.0 and Web 2.0 applications vulnerabilities may be discovered throughout the entire lifetime of the application. Therefore it is necessary to continuously carry out vulnerability testing, e.g. during development, during pre-production testing, and on a periodic basis while the application is deployed. Specialized security companies compile and update lists of attack vectors and provide products or on-line services for conducting such scans.

- **Self-contained** applications are those that manage their own security data, policies, and services. They have an advantage in terms of being isolated from attacks on common security infrastructure. E.g., if a fraudulent user is added to an identity management system, or a user's SSO password is guessed, it would have little effect on an application that does not share users, credentials, roles, or trust relationships with other applications.

  Applications that rely on common security infrastructure share the same risk of being compromised. However, from a maintenance standpoint, self-contained applications multiply the level of effort required to maintain and administer security, and often can be a greater overall systemic risk if procedures are not followed to ensure that proper security policies are enforced. These applications

may be forgotten when user accounts are purged, passwords changed, and/or administration events are audited. They may also act as an entry point to other systems that they share a trust relationship with. In addition, they may cause problems with maintaining or attesting to regulatory compliance, as their internal security implementations can make it difficult to determine "who has access to what".

- **Legacy, packaged, and leased apps.** Applications that are bought, leased, or were built years ago, may not be completely compatible with the preferred security architecture of the day. In some cases you are stuck with what you have, both good and bad. Additional security measures may need to be layered around these applications in order to fit them into the broader IT framework. For example, access to functionality and data could be wrapped in services that introduce security features that are in alignment with the current security architecture.
- **SOA Services.** SOA Services are usually designed with reuse in mind. Therefore they can provide the same functionality, or offer the same set of data, to many different consumers. In terms of access control, SOA Services must base access control decisions on roles, attributes, rules, etc. that are universal to all consumers. In addition, data provided by a SOA Service must adhere to data classification restrictions that might differ between consumers. For instance, the same query service may need to redact various rows or columns of data based on restrictions assigned to classes of consumers.
- **Monolithic vs. Composite / Distributed.** Applications with a higher degree of distribution, such as composite SOA applications, business processes, distributed portals, etc., require a greater total effort to secure. Concerns that may only affect the entry point into monolithic applications, such as user authentication, access control, message encryption, and identity federation, can be re-introduced into the distributed application at every distribution point.

  Likewise, access control for an application composed of multiple SOA Services or business activities may need to be represented as the intersection of the access restrictions of every SOA Service or activity that the application is composed of. In other words, a user shouldn't be given permission to begin a function/process that will eventually fail due to lack of permissions to one of its distributed parts.
- **Cloud Computing.** There are several forms of cloud computing and various hosting options, which makes cloud security a large topic. Cloud computing in all forms involves sharing resources, aka multi-tenancy. Sharing may occur at various levels of the solution stack, e.g. hardware, middleware/database platform, and application/service levels.

  Private cloud implementations involve sharing resources across solutions within an organization. Although multi-tenancy can complicate efforts to maintain confidentiality and integrity, the organization can still host infrastructure in their native environments and leverage their own preferred platforms, security infrastructure, and security configurations.

  Public forms of cloud computing involve the use of infrastructure, platforms, or applications that are hosted outside of the company's secure IT environment. So in addition to the normal application security concerns, one must also consider:

  - How well the organization’s security architecture can support the cloud solution. Will security (access control, auditing, etc.) be seamless, or will the cloud solution introduce an entirely new security silo.
  - Data, in all states (in transit, cache, queues, disk, backup media, etc.), must be protected. Encryption and key management are paramount concerns, particularly when data from multiple organizations are mixed within the same

database. Will the cloud solution support adequate encryption protections, and will there be any latency issues.

- Some parts of the infrastructure may be shared with other organizations or departments which increases the likelihood of data or applications being compromised.
- Cloud solutions may be hosted outside of the organization's home country. There may be regulations that prohibit processing or storing data in certain foreign countries.
- Administrators of these applications, databases, networks, etc., may not be under your direct control/governance, or work for your company. Administrative access should be limited according to job function.
- Administrative access to the operating system, virtual machine, platform, and/or application, may be provided over the Internet.

Note: Due to document size considerations, security for cloud computing will be addressed in a separate document. However, ORA Security is designed to be extensible to support cloud computing, and may be sufficient as-is for many private cloud implementations.

- **Clustered applications.** Clustered applications provide failover capabilities by sharing state information between machines and servers. State information can include sensitive data. One must consider the risks involved in having these data being transmitted over a network, particularly if the data are not encrypted. This is also something to consider with applications that use or share a distributed cache.
- **Business Intelligence (BI) systems.** This includes data warehouses, data marts, OLAP cubes, and the applications and tools that access them. These data stores provide a lot of historical data which is often collected from a large number of sources that span system and organizational boundaries. Data are exposed to new and different audiences in order to improve business efficiency and decision making.

  BI data stores must adhere to the same security policies as the transactional systems from which data originated. Information gleaned from them, and data exposed by them, must not compromise the integrity of the IT environment. As data are migrated from transactional databases to data marts, staging areas, and warehouses, security restrictions must be carried over as well.

  Access to business intelligence may be limited based on the level of detail. For instance, many users may have access to summary data, whereas only selected users may be able to drill down into specific detail records. As always, a balance must be struck between security and ease of information access.

1.1.2 Risks and Threats

Applications, particularly those exposed over a public network, must be protected from a number of threats, such as:

- Interruption and unavailability. Denial of service attacks are meant to cause outages or delays by either flooding the application with a large number of requests and/or sending extremely large requests that either overrun the application's memory capacity or cause long delays due to unnecessary message processing.

- Compromised control. Hacking and virus attacks aimed at injecting foreign code into the application or memory, which in turn provide some degree of control to the hackers. Once compromised, the application can be instructed to perform new functions which could include the destruction of data or transmitting sensitive information to outside parties.
- Fraudulent transaction processing. An application can be sent illegitimate requests which eventually render the application useless. For example, a reservations system could be fed a large number of "normal looking" requests from a malicious source until no more reservations are available for legitimate paying customers.
- Unauthorized access. Users gaining access to perform functions or view information beyond the limits intended by the site owner.
- Data-related threats. See Section 1.2.3, "Threats to Data."

Other threats and risks can come from within the organization without malicious intent. For example, an administrator could reduce application availability by improperly configuring it or the resources it needs to run effectively.

A major risk to an application is that one or more vulnerabilities are discovered by an attacker before the application vendor or owner discovers them. Application security must be proactive, to avoid attacks and learn from others' misfortunes, as well as reactive, to detect and counter attacks, and continuously make improvements to strengthen security.

1.1.3 Classification

For IT Security it is essential to have a view on the value of the information that needs to be protected. This is usually done by having an information classification scheme, (Section 1.2.4). One means of arriving at this classification is to also classify applications in terms of their importance to the organization. People will intuitively know if an application possesses key information that could be manipulated and abused by an attacker (outside hacker or insider). Studying the community of users that use the application will also help arrive at its importance or classification.

Applications can be classified in a number of ways, such as:

- By the user community it serves, such as HR, Finance, company executives, all employees, all persons working on behalf of the company (includes contractors and temporary workers), general public, etc.
- Based on network exposure. Levels might include: locked down (no network access), secure production environment access, general organization-wide intranet access, partner access, Internet access limited to a specific user community, and Internet access open to the public.
- Based on information confidentiality. Some applications process personal information while others do not. Likewise, in military terms, an application might be targeted towards individuals with a specific level of clearance.
- Based on the applicability of existing laws and regulations. For example, HIPPA puts more security emphasis on patient records than would otherwise exist.
- Based on business criticality. Some applications may have a direct and severe contribution or impact to revenue. Examples include order processing, credit card processing, call processing, securities trading, and travel reservations. Others may have little or no impact.

Each means of classification should be considered with respect to security requirements, risks, and ramifications. Each classification may have multiple levels

with varying security requirements. Policies may be defined for each classification and level in order to provide general guidelines for application security.

1.1.4 Ownership

Application ownership can have an effect on application security. Ownership, in this sense, refers to the person, department, or organization that has authority over the security protections and processes that are followed.

Owners have a role in expressing policy. They may choose to set strict policies and follow best practices and reference architectures. Or, they may choose to be lax on security and/or set their own standards. Ideally, an enterprise-class security architecture and processes are developed by pooling industry best practices and ideas, and all applications follow a common strategy without being compromised by ownership issues.

In some cases, as with cloud computing, there may be external application owners. There may also be applications with no owner at all. These cases need to be considered in order to establish a secure computing environment. Therefore in addition to establishing a security architecture, governance must also be established to ensure that security controls are met and the architecture is used properly.

1.2 Data Security

Congruent with application security is data security. Data security involves the protection of data from creation to destruction, and in all states in between. This includes transmission over the network, temporary storage in caches and queues, persisted storage in databases and files, as well as backup tapes and other media such as USB thumb drives and DVDs.

1.2.1 Types of Data

Data security pertains to both structured and unstructured (aka semi-structured) data. Structured data can be defined as forms of data that can be expressed using a data model. Structured data may include "unstructured" fields or columns, such as images and sound files, and still follow a structure that can be modeled.

Structured data can be represented in many forms including relational (object relational and table relational), hierarchical, flat tabular (e.g. comma-separated value), etc. Some forms are better represented by data security standards than others. For example, XML Encryption and XML Signature define standards for securing data in XML form. This offers an advantage over other forms that would require custom-designed protection mechanisms.

Note: This document will only address security of structured data. Unstructured data, also known as content, will be discussed within the Enterprise Content Management ETS documentation.

1.2.2 States of Data

Data can be defined as has having different states. For example:

- In Motion - data transmitted over a connection from one physical location to another. This includes all types of communications between clients, servers, applications, SOA Services, business processes, caches, databases, queues, files, backup devices, etc.

- At Rest - persisted for any length of time in any form, e.g. database, cache, file, tape, queue, etc. Data at rest may exist on devices located within a protected computer environment as well as devices that are transported in public (laptops, USB keys, tapes, etc.).
- In Memory - a term used here to describe data that is currently being processed and therefore exposed to any program, diagnostic tool, operating system command, virus, etc. that is able to access memory.

It is important to consider all states of data when designing a security architecture. For instance, transport security may be used to protect data in motion between two server processes, but the same data might also be exposed over the network by a distributed cache. Likewise, data at rest may be protected from a database administrator through proper access control policies, but may be processed in the clear and logged within an application as part of an error handling routine or activity log.

1.2.3 Threats to Data

Threats to data include:

- Disclosure (lack of confidentiality); making data available beyond the intended audience. Risks for disclosure include misuse, theft (such as when credit card or bank account data are compromised), embarrassment (disclosure of personal information), espionage, and possibly even injury or death (disclosure of military or law enforcement data).
- Alteration (lack of integrity); allowing changes to data by unauthorized parties. Risks for alteration include, theft (such as when financial records are changed), embarrassment (obvious presence of an intruder), loss of trust, and possibly even injury or death (alteration of military or vital health-related data).
- Destruction - can result in temporary inconvenience (until a backup is restored), or catastrophic loss (if no backups exist/remain, or if critical data were unavailable in an emergency).

1.2.4 Classifications of Data

In terms of security requirements, not all data are equal. Some are freely available in the public domain, while others have varying degrees of sensitivity. An important aspect of data security is the evaluation of data to determine their appropriate security requirements. In order to make the process manageable, organizations often adopt a simple classification scheme.

One means of classification addresses confidentiality. A sample[1], shown in Table 1–1, depicts classifications that range from Public to Top Secret. A description of each level is provided along with some examples.

Table 1–1 Sample Data Confidentiality Classification Scheme

Classification	Description
Top Secret	Highly sensitive internal documents e.g. pending mergers or acquisitions; investment strategies; plans or designs; that could seriously damage the organization if such information were lost or made public. Information classified as Top Secret has very restricted distribution and must be protected at all times. Security at this level is the highest possible.

[1] Data classification example provided by RUSecure

Table 1–1 (Cont.) Sample Data Confidentiality Classification Scheme

Classification	Description
Highly Confidential	Information that, if made public or even shared around the organization, could seriously impede the organization's operations and is considered critical to its ongoing operations. Information would include accounting information, business plans, sensitive customer information of bank's, solicitors and accountants etc., patient's medical records and similar highly sensitive data. Such information should not be copied or removed from the organization's operational control without specific authority. Security at this level should be very high.
Proprietary	Information of a proprietary nature; procedures, operational work routines, project plans, designs and specifications that define the way in which the organization operates. Such information is normally for proprietary use to authorized personnel only. Security at this level is high.
Internal Use Only	Information not approved for general circulation outside the organization where its loss would inconvenience the organization or management but where disclosure is unlikely to result in financial loss or serious damage to credibility. Examples would include, internal memos, minutes of meetings, internal project reports. Security at this level is controlled but normal.
Public	Information in the public domain; annual reports, press statements etc.; which has been approved for public use. Security at this level is minimal.

In addition to confidentiality, data should also be classified in terms of integrity and availability. These aspects complete the CIA triad mentioned earlier. For integrity, one must consider the degree to which data is guaranteed to be accurate and correct. It includes integrity of data in all three states, (at rest, in motion, and while being processed). A sample data integrity classification scheme is shown in Table 1–2.

Table 1–2 Sample Data Integrity Classification Scheme

Classification	Description
Guaranteed	Used for data with non-repudiation requirements. Data is guaranteed to be correct and the identity of its creator, and that of any modifier, cannot be refuted. Complete change history is available and is also guaranteed. Trust relationships are at a personal / individual level. This classification is used when the integrity of information must meet the strictest qualifications including litigation.
Verified	Information is verified to be correct in a manner that detects any corruption during transmission, processing, and storage. Access to information is controlled, and changes are audited. Audit logs are protected from tampering. This classification is sufficient for critical information where non-repudiation is not a concern.
Audited	Write access is managed and all changes to persisted information are recorded. Changes will be detected, and can be corrected, unless audit logs are also compromised. Therefore data integrity is a function of the integrity of system administrators as well as the systems and networks that process and transmit data. This level of integrity is most often used for transactional data that may affect revenue.

Table 1–2 (Cont.) Sample Data Integrity Classification Scheme

Classification	Description
Managed	Information updates are protected by access control. Data integrity is directly related to the integrity of the users given access and the systems and networks that process and transmit data. Data from departmental applications or data not directly tied to revenue generally fall into this category.
As-Is	Information open to anonymous creation and use. No guarantee of authenticity, accuracy, or correctness. Examples include public postings such as comments, open wikis, etc.

Availability classifications address the ability to avoid loss of data either via malicious or accidental means. Availability is often applied to the system as a whole, including all hardware, software, networks, data, etc. required to ensure proper operations. The sample classifications in Table 1–3 have been worded to more specifically address data.

Table 1–3 Sample Data Availability Classification Scheme

Classification	Description
Fail-Safe	Systems are fault-tolerant across multiple sites to support geographic redundancy. Availability ensured even in the event of a regional catastrophic event such as a natural disaster or terrorist attack. Data redundancy across multiple sites and offsite backup storage. Mechanisms actively detect and thwart DoS attacks.
Fault-Tolerant	No single point of failure exists within the system environment. Redundant copies of data are maintained online. Mechanisms are in place to support rapid rollback of both intentional and accidental data changes. Stringent access restrictions all but eliminate the threat of malicious or accidental data loss. Mechanisms actively detect and thwart DoS attacks.
Managed	Service level agreements in place. Systems have been designed with the appropriate level of redundancy and resiliency to meet availability requirements and are managed accordingly. Access is controlled to minimize the likelihood of unavailability due to malicious or accidental activity. Information is backed up and verified.
Minimal	Data are managed in a controlled environment (e.g. computer room) and regularly backed up. No SLAs established and no redundancy built into the system.
None	No guarantees or service level agreements in place. No steps taken to manage data or maximize availability. Backups may exist but are seldom verified.

The classifications that an organization chooses to adopt provide a means to describe the level of security that data must be given. These classifications then need to be mapped to security policies that define specific requirements, such as encryption strength, authentication mechanisms, etc., which can then be applied to the security architecture.

1.2.5 Lifetime

In order to provide security throughout the full data lifecycle, one must consider when the lifecycle should end, and how it will be terminated. A number of factors may influence data retention policies including government legal and tax regulations,

personal information privacy policies, transaction processing periods, and business intelligence usefulness.

Obviously, data must be retained long enough to satisfy all of these requirements, although it may take various forms. It may originate in online transactional databases and remain there until the completion of a (weekly, monthly, quarterly) transaction processing period. At some point it may be copied to business intelligence data marts or warehouses where it exists until it is no longer useful. Afterward it exists only on backup tapes for a period of time until all legal retention requirements have been met.

Data security is important at all stages of the lifecycle. As data transitions from one environment and medium to another, confidentiality and integrity must be maintained. Although in some cases the need for security may diminish over time, some data retain their criticality until the end. The organization carries forward a degree of risk and the responsibility to safeguard data for which it possesses.

Data termination processes need to be defined and followed. The process may vary depending on classification, or may be universally carried out regardless of classification. A common method of termination is called digital shredding. This involves encrypting data with a temporary key and then deleting the key. It can be combined with physical methods such as erasing (overwriting), reusing, and destroying backup tapes.

1.2.6 Ownership vs. Stewardship

Data ownership, or stewardship, is important to establish, as it affects security. As in the case of application ownership, those that feel they own data may be inclined to follow their own unique approach to data security. This is especially troublesome for data that needs to flow freely between applications, SOA Services, processes, and people. Silo'ed security approaches can be difficult to federate without introducing compromises and risk.

In contrast, when data are considered to be owned by the organization, and stewardship is embraced, then organization-wide security policies, standards, and architecture are easier to establish. This promotes a consistent approach to data security and free flow of information among qualified users.

1.2.7 Privacy and PII

Privacy is an aspect of data security that pertains to personally identifiable information (PII). PII such as a person's name, identification number, address, telephone number, account numbers, etc. must be protected from unauthorized access. Organizations in the U.S. are required by law to disclose their policies concerning the use and disclosure of PII.

With the advent of computerized records, ecommerce, and data theft, many countries have established computer privacy laws. Some laws prohibit the use of PII for any purpose other than which it was collected. Privacy laws make data security a necessity rather than an afterthought, and they influence the classification of data and definition of data security policies. For instance, an individual may have rights over their PII and how it can be used by an organization. Laws may prohibit data from being transferred across borders to another country, which presents obvious problems for information that traverses the Internet.

1.2.8 Regulatory Concerns

There have been many laws passed over the years that address the protection of information, and responsibility of organizations to safeguard information. The list is

long, and the laws vary from one country to another. In terms of data security, the following regulatory concerns are most pertinent:

- **PII.** The Privacy Act of 1974 (U.S.), the Gramm-Leach-Bliley Act of 1999 (U.S.), the Privacy Act of 1983 (CA), the Personal Information Protection and Electronic Documents Act (CA), the Data Protection Act 1998 (UK), and other local enactments of the Directive 95/46/EC on the Protection of Personal Data in EU countries are examples of regulations established to protect personal information. In addition, the Health Insurance Portability and Accountability Act (HIPPA), which mandated greater uniformity in health information data in the U.S., also included a security rule that imposed security requirements on electronic patient health information.
- **PCI-DSS.** The Payment Card Industry Data Security Standards have been established to help prevent credit card fraud by mandating certain security controls for all organizations, worldwide, that process credit cards. See Section 3.7.6 for more information.
- **Data Loss Disclosure.** Many states in the U.S. have enacted laws that require organizations to notify citizens when their PII has been compromised.
- **Information integrity.** The Sarbanes-Oxley Act of 2002 requires U.S. public companies to implement controls on financial accounting, which includes IT assets used in the collection, processing, and storage of such information.
- **Data Retention.** Each country has regulations regarding the length of time data must be retained. Data retention is necessary for financial reporting, tax purposes, and legal protection, among other uses.

Regulatory compliance is one of many reasons an organization must consider security, particularly data security, a primary concern. It places explicit monetary value on the implementation of proper security controls.

1.3 Users

Organizations need to take a systematic approach to ensure that only the right people have access to company resources. This includes classifying users and putting approvals processes in place to regulate privileges. Privileges should be properly administered so they remain aligned, even after changing roles. Accounts and privileges for those leaving the company must be promptly revoked.

When thinking about users, security policies need to address the activities of privileged users such as systems administrators, user administrators, DBAs, architects, etc. Their activities need to be constrained so that they can never access confidential data such as PII, credit card information, or business sensitive data.

1.3.1 User Classifications

Classifying users is done in order to support confidentiality and integrity across a large user base. The intent is to manage security for groups or classes of users rather than for individuals. This makes security easier to manage, especially when there are many applications and SOA Services, and/or a lot of confidential information at hand.

Users can be classified in terms of their relationship to the organization, e.g. employees, contractors, partners, and customers. This coarse-grained classification can be used to control access to information and functions in the broader sense. For instance, certain information may only be shared among employees, while other information is available to partners, or even the public.

Users may also be classified within the organization by rank, department, group, etc. While this seems like a logical scheme, it tends to be ineffective since the membership of a department or group can consist of people with very different responsibilities. For example, a group within a bank may have a manager, administrative assistant, financial advisors, and tellers. Since they all have different job functions, from a security perspective it makes little sense to provide everyone in the group access to everything anyone in the group might need.

A more effective way to classify users within the organization is by role. A role is defined for each job function. In addition to the function it may also be necessary to include other classifications in the role, such as department or geography. For example a financial advisor in one country should not be able to access customer information from another country, and a financial advisor for retail banking may not use the same systems as a financial advisor for corporate clients. Multiple users can be mapped to a role. Likewise, users can be mapped to multiple roles. The grouping or classification of users based on their functional needs, or roles, is the most valuable technique for bringing order and structure to user management. Roles enable an organization to be agile and flexible and to efficiently apply controls to restrict access to resources.

1.3.2 Attacker Classifications

Though there isn't a well established classification scheme for attackers, it is possible to produce one based on common motivations. Attackers can come from inside the organization, in which case they have much better information to base their attacks upon, or they can come from the outside, and are more likely to use a published or self-invented exploits. Attackers may be intended users of the system or they may be outsiders, i.e. undesired users. The following list offers a simple attacker profile representation.

- **Enthusiasts,** those motivated by thrills, challenge, curiosity, or perhaps boredom. They are interested in besting an opponent (the security experts of a site) in order to prove their abilities. They are usually more of a nuisance than a threat, unless they break something in the process of gaining access to your systems. They can gain notoriety by bragging about the number and types of sites they've broken into.
- **Vandals** are attackers that want to cause destruction to the site or information. They are less interested in finding anything of value to profit from. Their notoriety comes from disrupting a site or defacing it for others to see.
- **Spies** are involved in snooping around for profit. They are generally more interested in gaining and exploiting access to information than causing destruction. In fact, the longer they can remain undetected, the more they are likely to profit. Unlike enthusiasts and vandals, spies are likely to be insiders. This is because insider knowledge of systems, databases, networks, and security mechanisms makes the job of spying easier.
- **Organized Criminals** are probably the most dangerous type of attackers because they have both the determination and the resources to cause major harm. Organized criminals are in the majority of cases acting from outside and may use techniques such intelligence gathering, for example in social networking sites, code injection etc. to get access to a company's systems.
- **Klutzes** can be considered "unintentional attackers". They are mentioned here since frequent causes of data loss, system downtime, and disclosure of information can be attributed to mental lapses. Security, in a sense, requires a bit of "idiot-proofing".

1.3.3 Policy Considerations

There are a number of considerations that can be applied to prevent misuse of systems by internal users, such as:

- **The principle of least privilege.** Users are given the least amount of privileges necessary in order to carry out their job functions. This applies to interactions between systems as well as user interactions. This reduces the opportunity for unauthorized access to sensitive information.
- **Separation of Duties.** This technique involves splitting a process across multiple roles, (people). The underlying premise is that it is harder to get two or more people to collude. Ideally the individuals are from different departments so they do not share a common reporting structure. Many regulations rely on this type of control as part of good corporate governance. This control extends to ensuring that IT staff cannot interfere with such "business process splitting" by entering or changing data directly e.g. a DBA should never be able to alter business data.
- **Vacation, Job Rotation, and Transfer.** Once way to detect and deter misuse of systems is to have a new person perform the duties of an existing worker. The new person might notice irregularities or questionable circumstances and be able to report it. The new worker might be there temporarily, i.e. filling in for someone on vacation, or might be a replacement as a result of periodic job rotations and transfers. In addition, workers that expect periodic rotations are less likely to misuse systems as they know others following behind them will eventually discover it and report them.

1.3.4 Trust

There are many aspects of trust that can be advanced through the use of security measures. These include trust between users and systems, trust between systems, and trust that an organization will act appropriately to safeguard private information. By putting security solutions in place, organizations increase the reliability and dependability of systems and the trust and confidence that all stakeholders will have in the quality of information.

Access control mechanisms increase the level of trust an organization has in the integrity of its processes and information. Audit and reporting facilities further cement this trust, both internally and externally with auditors.

Strong authentication mechanisms can further enhance trust and confidence. They require multiple forms of proof to establish identity, (see Section 2.1.1.1, "Strong Authentication & Multi-Factor Authentication"). This is generally used to authenticate users to systems. Conversely, users establish trust for systems through means such as digital certificates, verifiable via certificate authorities, (see Section 2.4.3, "Public Key Infrastructure").

In addition, displaying a watermark, (e.g. personalized photograph or other identifier) on a web site will enhance the confidence the user will have that the site is authentic. This is displayed after the user enters a login identifier, but prior to entering the password. If the watermark is missing or incorrect, then the user should consider the site to be fraudulent or compromised, and not proceed with entering a password.

Another aspect of trust is gaining the trust of users that an organization will use their personal information appropriately. Organizations will therefore typically go to great lengths to clarify this on their website. They may, for example, have opt-in or opt-out options for emailing additional marketing information to the user. This helps give the impression that personal information will not be shared with other organizations.

Trust between organizations is required in order to carry out electronic trade between business partners. It requires trust that an organization will act fairly as well as trust in their ability to secure their end of the ecommerce process. Allowing third parties to connect to an organization will depend on whether the organization trusts the quality of the third parties' controls. They may request to see what security measures are in place and carry out periodic audits.

An organization will also put policies and related training in place to ensure that individuals behave properly and maintain confidentiality with regards to sensitive information. Likewise, when they leave the company, proper agreements and procedures are in place to protect sensitive information, particularly if they are going to work for a direct competitor.

In conclusion, trust is subjective and inexact but aided by putting auditable technical controls throughout a system to enforce and demonstrate compliance with policy. Trust can be lost if information losses or breaches occur or if individuals or organization do not behave responsibly.

1.4 Common Security Strategies

This section describes some of the common strategies used for security today. Each strategy offers a theme that can be carried over into the reference architecture.

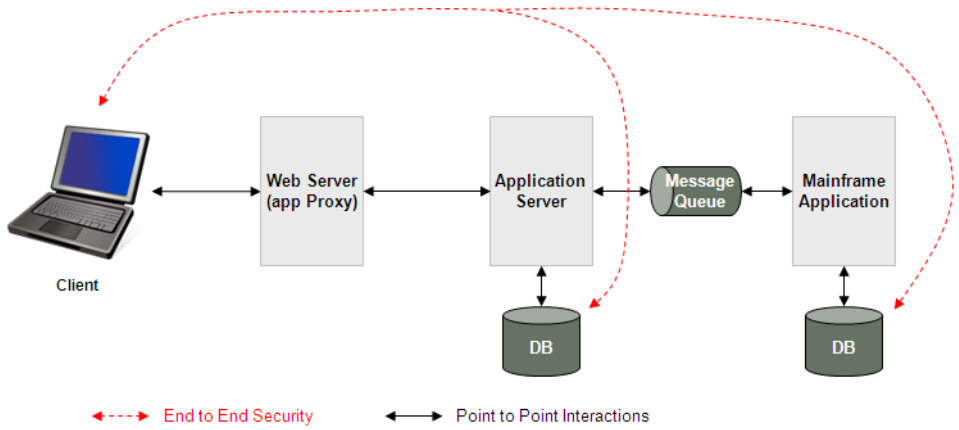
1.4.1 End to End Security

End to end security is an information-centric perspective of security where information is protected throughout the entire computing environment. That is, from the points where system interactions originate, through all points of integration, processing, and persistence.

For a typical web-based application, end to end security generally begins at the client/browser, and ends at the application database and all external dependencies of the application.

End to end security is often associated with the secure transmission, processing, and storage of data, where at no time are data unprotected. Using the example system environment in Figure 1–2, data protection would be pervasive from client to mainframe database (and back). Intermediaries in the flow of data are unable to see or alter the contents. This can be accomplished by applying message level security standards such as XML Encryption and XML Signature; and S/MIME (for email).

Figure 1–2 End to End Security



A common challenge in providing end to end security is finding a suitable way to secure data in all states and points along the processing path that does not interfere with any transmission, routing, processing, and storage functions that need to occur along the way. Sensitive data will usually need to be decrypted at certain points in order for processing or message routing to occur.

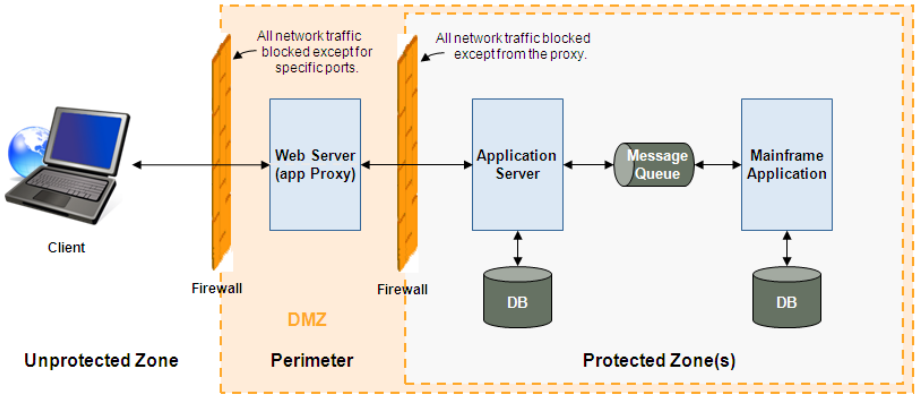
A lesser alternative to end to end security is point to point security. This is used to protect messages in transit. It assumes that other means of security are used to protect messages during processing and persistence. Point to point security is often used as a default or minimal security option in order to protect messages over insecure networks. Generally, less effort is made to protect data behind the corporate firewall. This opens up a number of vulnerabilities and risks to an organization.

1.4.2 Perimeter Security

Perimeter security is a term borrowed from military terminology by the IT community that refers to guarding resources behind a perimeter barrier. Before the age of computer networks, perimeter security consisted of mostly physical measures, e.g. secure buildings, security guards, locked computer rooms and entry codes/badges. Physical security provided a secure perimeter around computing resources. As computer networking emerged, perimeter security became more synonymous with network security measures such as firewalls, routers, network address translation (NAT), and so forth.

Many companies that have gone through the evolution of computer networking have a perimeter security strategy. In its simplest form, it consists of two zones - the unprotected zone, outside the perimeter, and the protected zone, within the perimeter. Usually the perimeter or barrier itself is expanded into a third zone, named the demilitarized zone (DMZ). The DMZ offers some protection to public-facing systems, and further protection to production systems behind the perimeter firewalls, as illustrated in Figure 1–3. The protected zone itself can be further segmented into multiple zones if desired.

Figure 1–3 Perimeter Security



While the concept of perimeter security is sound, the problem today is that networks, and network-connected devices, are ubiquitous. It can be difficult to determine exactly where the perimeter lies and where holes in the perimeter exist. In addition, one can never assume that people who are granted access to protected resources are completely trustworthy. Furthermore, government regulations do not recognize such trust relationships. It is not enough to say the sensitive personal information is protected by the corporate firewall. It must also be protected from people (employees,

contractors, managers, technicians, and administrators) that operate behind the firewall.

Perimeter security should not be confused with perimeter-based authentication, which is discussed in Section 2.1.2.1.

1.4.3 Defense in Depth

Defense in depth is a security strategy in which multiple, independent, and mutually

reinforcing security controls are leveraged to secure an IT environment. The basic premise is that a combination of mechanisms, procedures and policies at different layers within a system are harder to bypass than a single or small number security mechanisms. An attacker may penetrate the outer layers but will be stopped before reaching the target, which is usually the data or content stored in the 'innermost' layers of the environment. Defense in depth is also adopted from military defense strategy, where the enemy is defeated by attrition as it battles its way against several layers of defense.

Defense in depth should be applied so that a combination of firewalls, intrusion detection and prevention, user management, authentication, authorization, and encryption mechanisms are employed across tiers and network zones. The strategy also includes protection of data persisted in the form of backups and transportable/mobile devices. Defense in depth should take into account OS and VM hardening as well as configuration control as means of preventing attackers from thwarting the system by entering via the OS or by tampering with application files.

1.4.4 Complementary Strategies

End to end security, perimeter security, and defense in depth are compatible and complementary strategies. End to end security refers to the scope of information security - from endpoint to endpoint, while defense in depth is a strategy used to harden the environment itself. Defense in depth is compatible with perimeter security in that network perimeters (or more generically, protection zones) can make up part of the defense in depth strategy.

2

Security Concepts & Capabilities

This chapter introduces and provides background on a number of core topics that pertain to securing business solutions and data.

For the purpose of this document the following list of general topics has been established. Each general topic includes one or more specific security concepts, which are presented in subsections of this chapter.

- Identity Verification, Assertion, and Propagation
- Authorization
- Confidentiality
- Integrity and Authenticity
- Auditing
- Federation
- Web Service Security Policies
- Security Administration and Management

2.1 Identity Verification, Assertion, and Propagation

This section of the document introduces some of the key concepts pertaining to user identity. These concepts include identity verification (authentication), assertion, and propagation.

2.1.1 Authentication

Authentication is the process of verifying that a user or resource consumer is who he/she claims to be. This is generally accomplished by providing an identifier along with a password, token, or signature that is unique to the user and trusted to be secret.

Authenticating users is a common capability of any business solution. The way in which a user is authenticated varies based on the implementation of security architecture and the degree of stringency one places on proving identity.

With regards to architecture, years ago applications tended to have security logic and user information incorporated as part of the application. They acted as standalone computing resources, each in effect its own security domain. As network-based computing and common security architecture became prominent, a shift toward perimeter security emerged. The idea was to protect a group of applications, resources, SOA Services, etc., with a common set of user credentials and authentication schemes. This made it possible to share identity across solutions and reduce the overhead associated with maintaining security information.

2.1.1.1 Strong Authentication & Multi-Factor Authentication

Strong authentication is a general term used to describe authentication schemes that are more stringent than simple id/password challenges. It can refer to the use of more exotic forms of proof, such as random password tokens, smart cards, etc., or it can refer to multi-factor authentication.

Multi-factor authentication is the requirement of more than one form of proof of identity, from more than one type (factor) of proof. The three main types of factors are:

- Human Factors (something you are), which includes biometrics such as retina scans, fingerprints, etc.
- Personal Factors (something you know), such as passwords, PINs, etc.
- Technical Factors (something you have), for instance smart card, token, etc.

A multi-factor authentication scheme must include at least one form of proof from at least two of the above factor types. For instance, it could include the use of a smart card and PIN, but not a password and PIN.

Multi-factor authentication greatly reduces the risk of establishing fraudulent identity over a scheme that uses only one factor. It takes away the ability to fraudulently authenticate by obtaining any single piece of technology or password secret.

One way to achieve multi-factor authentication without requiring additional proofs from the user is to track which devices the user logs in from. The device can suffice as something the user has, for instance a laptop computer. If the user logs in from a different device, or the device is used for a different user, then additional authentication challenges may be warranted.

2.1.2 Single Sign-on

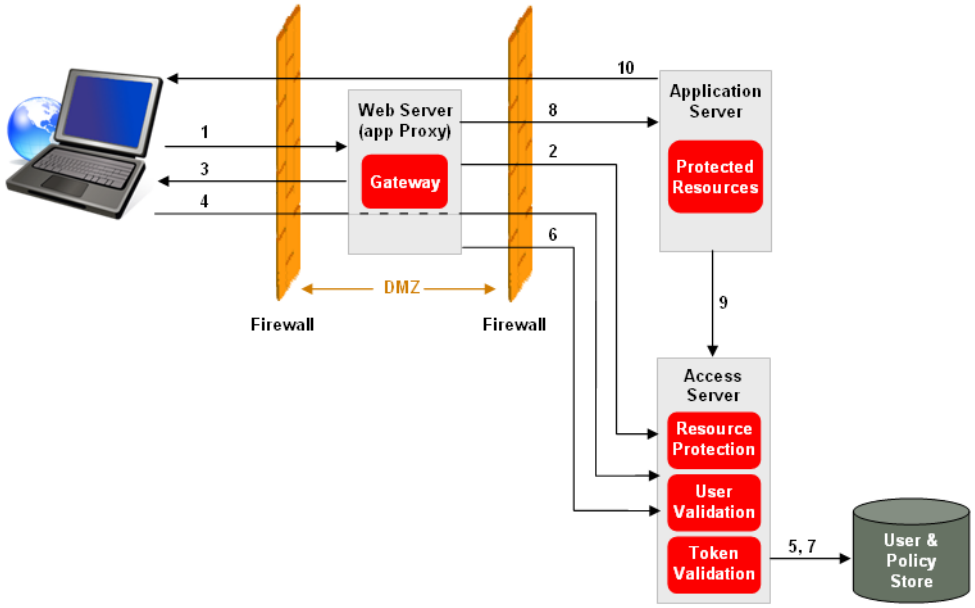
The purpose of single sign-on (SSO) is to permit users to access multiple applications and resources without having to authenticate more than once. Single sign-on solutions generally target the perimeter, or web tier, since that is the entry point into various applications. This is also known as web-based perimeter authentication. Other forms of SSO pertain to cross-domain and desktop scenarios.

The following sections briefly describe some of the more popular approaches to SSO.

2.1.2.1 Web-Based Perimeter SSO

Web-based perimeter SSO involves the use of a trusted party to vouch for authentication and a means to provide information to indicate to applications that authentication has taken place. A common scenario is provided in Figure 2–1.

Figure 2–1 Web-based Perimeter SSO



The SSO process works as follows:

1. A user attempts to accesses a protected resource hosted by the application server. The request must pass through the web proxy server gateway.
2. The gateway intercepts the request and checks with the access server to determine if the resource is protected.
3. If protected, the user is prompted for identification credentials.
4. Credentials are passed to the access server for validation.
5. The access server validates credentials against a backend directory server such as LDAP.
6. If validation is successful, the gateway can issue an authorization request for the user, action, and resource.
7. The access server fetches the policies from the directory and evaluates whether the user has access to the protected resource.
8. If the user is authorized, the request is passed to the application server along with a security token to indicate user identity and session characteristics.
9. The application server may verify the token with the issuing server.
10. The web page requested by the user is returned. Further interaction with the application server or other applications supporting the perimeter SSO mechanism leverage the token to bypass user authentication.

Note: SSO security tokens are generally placed inside an HTTP cookie. For cookie-based SSO to work, all web servers and application servers must be in the same cookie domain.

2.1.2.2 Kerberos

Kerberos is an authentication system designed to support single sign-on. In a basic configuration it consists of a Key Distribution Center (KDC) comprised of at least one

Authentication Server (AS) and one Ticket Granting Server (TGS). In order for a user to access an application or resource, it must first obtain a ticket, issued by the TGS.

To do so, the user first accesses the AS in order to become authenticated. Once authenticated, the AS issues a ticket for the user to talk to the TGS. This ticket is called the Ticket Granting Ticket (TGT), or the initial ticket. After receiving the TGT, any time the user wishes to contact a resource, he requests a ticket from the TGS. The TGS then issues tickets to talk to the various resources. The TGT has a time limit, so it is only valid for a certain amount of time.

More information about Kerberos can be found at: http://web.mit.edu/Kerberos/

2.1.2.3 Federated SSO

As previously noted, web-based perimeter-based SSO solutions require all participants to belong to the same cookie domain. The reason for this is quite simple - a cookie is used to indicate that authentication has taken place. Applications that do not belong to the same cookie domain will not be able to detect the SSO cookie.

There are alternatives to using cookies for data exchange. One method is to append data to the URL in the form of request parameters. Another is to include data in a POST operation. Both of these methods provide a means to send an assertion of identity from one domain to another. If the senders and receivers trust each other, agree on the security data exchange protocol, and can map the end user to a recognizable local identity, then SSO across multiple domains (federated SSO) is possible.

The OASIS SAML 2.0 standard has combined insights from a number of efforts pertaining to identity federation such as the Liberty Alliance, the Shibboleth project, and WS-Federation. This standard, described in Section 3.2.3, provides the message and exchange definitions needed to facilitate federated SSO.

2.1.2.4 Desktop SSO

A desktop SSO solution is one that lives on the user's personal computer and handles authentication challenges on behalf of the user. The user logs into his desktop environment, which in turn works on his behalf to authenticate to the applications he accesses. The user is no longer prompted for credentials - they are provided automatically by a process running on the desktop.

Desktop SSO can work in conjunction with other forms of SSO such as perimeter-based SSO. The only noticeable difference with desktop SSO enabled is that the perimeter SSO credential challenge is answered automatically by the user's desktop machine rather than being manually answered by the end user.

The desktop approach adds two key SSO capabilities. First, it works to limit the number of identifiers and passwords a user must remember to only one. Since PCs have become the ubiquitous end user device, the initial login is usually done to access the PC. Using the PC login to enable access to all other resources creates a SSO environment in the most literal sense.

Second, desktop SSO offers a way to log into applications that are not accessed via typical perimeter authentication. Legacy, (mainframe, packaged, etc.), and custom applications that are launched directly from the desktop are examples.

One must be careful when using desktop SSO to ensure that the desktop itself is secure. It must be sufficiently password protected at all times since anyone that can access the desktop will automatically gain access to all applications. Password strength for the desktop must match or exceed the strength requirements of the most secure applications it accesses via SSO.

2.1.3 Identity Propagation

A capability closely related to authentication and single sign-on is identity propagation. Once a user has been authenticated, the user's identity needs to be associated with his activities. This includes activities directly and indirectly initiated by the user.

For instance, if the user initiates a transaction, and that transaction involves many interactions between computing resources, then each interaction may need to be aware of the user's identity.

Likewise, for SSO to work properly, target resources must be able to identify the user. Even if they don't need to perform authentication, they still must consider authorization and auditing of requests, among other things. Therefore the HTTP request, SSO cookie, Kerberos ticket, or SAML assertion must include the identity of the user.

The need to propagate identity is heightened in SOA and BPM environments due to the distributed nature of business processes and SOA Services. Each resource in the chain of processes and services must be able to determine who the end user is and know that authentication has taken place.

Identity may be represented in different ways. Within a J2EE environment it gets represented as a Principal - a Java object that encapsulates information about an entity. It may be passed between J2EE (or other homogenous) applications in a proprietary form recognized by both applications. And it may be encapsulated within a common security token, such as a SAML assertion or X.509 certificate, when traversing heterogeneous environments.

Identity may be passed along in various ways, such as: at the transport layer (e.g., HTTP basic auth), in the message header (e.g., SOAP/WS-Security header), in the message body (e.g., an element in the request document), or as a separate request parameter. The first two options are most useful when interface protocols and service infrastructure support transport and/or header level access. In other cases, such as invoking SOA Services via plain old XML (POX) over a message bus, or when service providers only have access to the message body, it may be necessary to include identity within the message, or as a separate parameter.

Figure 2–2 Identity Propagation

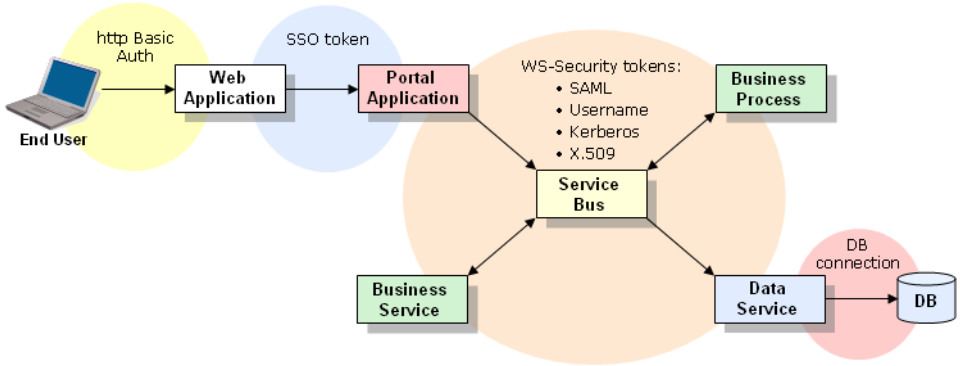


Figure 2–2 illustrates how identity is established when the user is authenticated by the web application. Identity is passed between computing entities in different ways, such as SSO tokens and WS-Security tokens. This sample interaction also highlights the need to recommend specific security standards in order to promote interoperability. A

service bus or security gateway may be used to bridge incompatibilities between SOA Service endpoints.

Identity is propagated from end to end, all the way to the database in this example. This allows each component of the architecture to evaluate security policy (if one has been established), and generate an audit entry (if policy dictates). Identity propagation is necessary to enable the principle of self-preservation, where each component of the architecture has the ability to introduce security constraints and audit activity.

Another aspect of identity propagation is the ability to identify the calling application. For instance if user Joe is logged into the Employee portal, which in turn invokes the ChangeBenefits service, it may be necessary for ChangeBenefits to know the request came via the Employee portal. ChangeBenefits may restrict access based on client application, and further restrict access to specific benefits information based on user identity.

For additional security, identity should be propagated in a way that it cannot be altered en route. Principles of integrity and confidentiality may be applied here. It may also be accompanied by proof of authentication. Together, these capabilities are essential for resources to know which user originated the request, and that the identity being provided is legitimate.

2.1.3.1 Propagating Proof of Authentication

The purpose for propagating proof of authentication is to eliminate the need for each computing resource in a distributed computing environment to re-authenticate the user. While SSO generally applies to participants in the web tier, proof of authentication also pertains to resources behind the web tier.

The reason this becomes an issue with modern computing environments that include BPM and SOA is that resource interactions must be able to traverse multiple application frameworks. Each framework, upon authenticating a user, must provide assurance to the resources it consumes that the user is valid. Otherwise, user credentials would need to be propagated, and the user re-authenticated by each resource.

For propagation of authentication to work, the recipient must be able to trust three things: the party that claims to have performed authentication, the method of authentication, and the authenticity of the object it receives claiming authentication has occurred. The degree of distrust among participants, or level of paranoia, will help determine what type of solution is required for a particular environment.

First, a decision must be made on who can make valid claims of authentication. The answer could be any entity in the trusted IT environment, or only a dedicated authentication authority. Limiting the number of entities that can make these claims increases the level of security by reducing the potential of a fraudulent entity being able to pose as a valid authority.

Second, a decision must be made on what evidence needs to be presented as proof of authentication. While some resources may permit username and password authentication, others might require tokens or digital certificates. As a result, strength of authentication should be taken into account when designing a propagation solution.

Third, a decision must be made on how evidence must be transported in order to be considered authentic. The evidence should minimally contain the identity of the authenticated user and method of authentication, and optionally could contain other information, such as the identity of the authenticating entity, time of authentication, time to live, and attributes pertaining to the user's identity (such as roles, entitlements, etc.). The evidence should be consistent across the enterprise, and signed in order to

detect if it has been altered. It can also be encrypted, if the information is sensitive and/or travels over insecure networks.

As an extension of identity propagation (discussed in the previous section), validating proof of authentication can be handled by the container. The container can intercept service requests, detect proof of authentication, verify the signature of the authority, extract the user id, and execute the business logic as the authenticated user. Likewise, subsequent outbound requests could propagate proof of authentication by having the container add the required information to the outbound request. The information could be a newly created proof (assertion) or a duplicate of what was received on the inbound request.

Propagating a single proof object, or assertion, can be susceptible to man-in-the-middle attacks and replay attacks. If a rogue entity observes an assertion, it could reuse that assertion for illegitimate requests. Possible solutions include:

- Invalidate the assertion after every request. In the case of chained SOA Services, service providers must verify each assertion they receives with the authority. The authority can invalidate assertions in its internal cache. Any future verifications with the same assertion would fail. SOA Service providers would need to obtain a new assertion in order to make subsequent service requests. This solves both types of problems mentioned above.
- Reduce and enforce the assertion's time to live attribute. This would narrow the window of opportunity to reuse an assertion. The assertion would have to be captured and reused in a short period of time (programmatically vs. manually). While this limits the potential for man-in-the-middle attacks, it's not as effective for replay attacks.
- Require the signature of a trusted service consumer (client application) in addition to the signed assertion. The caller's signature should cover the assertion to bind it to the message. If all service consumers are required to sign their request messages, then service providers can be shielded from rogue clients, thereby preventing man-in-the-middle attacks.

This solution would need to be enhanced to solve replay attacks. One option is to include a unique request id, timestamp, or sequence number in the request. The target resource could maintain a cache of ids and refuse duplicate requests. A common request id service could be created to issue unique request ids and validate all requests that are received within the security domain.

2.1.4 Fraud Detection & Prevention

Even with security measures in place for authentication, fraudulent users can gain access through a number of means including:

Phishing - Luring users to divulge sensitive information such as user ids, password, PINs, and social security numbers by tricking them into thinking they are interacting with a legitimate web site, when in fact the site is a counterfeit.

Trojan Horses - Installed without user's approval, a piece of malware designed to record keystrokes or capture screens in order to obtain sensitive information or allow malicious access or takeover of the compromised system.

Password Theft - In addition to key-loggers (mentioned previously), theft can occur either by watching users type sensitive information, or by guessing passwords. "Over-the-shoulder" spying can occur in person or via surveillance camera. Password guessing can happen in a number of ways including dictionary attacks and guessing obvious words like family names and pet names.

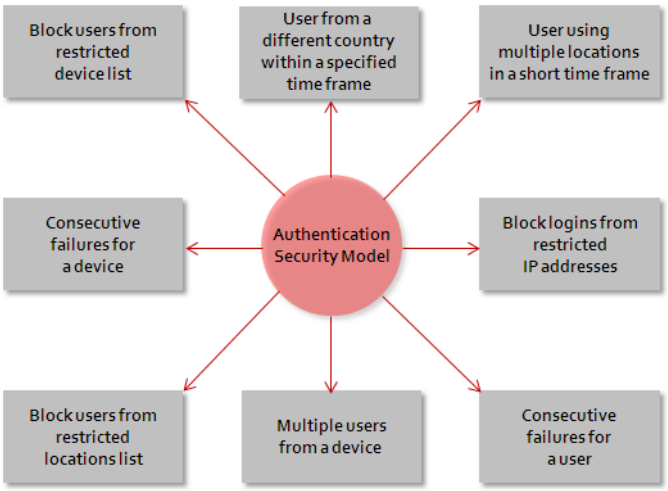
Man-In-The-Middle Attacks - These attacks involve combing through captured data transmissions for information that can be useful. They require the hacker to gain access to a portion of the network in order to capture the data. Captured data can be used to perform new fraudulent transactions or simply resent, (a.k.a. Replay Attacks), in order to cause problems related to duplicate transactions.

In order to avoid these types of attacks a security framework must have capabilities such as:

- Logging in users without the need to type passwords or PINs
- Dynamically challenging the user for different information, e.g., asking a random question for which only the user will know the answer
- Encrypting and signing transmissions from the client to the back end server
- Detecting replays using embedded transaction ids or timestamps
- Presenting proof to the user that the site they are visiting is authentic

The security framework may take into account a number of factors including the device used to access the site, the location the device is sending from, and the time since the user last logged in. An example authentication model to detect potential fraud is shown in Figure 2–3. Each login attempt would be checked against these conditions and failed or further challenged if any conditions were present.

Figure 2–3 Authentication Model



Analysis of these conditions must be done in real time when the user is attempting to access the system. The system could be set to deny access if any of the conditions exist. However, this would likely anger users that don't follow a specific access pattern. An alternative is to use some form of intelligent software to evaluate the conditions and adapt the authentication process based on risk analysis. This would allow the system to become more stringent when risk is considered high and less of a burden when conditions are perceived to be normal.

2.2 Authorization (Access Control)

Authorization is the process of granting or denying requests by a party (e.g. client or user) to perform actions on a resource. It is generally performed by comparing rights granted to the user with actions the user is attempting to perform. Access decisions

have been divided into three topics including coarse-grained, fine-grained, and data field level access control.

2.2.1 Types of Access Control

2.2.1.1 Discretionary Access Control

Discretionary access controls (DAC) are the controls placed on resources at the owner's discretion. Unlike mandatory access control, it is not subject to rules of resource classification or personal rank. Access can be granted at the owner's discretion, and possibly even passed on from one person to another at their discretion.

2.2.1.2 Mandatory Access Control

Mandatory access controls (MAC) are controls based on policies. Policies define rules by which a user, or subject, is permitted access to a resource, or object. Unlike DAC, access is universally applied based on pre-defined rules, e.g. non-discretionary.

2.2.1.3 Content Dependent Access Control

Content dependent access control involves restricting access to content, such as documents and emails, based on embedded keywords or certain assigned metadata. It works by inspecting the content and applying rules to determine if access is permitted. This approach is taken by many Data Loss Prevention solutions. It is possible to combine content dependent access control with role-based access control in order to restrict access to content by established roles.

2.2.1.4 Role-Based Access Control (RBAC)

Given the potentially large number of users of a system, access privileges are generally not assigned at the user level. Instead, users are assigned to groups (mimicking the organizational structure of a company), or roles (defined based on job functions that users perform), or some combination of the two. Access privileges are then assigned to groups and/or roles. The most natural case is that they are assigned to roles, since roles align more closely with operations users naturally perform to accomplish their job. The industry term for this is Role-Based Access Control (RBAC). RBAC is more flexible than defining access rights based on usernames or static groups and enables an organization to be more versatile when allocating resources.

With RBAC the system must determine if the subject (user or client) is associated with a role that has been granted access to a resource. This process of user to role ascertainment is called role mapping.

2.2.1.5 Attribute-Based Access Control

There are times when access should be based on characteristics the user has rather than the organization or roles to which the user belongs. For instance, a customer with premium status might be granted access to exclusive offers, and a sales representative that has achieved his target sales revenue might have access to certain perks. Such levels of status vary over time, making it difficult to manage access based on relatively static group or role assignments. Attribute-based access control offers a more dynamic method of evaluation. Decisions are based on attributes assigned to users, which are free to change as business events unfold. Access policies define the attributes and values a user must have, and access decisions are evaluated against the current values assigned to the user. Attributes can be used to support both course-grained and fine-grained authorization.

Attribute-based access control can also be used as the underlying enabler of content personalization. Where RBAC (or other methods) might enable access to a web page, specific content on the page is made available based on user attributes

2.2.1.6 Coarse-Grained Access Control

Coarse-grained access control is generally performed at the operation level, i.e., deciding if the user is permitted to perform a function or access a resource. It involves very simple access policies such as "grant role X read access to function Y". No further stipulations are applied. This form of access control is quite common in the industry and relatively easy to administer. It supports RBAC as well as control based on user identifiers, group, rank, etc.

2.2.1.7 Fine-Grained Access Control

Fine-grained access control refers to the ability to control access based on factors that are more detailed than groups and roles. The main distinction is the metadata which is used to make grant/deny decisions. The metadata comes in different forms: attributes on subjects and objects, environment and system properties, external functions, etc.

For instance, a bank teller may be authorized to perform cash withdraws up to a certain amount of money, only during work hours, and only from the home branch location. The user in this case belongs to the role "bank teller", and has attributes that describe work hours and home location. The attributes are updated as needed by the identity management system in order to remain current. They, along with other parameters such as environment variables, are are factored into the access control decision at run-time to provide more precise control over user activities.

Fine-grained access control requires:

- The establishment of policy (business rules to evaluate). Authorization policies can be expressed in a standard form using XACML (see Section 3.2.4), which allows them to be imported, exported, and enforced by different authorization systems that adhere to the industry standard.
- Attributes on users and groups and a hierarchy of attribute values (for example, location values could be "Denver", "Colorado", and "US" in a hierarchy)
- The collection of conditional data points (time, location, amount)
- Evaluation of policy
- Enforcement of the decision

Often all of these features were embedded into business logic, making maintenance of entitlements a development effort. Organizations wishing to validate entitlements for compliance reasons would need to rely on developers to translate source code back into entitlements rules. Furthermore, maintaining consistent policy across an organization required the orchestration of code updates to multiple applications.

As an alternative, organization may choose to perform some or most of these functions external to the application. Ideally, only the enforcement is coded into the applications and SOA Services. Policy management, information retrieval, and policy evaluation are offloaded to an entitlements engine, which exposes a common entitlements evaluation service. The entitlements engine permits security administrators to manage access control decisions, at a far greater scope (enterprise-wide), with far greater granularity than can be achieved through mechanisms such as deployment descriptors.

2.2.1.8 Rule-Based Access Control

Rule-based access control is very similar to fine-grained access control, where access is controlled by rules defined in policies. The twist is that rules might refer to each other. For instance, access may be granted to resource/function A as long as it is not also granted to resource/function B. This form of control can be used to ensure that a group or individual is not given privileges that create a conflict of interest or inappropriate level of authority. For instance, the approver of expenses or purchases cannot be the same as the requestor.

2.2.1.9 Data Field Level Access Control: Data Redaction

Redaction pertains to the function of reducing or limiting the set of data returned to a user based on the entitlements of that user. This is particularly relevant for SOA Services that are shared across multiple types of users. Through redaction, a single SOA Service can be offered to many types of users as opposed to creating separate SOA Services for each.

An example of this could be a customer record. Depending on the audience, certain fields in the customer record may not be appropriate for viewing. Personal information such as SSN and credit score may be required for a loan officer but not for a bank teller. Both users can access the same SOA Service to acquire data and the system automatically redacts personal information queried by tellers.

Redaction, like other forms of access control, benefits from centralized management of users, roles, and policy. It is also beneficial if enforcement of data security can be handled through infrastructure or security services as opposed to being hand coded into various disparate resources.

2.2.2 Access Control Categories

There are many different forms of access control, which in turn can be classified into one or more categories. The following access control categories can be used to evaluate the scope or fitness for purpose of an access control solution.

- Preventative - Controls that provide a barrier between the assets being protected and potential users. Barriers apply to both attackers and legitimate users. A foundational principle for preventative access control is to ensure that users are properly identified and authenticated.
- Detective - Detective controls are meant to record all activities. They are passive systems that are aware of events but are not designed to prevent them from happening. Audit logging is a form of detective access control.
- Deterrent - A control mechanism that helps to avoid attack based on the potential for being observed or discovered. Deterrent mechanisms are meant to be highly visible. They may consist of detective or preventative controls that provide a deliberate warning of security and consequence. For instance, when users realize that their activities are being logged they are much less likely to attempt to access functions or information not intended for them.
- Corrective - Corrective controls take affect after a security event has occurred. They generally function to help prevent the same type of event from happening again. Corrective controls may involve adjusting access control rules or changing processes to eliminate errors and oversights that resulted in undesired events.
- Recovery - Recovery controls, like corrective controls, take affect after a security event has occurred. They are designed to restore that system to a normal operating state. For example, restoring data following malicious or accidental deletion.

- Compensating - Compensating controls are introduced when the existing capabilities of a system do not support the requirement of a policy. Compensating controls can be technical, procedural, or managerial. For instance, management processes, such as staff supervision and log inspection, can be used to compensate for gaps in an access control solution. Compensating controls can be removed once the deficiencies for which they were designed to address have been removed.

2.2.3 Access Control Theoretical Models

2.2.3.1 Bell Lapadula

The Bell LaPadula (BLP) model has the goal of ensuring confidentiality of an automated information system. It uses Mandatory Access Control principles, where objects are labeled with a security classification and subjects are labeled with a clearance level. A subject of lower clearance cannot read an object of higher classification but a subject of higher clearance can read from classifications below. A higher level subject cannot send information to a lower level subject but a lower level subject can send information to a higher level subject. This model can be memorized as "Read Down - Write Up".

The intent of BLP is to prevent subjects from accessing objects of a higher clearance level, yet allowing subjects to provide information to equal or higher level subjects. This model protects the confidentiality of information.

2.2.3.2 Biba Integrity

The Biba Integrity Model is also based on object classifications and subject clearance levels as in BLP. However, Biba addresses information integrity by defining rules about information origination and dissemination. Biba states that a subject may have read access to an object if the security level of the subject is either lower or equal to the object, and write access to an object if the security level of the subject is equal to or higher than the level of the object . This model can be memorized as "Read Up - Write Down".

The Biba model prevents users of lesser clearance from overwriting information provided from a higher level.

2.2.3.3 Access Control Matrix

An access control matrix defines access privileges using a two-dimensional matrix. One dimension lists subject types (users, groups, roles, etc.), while the other lists objects. The intersecting cell of each subject and object is denoted with the access permissions to be granted, e.g. read, write, read/write, approve, etc. Fine-grained access rules may also be included in the cell.

2.2.3.4 Brewer Nash (Chinese Wall)

Brewer Nash is designed to prevent conflicts of interest where subjects might ordinarily have access to information from unrelated or competing sources. For instance, a firm that provides corporate advice on mergers and acquisitions, must guard against allowing such confidential information to be accessed by someone in the company providing investment advice.

Access restrictions can be erected dynamically such that no wall exists until a subject accesses information from one source/organization. Afterward the subject will not have access to competing sources/organizations. Therefore access control rules change based upon user behavior.

2.2.4 RBAC in a Distributed Environment

In legacy environments, roles were often established for individual applications or application domains. Permissions were then granted to roles at the application or domain level. An example of this is J2EE container managed security. Roles may be managed in the application's domain while access information is contained in deployment descriptors (or worse yet, hard coded into the application). Given the tight coupling between deployment descriptors and code, essentially it was up to the developers to configure and maintain access rights. Furthermore, since roles and entitlements were not shared across systems, maintaining a consistent access control strategy across the enterprise in a distributed computing environment presented an enormous challenge.

A solution to this problem is to remove the management of users and roles from the application domain and create centralized identity and role stores. Applications may continue to rely on container-managed security, which requires the declaration of access rights granted to roles, however the definition of roles and mapping of users and groups to them is done at a broader scope. As a result, containers continue to act as the access control enforcement point, though now they enforce rights based on roles defined external to the application itself. Since users, groups, and roles are shared across applications, centralized identity management in this way establishes continuity across the organization. The distributed nature of SOA and BPM fits nicely with this model. Business processes and SOA Services, working independently of each other, can leverage common shared identity and role information to ensure access to resources is handled consistently and correctly.

2.3 Confidentiality

Confidentiality refers to the ability to protect data from being seen by entities it is not intended for, both while stored and in transit. Sensitive data housed within a completely secure, locked down, computing environment may be considered safe enough without the need for extra security measures. However, most data eventually must travel beyond the boundaries of locked down environments and must be protected through some means. This problem has been solved through encryption, which scrambles the data so it becomes unrecognizable to everyone except those with the keys to decrypt it.

2.3.1 Encryption

2.3.1.1 Symmetric and Asymmetric Cryptography

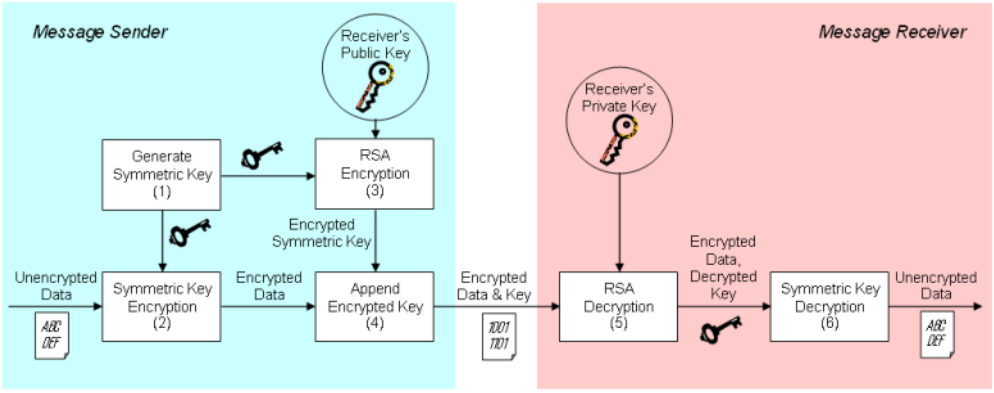
The process of encrypting data is generally available in two forms: symmetric key (secret key) cryptography, and asymmetric key (public key) cryptography. Symmetric key cryptography uses the same key to encrypt and decrypt data. Therefore the sender and receiver must each have a copy of the key. It is important that no other entity gains access to the key otherwise they would be able to decrypt the data as well. Common key algorithms include the Data Encryption Standard (DES), Triple DES (3DES), and Advanced Encryption Standard (AES).

Public (asymmetric) key encryption uses a pair of keys, one private and one public. The public key is freely distributed to any party that may wish to send encrypted data. Once encrypted, data can only be decrypted with the private key. Therefore the private key is maintained by the receiving party and is not shared with anyone else. The two keys are mathematically related, but can't be used to discover each other. A

popular asymmetric key algorithm is RSA, invented by Ron Rivest, Adi Shamir, and Leonard Adleman.

Since public key encryption avoids the need to share a common key, it is generally considered to be more secure. The downside is that it involves far more complex processing, which significantly increases the overhead involved with the encryption and decryption process. As a result, a combination of symmetric and asymmetric encryption is often used. The following diagram illustrates this process.

Figure 2–4 RSA-Based Encryption System



1. The message sender generates a symmetric key.
2. The symmetric key is used to encrypt the data.
3. The receiver's public key is used to encrypt the symmetric key.
4. The encrypted symmetric key is appended to the encrypted data.
5. The receiver uses its private key to decrypt the symmetric key.
6. The receiver uses the symmetric key to decrypt the data.

It is worth mentioning that encryption of single values, such as passwords, pose a special problem that encryption of complex data, such as documents, do not suffer. A hacker could use a public key to encrypt many single values and then compare these known encrypted values to valid data that is being sent by users. Such is the case of dictionary attacks. A large database of words (or character strings) with corresponding encrypted values is generated using a public key. If a user enters a password, and the password's encrypted value matches an entry in the hacker's database, then the hacker can determine the user's password. To thwart this type of attack, a random data value can be included with the password, which alters the encrypted value. This random value is referred to as salt. Since the password's encrypted value changes with each random salt value, a direct comparison, or dictionary attack, is no longer feasible.

A common method used to try to hack encrypted data is known as a brute force attack. This involves trying to decode encrypted text using many possible key values until the plain text message is revealed. The likelihood of success is (generally) inversely proportional to the strength of the encryption algorithm. As computing power has increased over the years, weaker strength encryption schemes have become vulnerable to brute force attacks.

Early encryption schemes used 56-bit and 64-bit keys. A 64-bit key has 2^{64} possible key values. A brute force attack on this type of key can be successful in a matter of days. To combat this, new encryption schemes offer key lengths with 128, 192, and 256 bits.

The amount of time necessary to conduct a brute force attack on these schemes is calculated to be in the billions of years.

Note: As this section describes data protection, it does not address the validity and management of keys. A discussion on key management is provided in Section 2.4.2, "Non-Repudiation / PKI".

2.3.1.2 Transport Layer and Message Level Encryption

Information traveling over a network is actually an aggregation of data from multiple layers of a protocol stack. Each layer of the stack has a specific purpose, which may be represented by one or more technologies. For example, a Web Service invocation may involve sending an XML request message via SOAP over HTTP over TCP over IP. As the request invocation is constructed, each lower layer "wraps" the higher layer's data. When the message is received, the layers are "peeled off" until only the XML request message remains, which is presented to the service provider. Response messages are returned using the same procedure.

Data encryption can occur at various points in the stack. Two common points are the transport layer and the message level. Transport Layer Security (TLS) and its predecessor, Secure Sockets Layer (SSL), provide secure transport for protocols traveling over TCP. These include secure HTTP (HTTPS), FTP, and Telnet. It uses public key cryptography to establish an encrypted session where the sender and receiver share a secret symmetric session key.

Since TLS applies at the transport layer, data is not encrypted until it is about to be sent over the network. It gets decrypted at the receiving end and then delivered to the intended recipient. One benefit of transport layer encryption is its transparency to the endpoint applications. Most application platforms handle SSL/TLS under the covers, so no development effort is required to achieve confidentiality. Another benefit is the greater efficiency of transport level encryption over XML encryption, which is often used at the message level.

The downside to TLS is that it only protects data in transit, or "point-to-point". Once data is received, it is no longer protected. Any service or application receiving the data can read, archive, or modify it. If messages need to be stored, perhaps in a database or persistent queue, they will be unprotected (until/unless they are re-encrypted by another means). If messages are meant to be received and forwarded to other destinations, data will be in clear text throughout the process except while traveling over the network.

TLS may be sufficient for some applications where the security of the application and its database(s) is not questioned. However, the lack of protection during processing and persistence may be a concern to some organizations. Particularly in an SOA environment where processing and persistence may involve multiple SOA Services provided by different groups across the company. Ideally, access to (sensitive) data should only be available to the resources that need it, not the infrastructure, intermediaries, or other destinations within a process or service hierarchy. Confidentiality should be "end-to-end" rather than "point-to-point".

For instance, a business process handling a loan application might include data containing a customer's SSN. If the SSN is only needed for credit verification, then it should only be visible to the credit verification activity. Otherwise, there is a potential for it to be mishandled at other steps in the process, or discovered if process data is persisted at any time. Message level security solves this problem by encrypting messages or elements within a message. It can be controlled such that only resources that need to act upon the encrypted elements have the ability to decrypt them. It can use symmetric key or asymmetric key algorithms to perform encryption.

Message level encryption can be used in conjunction with transport layer encryption if maximum protection is required. This is frequently done when data is transmitted across public networks. The two forms of encryption do not interfere with each other, so aside from potential performance concerns, there is no reason to avoid this practice.

2.3.1.3 Persistence Encryption

Once messages have been broken down into entities for storage in structured databases, they seldom retain any form of message level encryption. At this point it is up to the persistence mechanisms to ensure confidentiality. This involves protecting data in transit between devices (application to database, database to tape, etc.), and protecting information as it is stored in the database and on backup tapes.

Encryption at this stage is important mainly to safeguard data from people that have administrative access to the system. Historically, DBAs, system administrators, application developers, and support personnel have had the ability to view data without limitation using their administrative access privileges. While these users are generally well trusted, government regulations for information privacy reinforce the need to enforce confidentiality in this regard. Ideally, this form of encryption (and subsequent decryption) is configuration-driven, automatic, and easily managed. The drawback of database encryption is the effects it might have on performance and the possible interference with value-driven features such as partitions, keys, and indexes.

2.4 Integrity and Authenticity

This section pertains to the ability to ensure:

1. The integrity of information - that messages, data elements, documents, and other forms of content have not been modified, either in transit or at rest.
2. The authenticity of a party - that the party is not an imposter.

These concerns can be addressed through the use of digital signatures (for point 1), and Public Key Infrastructure along with 2-way SSL (for point 2). Digital signatures and PKI are discussed in this section. SSL is described in Section 3.1.1, "TLS & SSL".

2.4.1 Digital Signatures

A very useful feature of public key cryptography is the ability to encrypt a message with the private key and decrypt it with the public key. Since only the private key can encrypt the message in a way that is decipherable by the public key, one can trust that the private key's owner was responsible for encryption. Digital signatures are based on this concept.

Since performance characteristics of asymmetric algorithms generally prohibit the encryption of entire messages, a message digest is often used for digital signatures. A digest is a one way hash of the message. The digest represents the message but cannot be used to recreate it. Also, it is computationally infeasible to create a different message with the same digest. Message digest algorithms include MD5 (128 bit), SHA-1 (160 bit), and SHA-2 (224, 256, 384, and 512 bit).[1]

Digital signatures are created by computing a digest for the message, and encrypting the digest with the private key. The encrypted digest constitutes the digital signature that gets appended to the message. The recipient uses the public key to decrypt the signature, obtaining the digest, and executes the same hashing algorithm on the

[1] The MD5 hash function has been proven to have flaws that allow different messages to share a common hash value. For this reason MD5 is no longer recommended. SHA-1 is also suspected to have a mathematical weakness, though it has not been proven at this time.

message that the sender used. The hash results should match the decrypted digest. If so, then the receiver knows who signed the message, and it knows the message has not been modified since it was signed.

Digital signatures alone will not provide confidentiality, since only the digest is encrypted, (not the message text). They can be used in conjunction with encryption schemes mentioned in the previous section to provide maximum data protection. Encryption provides confidentiality, while digital signatures ensure integrity.

Though signatures can be used to detect when a message has been changed, they do not indicate when messages are replayed (repeated) by another entity. To detect replayed messages, a unique id can be incorporated into the message. The id might be a timestamp, sequence number, transaction id, or a similar unique designator. Since the id can't be changed without invalidating the signature, replayed messages would either be invalid or contain duplicate ids. The recipient can detect duplicate or invalid messages and ignore them.

2.4.2 Non-Repudiation / PKI

Non-repudiation is a condition where an entity cannot deny having performed a particular action. It is especially necessary where legal or financial issues are concerned. For example, the admission that a contract was agreed to, or the agreement to purchase goods, rely on the authenticity of the user. Since passwords can be guessed, a user could argue that an imposter made the agreement if a dispute arose. What is needed is a way to identify a user that cannot be compromised (given a reasonable amount of time and CPU power).

Non-repudiation can leverage concepts discussed in previous sections, such as encryption and digital signatures. By creating a digest, messages are guaranteed to have not been changed, and by signing the digest with a private key there is a high degree of proof that the key holder had to have originated the request. But there is still one factor to consider: how can we be sure the private key holder is who he/she claims to be? The answer to this problem is public key certificates and Public Key Infrastructure, discussed in Section 2.4.3.

2.4.3 Public Key Infrastructure

Public Key Infrastructure (PKI) is a method to manage the generation, distribution, validation, and revocation of public key certificates. Public key certificates tie a key owner’s identity to a public encryption key.

Public key certificates are issued by Certificate Authorities (CA), who act as trusted organizations that vouch for the identity of the key owner. Verisign, Entrust, and Identrus are examples of well known CAs. The certificates they provide may come in various formats. One of the most popular formats is X.509. The X.509 certificate contains the owner's public key, the CA's signature on the public key, and the CA's public key. It also includes other information such as: the certificate's version, the key owner's name, the CA's name, and a validity period.

Obtaining a public key certificate is the domain of PKI. The objective is to regulate certificates in order to avoid fraudulent claims of identity - effectively a digital form of identity theft. To accomplish this, CAs set up a process to obtain certificates that involve steps to prove identity before a certificate is issued. Though anyone can generate a set of keys, identity cannot be guaranteed without the presence of a valid certificate from a trusted authority.

As an alternative to going through the process of obtaining certificates from a CA, one might consider a self-regulated key management environment. This practice, called

Pretty Good Privacy (PGP), may suffice for organizations that want to manage their own distribution of keys and are willing to accept the security trade-offs involved with such a decision. In this case a company will act as its own CA and set up its own procedures for key management. In doing so the company must secure the distribution of the top-level certificate and ensure that no one can add trusted CAs without going through proper channels.

2.5 Auditing

Auditing provides a record of activities or transactions that have occurred in the system. It may be used for many purposes such as retracing steps of a transaction, quantifying activities in the system, identifying erroneous behavior, or attempting to determine if a specific action did or did not occur.

Auditing is an important concept for security, although is often used outside of the context of security. Without security auditing some attacks can go undetected, such as brute-force attacks, password hacking, and inappropriate activity perpetrated by insiders.

2.5.1 Auditing User Activities

One form of auditing pertains to tracking activities that end users perform. It involves the capture of data such as the user identity, type of request made, time of request, resources involved, and key parameters of the request. These data can be used for a number of purposes such as:

- Tracing activities of a user
- Tracking usage of resources
- Tracking access to, or operations performed on, specific data
- Charting usage or activity over time
- Detecting patterns of activity

In addition, auditing can be used to track invalid login attempts or attempts to access resources that the user does not have privileges for. Such purposes may be necessary when there is suspicion of improper behavior.

An essential element of auditing is the certainty one has to the actual identity of the user performing the activities. As previously mentioned, user identities and passwords can be guessed or compromised. In order for audits to hold up under scrutiny, one must be certain that the identity captured in the audit logs can be obtained by only one end user. This is a reason to consider multi-factor authentication schemes or PKI.

In addition, one must be certain that audit logs have not been altered. Therefore audit data integrity must be protected through the use of digests and digital signatures. Audit data may also be encrypted in order to protect it from being viewed by anyone other than the auditors.

2.5.1.1 Transaction Watermarking

Auditing can also benefit from transaction watermarking. This is a technique used to correlate multiple operations that are performed as a result of a single user request. A unique transaction id is generated and assigned to the initial user request and is passed along to all downstream entities that are indirectly involved in the request. The id is included in each audit entry of every processing entity so that all processing related to the request can be tracked and examined.

2.5.2 Auditing Privileged User Activities

While traditional forms of auditing targeted ordinary users and business transactions, another area of concern is the actions of privileged users. This includes administrators, analysts, developers, and architects of the IT environment. Two areas in particular are privileged database users and security administrators.

2.5.2.1 Auditing Database Administration Activity

As a matter of convenience database administration is often left wide open. That is, a DBA has full access to design and maintain the database in terms of tables, columns, rows, constraints, indexes, security, etc. The DBA also has full read and write access to all the data, even in production environments.

This in itself poses obvious security risks. Therefore, at a minimum, the functions available to a DBA, or any privileged user, should be limited to what they need to perform their job. In addition, any action taken by a privileged user of the database should be audited in a manner that cannot be undone (even by someone with such technical expertise).

2.5.2.2 Auditing Security Administrators

Another form of auditing pertains to security administration itself. It is used to ensure that role assignments, user attributes, policies, entitlements, etc. are correct and that any changes to these data are recorded. Much like activity auditing, requests to modify security settings must be captured and include the requestor's identity, the change being made, the time it was requested and made, etc.

Security administration auditing is critical to maintaining a secure computing environment. A person attempting to perform activities illegally or without permission may first attempt to modify the security environment in order to gain access to the necessary systems. Therefore it is often desirable to either wrap security management capabilities in an approval process, or send alerts or notifications when changes are made.

Security administration auditing capabilities can be quite useful to help establish compliance with government regulations regarding access to confidential data.

2.6 Federation

Federation, in this context, refers to interoperation between entities in different security domains. Domains may be in different companies, different divisions within the same company, or even different computing tiers belonging to the same IT organization. Each domain may manage its own collection of users, groups, roles, and policies. And, each domain may rely on different technologies and security mechanisms to perform authentication, authorization, and other security related concerns.

Another form of federation, identity federation, is discussed in Section 2.1.

2.6.1 Establishing Trust Between Security Domains

In order to federate authentication and user identity, a trust relationship needs to be set up between the entities involved in the interaction. For example, SSO solutions presented earlier all employed an authentication service that applications and SOA Services could trust to perform authentication correctly. This trust relationship created an environment where interoperability could occur, i.e., a circle of trust.

In order to achieve trust within an IT environment certain issues must be addressed. For instance, credentials used to perform authentication must be secure enough for each application or SOA Service; users must be recognized as being valid throughout the circle of trust; and encryption methods used to protect tickets, cookies, etc., must be universally supported.

While this may not be an issue within a single domain, it could present problems for requests that cross domain boundaries. Each domain may maintain its own SSO mechanisms, users, groups, roles, encryption keys, tokens, etc. In this case extra measures must be taken to create trust between companies, or between organizations with separate security domains.

In areas where security domains differ, infrastructure and/or services must be introduced to bridge the gap between domains. Example services may include:

- **Identity and authentication translation** - A service may need to convert one means of identity and authentication propagation to another. For example, one domain may use Kerberos tickets while another uses SAML. The SAML domain will not recognize Kerberos tickets, and vice versa. The service would need to convert between tickets and assertions, maintaining access to each domain's trusted identity source in order to create the necessary artifacts.
- **Encryption handling** - Security domains are likely to work with a set of keys when communicating with each other. This means that all data must be signed or encrypted with specific keys - perhaps not the same keys used by consumers and services within a single domain. To facilitate encrypted communications between domains, a service may need to decrypt messages and re-encrypt them with different sets of keys.
- **Identity mapping** - It is unlikely that users in one security domain will directly map to identical users in another domain. If domains can trust the translation of authentication data by a common service, then they may avoid the need to authenticate the new user identity. In this case it may be acceptable to simply map the identity from one domain to another rather than perform user authentication.
- **Role and attribute assignments** - In order to control access to resources across domains, federated identities will need to be associated with roles and attributes that are meaningful in the target domain.

2.6.2 Credential Mapping

As a result of siloed security data, users often have multiple identities. Along with each identity is a password, or proof, that is required in order to authenticate. As user requests traverse boundaries between applications, the user must somehow authenticate in order to proceed. Often the authentication process happens automatically on behalf of the user. The source system locates the appropriate credentials (id and password) for the user and performs authentication. The process of obtaining the appropriate credentials for the target application based on credentials used for the source application is called credential mapping.

The difference between identity mapping and credential mapping is that credential mapping involves authentication of the "mapped" user (which requires a password or other credential), while identity mapping does not. Credential mapping is most common when the target service does not recognize users from the client domain and does not trust the assertion of identity. It must perform its own authentication in order to establish an authenticated session. This is especially true with older legacy systems (packaged applications and mainframes), where security was completely managed in isolation. Also, when crossing company boundaries, such as in B2B scenarios.

Often this is accomplished by assigning a generic identity to the user, such as purchaser, or partner. Identity may also be assigned based on the organization or company they originated from, e.g., "Oracle user". The generic users are maintained in the target system or domain. A credential mapping service intercepts the outbound service request, maps the current user identity to the target credentials, and embeds the credentials within the outbound request. The receiver then extracts the credentials and authenticates the user. Mapping may be performed by an intermediary in order to avoid embedding such security concerns within the requestor or target resource.

2.7 Web Service Security Policies

In the context of IT, a security policy is a definition of what it means for a computing environment (system, application, SOA Service, etc.) to be secure. The general definition of policy must be recorded in a way that security architects can read and understand so they can apply it to IT assets. Quite often it involves the configuration of many endpoints, gateways, firewalls, applications, and infrastructure in order to achieve a state of security described by the policy. As such, security policy specification and enforcement mostly involves manual efforts to articulate policy and enact methods for enforcement.

The advent of Web Services, SOAP, and WS-Security specifications brought forth an aspect of security policy that has achieved a level of standardization and automation. It addresses the declaration of message-based security that is applied to Web Services. Web Service security policies identify the security requirements of a Web Service and provide a mechanism to enforce security policies so that requests must either meet the policy or be denied.

The following sections discuss security policy specification and enforcement as it pertains to Web Services.

2.7.1 Policy Specification

In order to articulate the security requirements of a Web Service, the service provider must create a security policy. A policy may require the user to be authenticated, and may require proof of authentication in a certain form. It may also require the encryption of message data, and/or the inclusion of a digital signature. This information can be captured in WS-SecurityPolicy format (see Section 3.3.5) for use with services that adhere to WS-* standards.

Policies may be unique to a particular service or may be common to multiple services. In order to improve consistency and reduce the overhead involved in maintaining policies, it is good practice to develop policies that can be associated with many services.

Policy specifications are attached to, or referenced by, the Web Service interface. This allows the security policy to be discovered along with the service itself. If a single policy is used by more than one service, then associating the policy by reference helps avoid the need to create multiple copies of the same policy.

2.7.2 Policy Discovery

Just as a client must be able to discover a service in order to use it, it must be able to discover the security requirements of the service as well. The method used to discover policy depends on when policy conformance takes place. If conformance is handled as part of the software development effort, then policy discovery must occur at design-time. The developer or security representative must have access to policies from a development environment and must ensure that the client adheres to the

policy. However, if conformance happens at run-time, then discovery can occur when the service is initiated. Policies must be available in the production environment and must be retrieved and processed with as little performance overhead as possible.

2.7.3 Policy Conformance

Policy conformance pertains to the client, either at design-time or run-time, and involves the production of a service request in accordance with the security policy of the target service. A design-time metaphor would involve the implementation of client side security logic to satisfy policy specifications. Changes in the service's policy would likely require a code change to the client. A more elegant solution would involve the automatic processing of policy information to produce the desired request semantics. Processing would happen transparently, without the need for any custom client code. Ideally, the client would be able to directly consume policy specifications provided by the service and make adjustments in accordance to policy changes.

2.7.4 Policy Enforcement

In addition to specifying the security policy, services must be able to ensure that policy requirements are being met. Enforcement can be handled by the service logic itself, or better yet, handled by service infrastructure or framework components. Similar to the run-time policy model just described, an ideal enforcement solution would provide a means to automatically process policy information and detect compliance or non-compliance to policy without custom code or coding changes.

Policy enforcement may be handled by the service provider, or by an intermediary (proxy) such as an ESB. The proxy model provides certain advantages, such as:

- Separation of concerns. It allows service providers to focus on service logic without having to include policy enforcement concerns as well. This may shorten service delivery times (particularly for code revisions) depending on the development and testing effort that may be avoided.
- Central point of enforcement. If all policy enforcement is performed by the ESB, then security infrastructure and design may be simplified by the use of a common pattern and technology stack. Consistency should be improved vs. the alternative of having every service provider responsible for its own enforcement paradigm.
- Offloading enforcement overhead. Offloading the enforcement reduces overhead on the services themselves, including the processing of requests that do not comply with policy and are therefore rejected before they reach the service provider.

Although the proxy model has advantages, it may not always be feasible to use this pattern. Doing so requires the configuration of infrastructure to ensure that services can only be accessed via the proxy, e.g., only the proxy is permitted direct access to the service provider. While this is often the desired invocation pattern, for security purposes one must protect against those wishing to circumvent it. Solutions for this may include:

- Firewalls and routers. Services are isolated from consumers by means of network components that only permit access via a service proxy. This works as a clean and easy solution when services are autonomously packaged and deployed. It is less effective where services are bundled within applications, and where service consumers and providers are mixed together in a common deployment.
- Certificates, keys, and signatures. The service provider requires proof that the consumer is a proxy. Proof can be in the form of mutual authentication (i.e. 2-way

SSL), or a field that is signed or encrypted by the proxy's private key. This method is quite effective but requires proper configuration of every (protected) service.

Both solutions mentioned above assume that a proxy will always exist. There may be cases where proprietary interfaces are required that are not compatible with the proxy pattern, such as high-speed language-specific protocols and interfaces designed to support distributed transactions with two-phase commit capabilities. In these cases it may be more feasible to use service provider infrastructure or endpoint agents (see Section 5.1.3) to support policy.

2.8 Security Administration and Management

A common desire with regards to IT security is the ability to provide a consistent, unified approach across the enterprise. Doing this means breaking down barriers between applications that are caused by proprietary, standalone security implementations, and moving toward a framework of shared security resources and services. This enhances the ability to automate provisioning, synchronize resources, and effectively audit and manage enterprise security.

Creating a unified security framework involves the holistic management and administration of security data, and the dissemination of those data to the various systems that enforce security. This section introduces key capabilities related to security administration and management.

2.8.1 Identity Management

Identity Management (IdM) includes the capability to store, manage, synchronize, provision, administer, and audit security data related to user identity.

2.8.1.1 Directory Services

Directory services provide a means to associate attributes with names, much like a telephone directory associates phone numbers with phone subscribers. A common standard for electronic directory services is X.500. It defines the model by which names and attributes are organized. This standard can be used for all sorts of directory lookups, including user credentials.

In the context of security, user identities are associated with passwords, roles, attributes, etc. A common implementation of security directory service is Lightweight Directory Access Protocol (LDAP), which defines a standard directory access protocol for X.500 directory services over TCP/IP networks. Security data is most frequently stored in an LDAP directory, although custom solutions and proprietary implementations also exist - particularly in legacy applications. LDAP provides a standard mechanism for storing and accessing identity data such as user credentials (for authentication), access privileges (for authorization), and profile information.

It is quite common for organizations to have similar security data stored in multiple directories. Reasons for this include the fact that some legacy applications include their own built-in security data management, and mergers and acquisitions result in duplication of resources. In order to handle these redundancies, organizations can either attempt to synchronize security data across systems, consolidate resources as is technically feasible, or aggregate security data into a single virtual representation.

Each approach has benefits. Data synchronization keeps data of similar type consistent across multiple physical stores. Consolidation reduces the cost and complexity of maintaining multiple physical stores where they are no longer necessary. And virtualization allows organizations to combine different user information from disparate sources to create a holistic representation of security data for that user.

2.8.1.2 Identity Provisioning

Given the large number of applications found in a typical IT environment, provisioning and maintaining user identity information can be quite a burden. The task involves making sure the identities and attributes persisted in each security data store are correct, consistent, and up to date with the current set of users and their current roles, responsibilities, and entitlements.

Identity provisioning includes activities such as:

- The creation of directory entries for users
- Assigning attributes, roles, and privileges to users
- Setting and resetting passwords
- Altering attributes, roles, and privileges as job functions and organizations change
- Altering security data based on changes to the IT landscape, e.g., the addition of new computing resources that require new attributes and entitlements, or the removal of legacy systems and associated security data
- Removing users in a timely manner when they leave the company or should no longer have access to resources
- Ensuring that accounts and privileges are correct and consistent across the enterprise
- Detecting accounts that are created outside of normal channels (rogue accounts) and sending alerts when such accounts are detected
- Automating repeatable administration tasks

Identity provisioning may include workflow processes to handle activities such as the creation of new users, requests for additional access, the removal of users, and password resets. Workflow processes help by adding structure to the activities. They allow ordinary users to initiate provisioning processes which may trigger corresponding approval processes that involve the appropriate levels of management. Processes also coordinate the technical aspects of provisioning such as updating directories and propagating security changes to other systems and applications that maintain their own copies of security data.

2.8.2 Role Management

Role management involves many of the same capabilities as identity management. Like user credentials, role assignments must be created, stored, disseminated, synchronized, managed, and audited. These operations should be centrally managed in order to ensure consistency across all of IT, although administration may be delegated according to how each organization wants to manage its resources.

For role management to be effective, security infrastructure must provide the means to map complex organizational structures and resources onto a set of roles. These roles represent privileges granted to users based on the job functions they perform. Roles may affect access to many resources including applications, business processes, SOA Services, and data.

In a modern computing environment where strategies such as SOA, BPM, and BI have been adopted, it is crucial to establish a consistent set of access rights for resources such as SOA Services, data, and process activities. Doing so enables resource sharing across an organization, which in turn drives greater ROI and efficiencies. Otherwise each of these strategies tend to either be limited to a departmental scope or provided at a greater scope with security capabilities disabled.

Just as identity management involves provisioning and approval processes as well as integration with applications in order to disseminate data, role management includes the same capabilities. Role management activities can be integrated into the identity management processes in order to ensure that resource access assignments are instituted at the same time that user credentials are established.

2.8.3 Entitlements Management

Entitlements management pertains to the way low level security rules are handled. Entitlements may be hard coded into applications or may be centrally managed through an entitlements engine. Since important goals of IT include agility, consistency, and reuse, an entitlements engine is a much more favorable approach. It provides a way to manage this aspect of security policy from a central point in a consistent way. It also allows changes to be made via configuration; without the usual code/test/deploy maintenance cycles.

2.8.4 Attestation

Attestation, with respect to identity and role management, is the process of certifying access rights to particular systems and sensitive data. It is an essential part of demonstrating compliance with government regulations such as Sarbanes-Oxley, Gramm-Leach-Bliley, and HIPAA.

The process required for attestation depends heavily on the degree of visibility and automation an organization has with identity and role management. If security data is managed as individual silos, then collecting, aggregating, and attesting to access privileges can be a labor-intensive, manual process. However, if security data is managed centrally, then much of the work can be automated including the reporting of user access and any subsequent adjustment of privileges based on findings.

2.8.5 Policy Management

Policy management pertains to the management of rules and the software and infrastructure that supports those rules. Security policy governs the way various aspects of security need to be handled. Policy may dictate such things as:

- The types of authentication mechanisms to use
- Password restrictions
- Guidelines on data encryption and strength of keys
- Entitlements rules and mapping to roles
- Auditing requirements
- Session timeouts
- Data classification access rules
- Storage and backup encryption requirements
- Rules for handling PII

2.8.5.1 Centralized Management

In general, policy management is greatly simplified when common security infrastructure and services are used rather than disparate standalone implementations. It reduces the number of entities to manage as well as the complexities involved in trying to map policy to different security schemes used by different systems. Though some applications, particularly legacy mainframe and packaged apps, may not

support external security capabilities, the adoption of a common security framework will support the migration toward centralized management.

2.8.5.2 Delegated Administration

While policy management benefits from consolidation and centralization, the administration of policy may need to be handled by multiple individuals from various organizations. The gains made through rationalization should not eliminate the efficiencies of having policies managed by the persons most knowledgeable about them. Likewise, security should not be compromised by allowing every security policy administrator to have access to every policy, regardless of the organization(s) those policies pertain to.

The goal for administration, as it applies to centralized management, is to be able to delegate access for specific areas of policy to the responsible individuals or organizations. That provides an efficient and effective means of administration, and at the same time supports a unified management platform.

2.8.5.3 Localized Decision Making & Enforcement

Policy enforcement is the run-time function of granting or denying access to resources. It enforces the outcome of run-time security decisions. Since decision making and enforcement add to processing overhead and overall response times, they must be handled in the most expedient manner. In general, the closer decision making and enforcement can happen to the resources being protected, the more efficient the process can be. For instance, a local Java call can be executed much faster than a remote RMI or SOAP call. Therefore, while policy management may best be handled centrally, policy decision making and enforcement is usually distributed, or local to the protected resources.

An exception to this rule is the case of policy decision points (PDPs) for computing nodes located outside the secure environment. For example, web servers located in the DMZ might leverage a central PDP, deployed behind a firewall. Policy enforcement is still local to the web servers but decisions are made remotely.

2.8.6 Run-time Threat Detection & Analysis

This aspect of security management deals with the analysis of run-time security events and responsive actions to prevent security breaches from occurring. It requires the collection of security events such as login attempts, failed logins, blocked access, etc.

Events should include information on the device used for access and the general location of that device. Such information is used to establish patterns of normal behavior, which in turn helps to identify abnormal behavior. Administrators must determine, either through manual evaluation, or automated rule-based analysis, when abnormal behavior occurs. They must then be able to lock user accounts, restrict privileges, and/or terminate sessions in order to prevent further potentially fraudulent activities.

Risk analysis ties into other security capabilities in that it requires information that must be gathered at the point of authentication and authorization. This information is used to analyze risks. Once a risk is identified, the solution must interact with identity management infrastructure in order to block logins and remove access to all systems the user might have access to. Likewise, if a user account needs to be unblocked, restored, or eliminated, identity management systems can provide the means to do this.

2.8.7 Data Security Management

For the purpose of this document, data security refers to the subset of information security that includes structured data. Managing the security of structured data involves tasks such as:

- Managing data classifications, as well as the associated access assignments. Data classifications need to be revisited from time to time as the business grows and regulations change. They must be examined (or re-examined) as data sources are deployed and/or updated. Access assignments must be reviewed periodically to avoid unnecessary exposure. The infrastructure must provide a means to tag or label data structures according to classifications, and a means to audit access assignments and changes.
- Managing information privacy. This particularly pertains to personal information and credit card / account data affected by government regulations. Infrastructure must provide a means to identify these elements, assign confidentiality and integrity policies, report and manage access, and audit changes.
- Managing and auditing privileged user access. Privileged users must be able to perform their job functions, without having complete access to all information maintained by the system.
- Maintaining confidentiality of persisted data including backups, and managing the lifespan of data so that is does not exceed the organization's maximum retention policies. The latter point relates to minimizing liability both in terms of breach of security, as well as retaining records that could be used in a legal case against an organization for longer than is legally necessary.

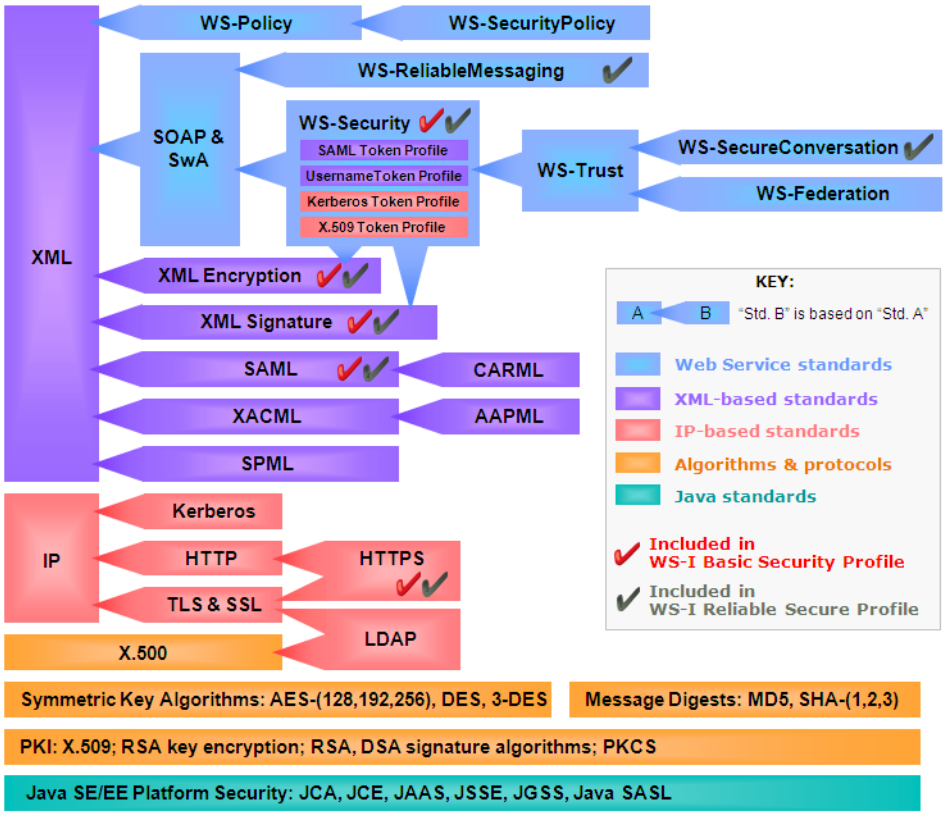
3

Common Security Standards

This chapter introduces some of the most common security standards available today. It is not meant to be an exhaustive list of everything that pertains to security, but rather a look at many of the newer and/or most widely adopted standards that support a modern computing environment.

A number of technology standards and specifications have been created to help provide security for SOA in an interoperable way. Some standards are based on SOAP, and therefore can be used to provide security for Web Service interactions that leverage the SOAP standard. Other standards are based on XML, which enables them to be used for Web Service (SOAP) interfaces as well as plain XML messaging interfaces. The following diagram, Figure 3–1, illustrates how these standards relate to, and extend from, each other.

Figure 3–1 Common Technology Standards for Security



In addition to technology standards, regulatory and compliance standards must be considered when defining a security architecture. The following sections provide a brief overview of the common technology, regulatory, and compliance standards. Links have been provided to additional reference material.

3.1 IP-Based Security

This section includes a few of the high level standards used to protect data as is transmitted over unprotected networks. For the sake of brevity, only open standards related to securing TCP networks are included.

3.1.1 TLS & SSL

Transport Layer Security (TLS) is a standard designed to provide secure transport for protocols built on top of TCP. Examples include HTTP, FTP, and Telnet. TLS originated from Secure Sockets Layer (SSL), which was developed by Netscape as a way to secure transactions over the Internet. Embedded into popular web browsers, it quickly became the de-facto standard for securing messages over public TCP networks.

TLS (and SSL) provide a means to authenticate endpoints, encrypt message payloads, and negotiate algorithms used in the authentication and encryption processes. It supports a number of algorithms such as RSA and Diffie-Hellman for key exchange, RSA and DSA for authentication, and RC4, DES, and Triple DES symmetric encryption ciphers. Authentication is usually performed using X.509 certificates.

With TLS/SSL between a client web browser and web server, the web server must provide a certificate in order to prove its identity. The certificate should come from a trusted certificate authority (CA) in order for the client to be confident of the server's identity. This allows the user to supply confidential information such as identities, passwords, and credit card information with some degree of certainty that the receiver is authentic. Authentication is usually one way, e.g. the client remains anonymous unless or until logging into the server application. Two-way (mutual) authentication is also supported, which requires the client to provide a certificate as well.

When a TLS/SSL session begins, the client and server must first negotiate algorithms for key exchange, authentication, and encryption. Once the algorithms have been decided, encryption keys are exchanged and authentication takes place. Following that, messages transmitted between the client and server are encrypted and decrypted. Encryption and decryption are performed automatically without any end user involvement. This allows data to be transmitted confidentially in a manner that is transparent to the sender's and receiver's application layer.

More information on TLS can be found at
http://www.ietf.org/html.charters/tls-charter.html

3.1.2 HTTP & HTTPS

Hypertext Transfer Protocol Secure (HTTPS) refers to the use of Hypertext Transfer Protocol (HTTP) over a secure transport layer such as SSL and TLS. Since HTTP is primarily used as a method to format and display web-based information that contains hyper-links, HTTPS is most commonly used to secure traffic between web browsers and web or application servers.

Since TLS/SSL are transport level security protocols, HTTPS applies to message transport as well. It does not provide a means to secure messages that are not in transit. Once a message arrives at its destination, decryption takes place. Other forms

of security are required in order to protect messages beyond the "front door" of the receiving end's computing environment.

3.1.3 Kerberos

Kerberos is an authentication protocol for IP networks that provides secure mutual authentication between clients and servers. It was developed by the Massachusetts Institute of Technology (MIT) in the 1980's. A free version of software that implements this protocol is available from MIT as well.

Kerberos uses a trusted third party to establish authenticated sessions for clients and issue tickets that enable authenticated access to resources available via the network. It is considered a protocol for "trusted hosts on an untrusted network" as it uses encrypted messages to establish authenticated sessions with hosts that are known and trusted by the Kerberos servers. Since end users do not need to re-authenticate to each resource, Kerberos also acts as a form of single sign-on (SSO).

The trusted third party, e.g. Kerberos Distribution Center (KDC), is comprised of an authentication server (AS), ticket granting server (TGS), and database. When a client wants to interact with a resource, it must first contact the authentication server by sending a user identifier. The user's password (stored in the database and known to the client) becomes the secret key used to encrypt a response that includes a session key for the TGS. The client then contacts the TGS to obtain a ticket that is required in order to access a resource. All interactions are encrypted using session keys that are established throughout the authentication and ticket granting process. Tickets (and keys) expire after a specified period of time - usually 8 hours - to reduce the amount of time available to crack the encryption keys. This requires all participants to keep their times synchronized well enough to work with ticket issuance and expiry timestamps.

Further information on Kerberos is available at http://web.mit.edu/kerberos/

3.2 XML Security

This section describes some of the key security standards based on XML and/or designed to protect XML messages.

3.2.1 XML Signature

Digital signatures provide a means of identifying the signer of a message as well as the ability to ensure that the message has not been modified since it was signed. Signatures generally involve the creation of a one-way hash of the message, also known as a digest. The digest is encrypted with the signer's private key. The recipient decrypts the digest using the signer's public key, to uncover the original hash code. It then performs the same one-way hash algorithm to determine if the current hash value matches the value provided by the sender. If both values match, then the contents have not been modified.

The XML Signature specification defines XML syntax and processing rules for creating and representing digital signatures. It defines how data is signed and how the resulting signature should be represented in XML. It may be used to sign an entire XML document or selected parts (elements) within the document. It relies on existing algorithms for signatures and digests, and supports several certificate types such as X.509.

More information on this specification can be found at:
http://www.w3.org/TR/xmldsig-core/

3.2.2 XML Encryption

Encryption is a means to provide confidentiality by encoding a message so that only the intended recipient can decode and understand it. XML Encryption defines a process for encrypting and decrypting information and an XML syntax used to represent the encrypted content and information that enables an intended recipient to decrypt it. XML Encryption supports the encryption of entire XML documents or individual elements within a document. It supports existing encryption algorithms and addresses both symmetric and asymmetric cryptography.

More information on this specification can be found at:
http://www.w3.org/Encryption/2001/

3.2.3 SAML

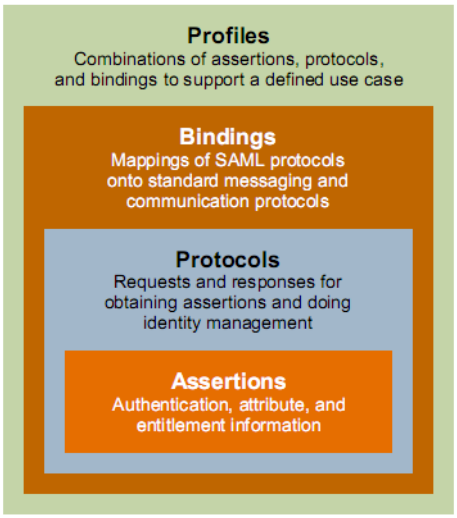
The OASIS Security Assertion Markup Language (SAML) standard defines an XML-based framework for describing and exchanging security information. Security information is stated in the form of assertions (claims) made by a trusted authority. All entities that trust the authority can use SAML assertions generated by that authority as valid claims of identity, authentication, or authorization. SAML is especially useful in a distributed computing environment where such security information needs to be propagated across the enterprise in a standard, language-neutral way. Typical use cases where SAML applies include:

- Inter-Domain SSO. SSO solutions deployed for a localized domain often exchange state information in a browser cookie. These implementations are limited to the scope of the DNS domain as cookies are not visible across domains. SAML offers alternatives solutions that do not have this limitation.
- Federated Identity. SAML offers a common standard for the exchange of identity information between entities that may not share the same authentication and authorization schemes, technologies and products. It is based on widely used standards such as XML, with support for HTTP and SOAP bindings.
- Web Service Security. Identity propagation is a critical component of secure Web Service transactions. WS-Security includes a token profile for the exchange of identity information using SAML assertions.

The SAML standard is officially described in *Security Assertion Markup Language (SAML) V2.0 Technical Overview,* OASIS Draft, February 2007, available here. SAML V2.0 represents a convergence of V1.1 and the Liberty Identity Federation Framework (ID-FF). It differs from previous SAML versions in that it incorporates more capabilities, and clearly defines more "layers". It is also not backward compatible, e.g. version 1.0 and 1.1 implementations are not interoperable with version 2.0.

This section includes excerpts from the SAML Technical Overview to introduce key concepts such as assertions, protocols, bindings, and profiles. These concepts, depicted in Figure 3–2, are defined in the following sections.

Figure 3–2 SAML Concepts



3.2.3.1 SAML Assertions

SAML defines the syntax and semantics for creating XML-encoded assertions to describe authentication, attribute, and authorization (entitlement) information, and for the protocol messages to carry this information between systems. A brief description of the three SAML assertions is provided below.

Authentication Assertion - Generated by the authority when a subject successfully authenticates. It includes identity of the issuer and the principal, time of authentication, and how long it is valid. Many authentication methods are supported, including: passwords, Kerberos, hardware tokens, certificate-based client authentication (SSL/TLS), X.509 public key, PGP, XML digital signature, etc.

Attribute Assertion - Generally issued by the authority in response to a request containing an authentication assertion. It contains a collection of attribute name/value pairs, in addition to identity and other elements. Attribute assertions can be passed to the authority when authorization decisions need to be made.

Authorization Decision Assertion - Issued by a policy decision point (PDP) containing the result of an access control decision. Authentication and attribute assertions may be provided in order to make authorization decisions. The resulting authorization assertion is used to claim access to protected resources. It includes the decision (Permit or Deny), along with the resource URI being accessed, and the action that the principal is authorized to perform.

3.2.3.2 SAML Protocols

Protocols define the structure and contents of messages that comprise the SAML requests and responses between participants. SAML 2.0 includes the following protocols:

- **Authentication Request Protocol**: Defines a means by which a principal (or an agent acting on behalf of the principal) can request assertions containing authentication statements and, optionally, attribute statements.
- **Single Logout Protocol**: Defines a mechanism to allow near-simultaneous logout of active sessions associated with a principal. The logout can be directly initiated by the user, or initiated by an identity provider (IdP) or service provider (SP) because of a session timeout, administrator command, etc.

- **Assertion Query and Request Protocol**: Defines a set of queries by which SAML assertions may be obtained.
- **Artifact Resolution Protocol**: Provides a mechanism by which SAML protocol messages may be passed by reference using a small, fixed-length value called an artifact. The artifact receiver uses the Artifact Resolution Protocol to ask the message creator to dereference the artifact and return the actual protocol message.
- **Name Identifier Management Protocol**: Provides mechanisms to change the value or format of the name identifier used to refer to a principal. The protocol also provides a mechanism to terminate an association of a name identifier between an IdP and SP.
- **Name Identifier Mapping Protocol**: Provides a mechanism to programmatically map one SAML name identifier into another. It permits, for example, one SP to request from an IdP an identifier for a user that the SP can use at another SP in an application integration scenario.

3.2.3.3 SAML Bindings

Bindings define how SAML assertions and request-response protocol messages can be exchanged between systems using common underlying communication protocols. The bindings defined by SAML 2.0 are:

- **HTTP Redirect Binding**: Defines how SAML protocol messages can be transported using HTTP redirect messages (302 status code responses).
- **HTTP POST Binding**: Defines how SAML protocol messages can be transported within the base64-encoded content of an HTML form control.
- **HTTP Artifact Binding**: Defines how an artifact (described above in the Artifact Resolution Protocol) is transported from a message sender to a message receiver using HTTP. Two mechanisms are provided: either an HTML form control or a query string in the URL.
- **SAML SOAP Binding**: Defines how SAML protocol messages are transported within SOAP 1.1 messages, with details about using SOAP over HTTP.
- **Reverse SOAP (PAOS) Binding**: Defines a multi-stage SOAP/HTTP message exchange that permits an HTTP client to be a SOAP responder. Used in the Enhanced Client and Proxy Profile and particularly designed to support WAP gateways.
- **SAML URI Binding**: Defines a means for retrieving an existing SAML assertion by resolving a URI.

3.2.3.4 SAML Profiles

SAML profiles define how the SAML assertions, protocols, and bindings are combined and constrained to provide greater interoperability in particular usage scenarios. The profiles defined by SAML 2.0 are:

- **Web Browser SSO Profile**: Defines how SAML entities use the Authentication Request Protocol and SAML Response messages and assertions to achieve single sign-on with standard web browsers. It defines how the messages are used in combination with the HTTP Redirect, HTTP POST, and HTTP Artifact bindings.
- **Enhanced Client and Proxy (ECP) Profile**: Defines a specialized SSO profile where specialized clients or gateway proxies can use the Reverse-SOAP (PAOS) and SOAP bindings.

- **Identity Provider Discovery Profile**: Defines one possible mechanism for service providers to learn about the identity providers that a user has previously visited.
- **Single Logout Profile**: Defines how the SAML Single Logout Protocol can be used with SOAP, HTTP Redirect, HTTP POST, and HTTP Artifact bindings.
- **Assertion Query/Request Profile**: Defines how SAML entities can use the SAML Query and Request Protocol to obtain SAML assertions over a synchronous binding, such as SOAP.
- **Artifact Resolution Profile**: Defines how SAML entities can use the Artifact Resolution Protocol over a synchronous binding, such as SOAP, to obtain the protocol message referred to by an artifact.
- **Name Identifier Management Profile**: Defines how the Name Identifier Management Protocol may be used with SOAP, HTTP Redirect, HTTP POST, and HTTP Artifact bindings.
- **Name Identifier Mapping Profile**: Defines how the Name Identifier Mapping Protocol uses a synchronous binding such as SOAP.

3.2.3.5 Web Browser SSO Profiles

A common usage of SAML is to facilitate single sign-on. SAML provides this capability through the use of an identity provider (IdP) that performs authentication and creates authentication assertions. Clients (browsers) wishing to access protected resources on a destination site, or service provider (SP), must obtain an assertion from the IdP through some proof of identity. The assertion is then presented to the SP, which in turn may verify that assertion with the IdP. Two bindings have been defined specifically for this purpose, each with a slightly different interaction pattern.

HTTP Artifact Binding - In this scenario a browser authenticates to an IdP using some means, such as SSL/TLS, basic authentication, etc., and is given an artifact. The artifact is then presented to a SP, which may then validate the artifact with the IdP. The artifact is generally included in the HTTP URL, (given the alternative method of using cookies does not work across domains). Since URLs have size constraints, the artifact does not include the contents of an assertion, but instead represents the assertion with a 2-byte code. The artifact may be presented to multiple SPs, hence achieving SSO.

Once the SP is presented with the artifact it can request assertions from the IdP. It uses the Artifact Resolution Profile to access the IdP and obtain assertions that are associated with a specific artifact.

HTTP Post Binding - In this scenario a browser makes a request to the IdP passing the target URL that the browser wishes to access. The IdP constructs one or more assertions and puts them into an HTTP form. The IdP redirects the request to the target site along with the HTTP form. Since the form does not have size restrictions similar to URLs, it can contain actual assertions rather than artifacts. To protect assertions, this interaction between sites should use a secure transport, such as SSL/TLS.

3.2.4 XACML

eXtensible Access Control Markup Language (XACML) provides a standard way to represent access control policy information using XML. It also defines an access control architecture that provides abstraction between policy enforcement, decision making, information gathering, and administration points. Since policy enforcement is generally done by application framework components and decision logic can be abstracted out to a security framework, XACML also provides an interface specification between the enforcement and decision points.

3.2.4.1 XACML Rules

XAMCL defines access control policies in terms of rules, which in turn are defined to include a target, an effect, and a set of conditions. A target pertains to actions on resources (SOA Services, components, systems) by subjects (users or other types of clients). An effect is the outcome of a policy decision, e.g., "permit" or "deny". Conditions can be included in the rule to make it dynamic. They involve the evaluation of attributes that may pertain to the subject, resource, action, or environment (e.g., time of day). The evaluation of attributes is referred to as the "predicate" of a rule.

Using this scheme, XACML can accommodate very complex and detailed access control rules, such as: "Permit read access to patient files if user = patient". Using this example the following assignments are made:

- Effect = "permit"
- Target = "read access to patient records by user"
- Condition = "user = patient"

XACML defines an XML schema used to represent rules. Rules pertaining to a common target may be combined into a rule set. The combination of a target and rule set is called a policy statement. Given the ability to define multiple rules on a single target opens up the possibility that rules may conflict with each other. XACML addresses this situation with the inclusion of rule combining algorithms.

It is worth mentioning that policies and rules represented in XACML format are used internally within the security framework, not by custom developed services, infrastructure, or applications that enforce access control decisions. Therefore the value in defining a standard representation lies in the ability to import and export policy administratively, i.e., between security frameworks, as opposed to the run-time usage of policy statements outside of the security framework. However, even though policy statements are not accessible, the request/response interface to the security framework to obtain policy decisions is applicable, and is also defined by XACML.

3.2.4.2 XACML Architecture

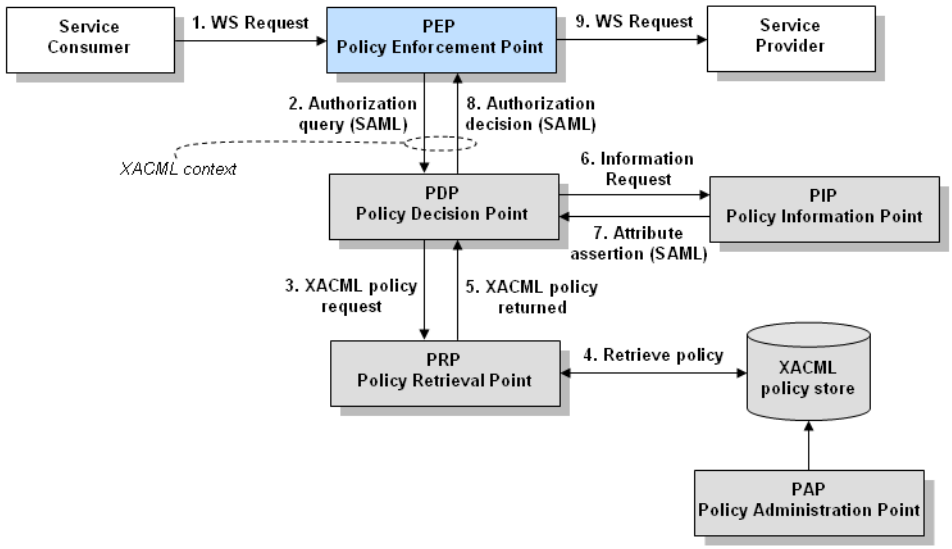
The architecture defined by XACML creates a separation of concerns between various components of the security framework. It identifies components as follows:

- PEP - Policy Enforcement Point, where permit/deny access decisions are enforced. This is generally included in SOA Service or application infrastructure, such as J2EE containers that manage security. It may also be represented as custom code within a SOA Service or application, providing fine grained entitlements evaluation.
- PDP - Policy Decision Point, where policy is evaluated and a decision is made. PDPs may be distributed throughout the IT environment and physically co-located with PEPs to avoid network latency.
- PRP - Policy Retrieval Point, provides access to the policy store. PDPs would ideally cache policy information in order to avoid the performance penalty associated with policy retrieval. PRPs would then only be invoked for policies that have not yet been cached.
- PIP - Policy Information Point, where information can be retrieved to evaluate policy conditions. For example, a user's role or time of day may be needed by the PDP to make a policy decision.

- PAP - Policy Administration Point, where policy can be managed. This may be a central administration point that is used throughout the organization to create holistic policy statements.

An illustration of this architecture is shown in Figure 3–3 below.

Figure 3–3 XACML Security Architecture



Here we see the service provider and service consumer across the top of the diagram, with the security logic in between to control access. Access control (enforcement) may be provided by application framework that the service provider leverages, or by an intermediary such as a service bus. The enforcement point uses the security framework, (below it in the diagram), to perform authorization.

While the base XACML specification defines the content of messages passed between entities in this security model, it does not specify assertions, protocols, and transport mechanisms used to communicate messages. Instead, it provides profiles that allow other standards to be incorporated. One such standard is SAML. The SAML 2.0 profile of XACML 2.0 defines SAML assertions used to carry policies, policy queries and responses, authorization decisions, authorization query decisions and responses, and attribute assertions. In this way SAML authentication, attribute, and authorization assertions are incorporated into the security framework to complement XACML.

More information on XACML and the SAML profile can be found on the OASIS web site.

3.2.5 SPML

In the past several years, the acceptance of SAML has helped organizations broker transactions with partners and third party systems outside of their own security domain. This has allowed businesses to conduct business more fluidly by orchestrating more sophisticated business processes that span business boundaries. It also enables cloud and SaaS computing models where an organization's computing environment is distributed across multiple IT environments.

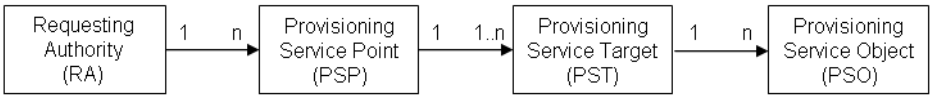
However, a federated environment such as this is dependent upon the notion that users' identities are present in the foreign environments. The usual approach to identity provisioning involves manual "out-of-band" processes. The obvious

drawbacks to this are the time lag in getting information provisioned, and the effort required for manual data transfer and entry.

Service Provisioning Markup Language (SPML) is an XML-based standard that describes provisioning requests, responses, syntax, and constructs needed to automate the provisioning of identity information. This allows systems to communicate and handle identity provisioning using a common messaging protocol, which helps to automate the process of establishing users across security domains in a consistent manner.

SPML defines the notion of a Requesting Authority, Provisioning Service Point, Provisioning Service Target, and Provisioning Service Object. The relationships between these entities is shown in Figure 3–4.

Figure 3–4 SPML Entity Relationship Diagram



A Requesting Authority (RA) initiates a provisioning request to a Provisioning Service Point (PSP). It may be aware of, and issue requests to multiple PSPs. The PSP controls one or more target entities that maintain identity information, aka Provisioning Service Targets (PSTs). The PST can be a physical identity store, such as an LDAP directory or database, or can be a logical or virtual target that accepts commands from the PSP. Each identity stored within the PST is considered a Provisioning Service Object (PSO). A PSP can issue a provisioning request to another PSP, in which case it becomes an RA for that interaction.

SPML is an OASIS standard, and is currently at version 2.0. More information is available at: http://www.oasis-open.org/specs/index.php#spml

3.3 Web Service Security

The topics in this section pertain to security standards that have been created specifically for SOAP. They may leverage XML standards from the previous section by specifying how those standards are incorporated into SOAP messages.

3.3.1 WS-Security

The goal of WS-Security is to enable the exchange of secure SOAP messages. It does so by defining a standard way to incorporate integrity and confidentiality constructs within the SOAP header. It also provides a general purpose mechanism for associating security tokens with message content. The specification is designed to be extensible and versatile, leveraging existing methods of encryption, signature mechanisms, and token formats.

Security information is contained within a <wsse:Security> header block. A header block may be targeted at a specific recipient. There may also be multiple header blocks within the same SOAP message targeted at different recipients. In this way a message may route through intermediaries that add, remove, and react to different security conditions. However, the specification forbids having multiple blocks target the same recipient.

WS-Security leverages the XML Signature specification for message integrity, e.g., elements defined within that specification are used to represent signature information within the <wsse:Security> header. Likewise, it also leverages the XML Encryption

specification for message encryption. It allows encryption of any combination of message body and header blocks, and elements. The common key may be shared by the producer and recipient, or may be included within the <wsse:Security> header in an encrypted form.

WS-Security defines how security tokens can be attached to message headers, and outlines a number of token types supported. Token types include Username Token, (which are basically an XML block containing the user's identity); binary security tokens, such as X.509 certificates and Kerberos tokens, (which must be encoded to conform to an XML format); and other forms of XML-based security tokens, such as SAML. The use of each of these token types is defined in detail in separate WS-Security token profile specifications. The current version of WS-Security (1.1), supports the following token profiles:

- Username Token Profile 1.1
- X.509 Token Profile 1.1
- SAML Token Profile 1.1
- Kerberos Token Profile 1.1
- Rights Expression Language (REL) Token Profile 1.1

More information on WS-Security can be found at:
http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss

3.3.2 WS-Trust

The goal of WS-Trust is to enable applications to construct trusted SOAP message exchanges. WS-Trust builds upon the secure message mechanisms provided by WS-Security in order to establish trust relationships and exchange credentials between security domains.

In an environment of multiple security domains, the credentials required by a service in one domain may not directly be satisfied by a consumer in another domain. If the requestor does not have the necessary tokens(s), it can contact appropriate authorities (as indicated in the service's policy) and request the needed tokens. These "authorities", called Security Token Services (STSs), form the basis of trust by issuing a range of security tokens that can be used to broker trust between domains.

WS-Trust supports the use of existing technologies, such as X.509 certificates, XML-based tokens, Kerberos tickets, and password digests. It specifies message protocols and rules for the issuing, renewing, and validating of security tokens. And, it includes request/response formats and language for interaction between service providers, consumers, and security token services.

Since WS-Trust is intended to bridge gaps between security domains, it is not applicable for use within a single domain. Therefore, companies that align security policy and use of technology can avoid the overhead involved in brokering trust relationships. In a way, the need for a specification of this nature helps underscore the benefit of proper security planning. By avoiding siloed security, one can avoid the need to broker trust between silos. WS-Trust becomes most useful in external facing service exchanges, such as B2B scenarios, where having a common domain is not feasible.

More information on WS-Trust can be found at:
http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-spec-cd-01.pdf

3.3.3 WS-Federation

WS-Federation is a specification that extends WS-Trust to include additional services that are commonly needed in a federated environment. Where WS-Trust defined a Security Token Service (STS) and request/response messages to broker trust in the form of credentials, WS-Federation adds additional services while leveraging the same STS model.

The services added by WS-Federation include:

Authorization Service - Provides access control decision brokering for participants in a federation. It also defines a way to indicate claims that are required in order to grant a request.

Attribute Service - Access control decisions may require attributes of a user that are not included in the initial authentication token. The attribute service provides a way to obtain these attributes in order to make authorization decisions.

Pseudonym Service - A type of attribute service that provides alternative identity information when there is a concern about the risks of identity fraud. It is incorporated into the STS in a way that allows the STS to automatically issue pseudonym tokens in place of an identity token.

In addition to these services, WS-Federation includes semantics to support federated sign-out, privacy requirements for token encryption, and the specification of metadata to establish federation between domains.

Based on this list of services, on the surface it would appear that WS-Federation and SAML 2.0 standards overlap. The main difference is that the similar services that SAML defines are intended for browser-based interactions. WS-Federation services are designed to support both Web Service applications and browser-based scenarios using a common WS-Trust/STS protocol definition. So there is in fact an overlap between the two standards when WS-Federation is used for browser clients and web applications over HTTP.

The complete OASIS WS-Federation standard is posted on online.

3.3.4 WS-SecureConversation

This specification defines extensions that build on WS-Security and WS-Trust to provide secure communication across one or more messages. Specifically, this specification defines mechanisms for establishing and sharing security contexts, and deriving keys from established security contexts (or any shared secret).

While WS-Security focuses on the message authentication model, e.g., authentication of individual messages, WS-SecureConversation introduces the concept of a security context. It involves the establishment of a context via a series of messages. This message exchange initially adds additional overhead, but avoids the need to authenticate each subsequent Web Service request. It effectively acts as SSL/TLS at the SOAP layer.

The security context is defined as a new WS-Security token type that is obtained using a binding of WS-Trust. Building on concepts introduces in WS-Trust, WS-SecureConversation uses a security token service. It defines a request and response protocol to acquire tokens used to protect messages in a conversation.

More information on WS-SecureConversation can be found here at OASIS.

3.3.5 WS-Policy and WS-SecurityPolicy

The goal of WS-Policy is to provide the mechanisms needed to enable Web Services to specify policy information. It provides a flexible and extensible XML grammar for expressing the capabilities, requirements, and general characteristics of Web Services.

WS-Policy defines a policy to be a collection of policy alternatives, where each policy alternative is a collection of policy assertions. Assertions may pertain to functional capabilities, such as security or protocol requirements, while others may be non-functional, such as QoS characteristics.

WS-Policy does not specify how policies are discovered or attached to a Web Service, nor does it define specific policy characteristics. Instead, it is more of an abstract specification, designed to enable different types of policies and different types of attachment capabilities. It relies on other specifications, such as WS-PolicyAttachment, to describe discovery and attachment scenarios, and WS-SecurityPolicy - one example of a specific policy definition specification.

More information on WS-Policy can be found at:
http://www.w3.org/Submission/WS-Policy/

3.3.5.1 WS-PolicyAttachment

WS-PolicyAttachment defines two general-purpose mechanisms for associating policies with the subjects to which they apply. They may be defined as part of existing metadata about the subject (e.g., attached to the service definition WSDL), or defined independently and associated through an external binding (e.g., referenced to a UDDI entry). As such, the specification describes the use of policies with WSDL 1.1, UDDI 2.0, and UDDI 3.0. It defines:

- How to reference policies from WSDL definitions
- How to associate policies with deployed Web Service endpoints
- How to associate policies with UDDI entities

Policies may be attached to different parts of a WSDL depending on the applicability of the policy. For example, it may be attached to the service itself, the endpoint, the operation, or the message. The attachment point determines the scope of the policy. Likewise, policy may be attached to a UDDI entity in a similar manner.

Policies may also be registered as a distinct tModels in a UDDI registry. This enables common policies to be used by multiple services.

More information on WS-PolicyAttachment can be found at:
http://www.w3.org/Submission/WS-PolicyAttachment/

3.3.5.2 WS-SecurityPolicy

WS-SecurityPolicy defines a set of security policy assertions for use with the WS-Policy framework with respect to security features provided in WS-Security, WS-Trust, and WS-SecureConversation. It defines a base set of assertions that describe how messages are to be secured. It is meant to be flexible with respect to token types, algorithms, and mechanisms used, in order to allow for evolution over time.

Assertion types include these categories of assertions, plus many others:

- **Protection assertions**, such as the integrity assertion, confidentiality assertion, and required elements assertion, which describe how messages should be protected.
- **Token assertions**, which specify the types of tokens to use to protect or bind tokens and claims to the message. Token types include Username Token, IssuedToken, X509Token, KerberosToken, SpnegoContextToken,

SecurityContextToken, SecureConversationToken, SamlToken, RelToken, and HttpsToken assertions.

- **Security binding assertions**, such as AlgorithmSuite, Layout, TransportBinding, SymmetricBinding, and AsymmetricBinding assertions.

WS-Policy also defines assertions to enable a Web Service to describe which WS-Security and/or WS-Trust options are supported.

More information on WS-SecurityPolicy is available here at OASIS.

3.4 WS-I Standards Interoperability

The Web Services Interoperability Organization (WS-I) is an open industry organization chartered to promote Web Services interoperability across platforms, operating systems and programming languages. It acts as a standards integrator to help Web Services advance in a structured, coherent manner. Its goals are to:

- Achieve Web Services interoperability by integrating existing specifications in a consistent manner, and by providing reference implementations.
- Accelerate Web Services deployment by offering implementation guidance, best practices, tools, sample applications, and a developer's forum.
- Encourage Web Services adoption by building industry consensus; thereby reducing adoption risks.

WS-I defines categories of interoperability through the use of profiles. Profiles are a set of specifications or standards at a specific version level. They specify constraints, guidelines and conventions for using these specifications together in ways that promote interoperability. Regarding security, the Basic Security Profile Working Group focuses on SOAP message security, transport, and other security considerations. The following section describe two security profiles from WS-I.

3.4.1 Basic Security Profile

The Basic Security Profile is designed to support the addition of security functionality to SOAP messaging. It identifies requirements, recommendations, and extensibility points to address security concerns in an interoperable manner. Basic Security Profile focuses on usage of the following group of standards:

- WS-Security: SOAP Message Security 1.1 (and associated token profiles)
- SOAP with Attachments (SwA)
- SAML
- XML Encryption
- XML Signatures
- Basic Profile Versions 1.0 and 1.1
- SSL/TLS

Basic Security Profile addresses the most common needs of Web Service security, including message integrity, confidentiality, identity, and proof of authentication. Being compliant with this profile greatly increases the likelihood of compatibility between producers and consumers when the above mentioned standards are being used.

More information on Basic Security Profile version 1.0 can be found at:
http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0.html

3.4.2 Reliable Secure Profile

The Reliable Secure Profile (RSP) 1.0 is meant to provide interoperability guidance for the following specifications:

- WS-SecureConversation 1.3
- WS-ReliableMessaging 1.1

When practical, RSP will compose with (compatibly extend) the following specifications:

- WS-I Basic Profile 1.2
- WS-I Basic Profile 2.0
- WS-I Basic Security Profile 1.0
- WS-I Basic Security Profile 1.1

This specification improves interoperability between nodes that need to establish reliable and secure conversations, establishing SOAP level encryption spanning entire conversations.

3.5 Identity Governance Framework

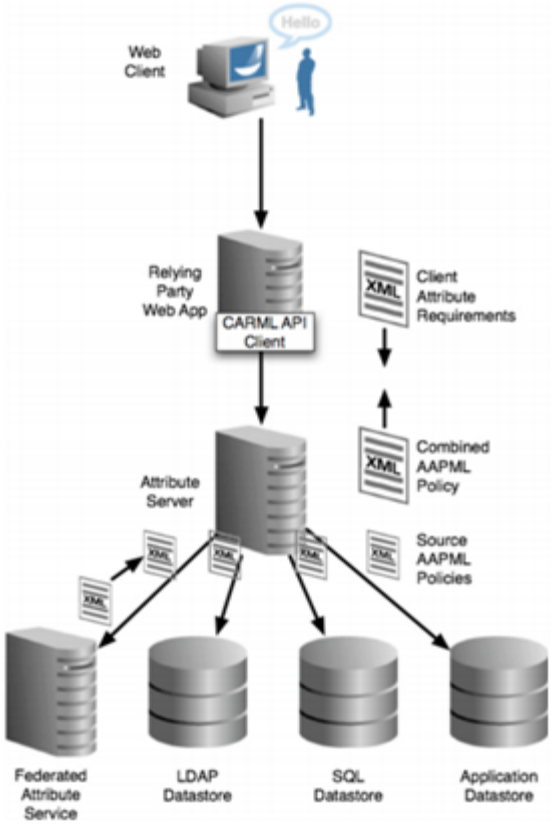
The Identity Governance Framework (IGF) is a definition of standards and constructs designed to provide and govern access to personally identifiable information (PII) in a standardized way. PII is (roughly) defined as information that can be used to uniquely identify or locate an individual, e.g. name, address, phone number, SSN, birth date, etc. This information may factor into authorization decisions, or it may be required in order to perform ordinary transaction processing. Regardless of use, the basic premise is that PII needs to be processed in accordance with policies set forth by governments through regulation, by individuals themselves, and also by organizations holding the data.

IGF is being developed under the auspices of the Liberty Alliance. Specifications and APIs have been produced and many are now implemented by Oracle and others (such as the Open Source community).

According to the Liberty Alliance Project, IGF consists of:

- An identity attribute service that supports access to many different sources of PII and enforces administrative policy.
- Client Attribute Requirements Markup Language (CARML): declarative syntax used by clients to specify their requirements for PII.
- Attribute Authority Policy Markup Language (AAPML): declarative syntax which enables providers of identity-related data to express policy on the usage of information.
- A multi-language API (Java, .NET, Perl) for reading and writing identity-related attributes.

Figure 3–5 IGF Architecture



IGF proposes an architecture where PII conditions of use are formally defined in the form of policies. Policies are expressed using AAPML (XML-based) syntax. Given that PII may often reside in multiple places, aggregation and mapping of data and policies are supported.

Access to PII is provided through an Identity Attribute Service; (Attribute Server above). This service acts as a single point of access, and where policies can be evaluated and access can be audited. Policy evaluation is handled at design-time, i.e. clients must provide a detailed request defining the type of information needed, for what purpose, by whom, what operations will be performed, etc. A security analyst will then evaluate the request against AAPML-based policies and create a view, or mapping, of the client request to existing PII (attributes).

Client requests are in the form of XML documents following the CARML specification. The CARML document acts as a client interface, much like WSDL for Web Services. Therefore, once a CARML document is approved, the developers can begin coding directly to that interface. They do not need to know where the attributes originate from or any such technical details.

3.5.1 AAPML

AAPML is a XACML profile designed to allow attribute authorities to specify conditions under which information under management may be used (and possibly modified) by other applications[1]. It enables each identity attribute, (PII data element)

[1] Attribute Authority Policy Markup Language, Working Draft 08, November 28 2006

to have an associated privacy policy. Using an XML-based policy language, it is possible to establish specific definitions, such as:

- purpose of storage/existence
- propagation limits, e.g. not beyond existing holding organization
- retention rules, including whether the attribute can be persisted, cached, whether it must be encrypted before persisting, and whether it can be logged or not
- permitted duration of retention
- what to do in the event of data loss/breach
- contract or regulation governing the data
- how the data must be masked when logged or displayed on the screen

3.5.2 CARML

CARML provides a means for a client to interact with an Identity Service while respecting privacy constraints. The interactions supported include: reading, changing, comparing, and deleting attributes and searching for subjects with certain attributes. It is possible to specify the attributes, roles, predicates (conditions) and policies relevant to the interaction. CARML is in effect an XML based API substitute for LDAP, SQL, and other mechanisms previously used to access identity attribute data.

See http://www.openliberty.org/wiki/index.php/ProjectAris for more detailed information and http://www.projectliberty.org/resource_center/specifications/igf_v1_0_final/ for the IGF specifications themselves.

3.6 Java™ Security Standards

The Java platform strongly emphasizes security, including language safety, cryptography, public key infrastructure, authentication, secure communication, and access control. This section introduces security standards for the Java programming language. These standards have been introduced into the Java SE and/or Java EE platforms over the years. Some may have originated as add-on libraries, but all are now part of the latest platform version.

This section is intended as an introduction to Java security standards. It is derived from an extensive collection of information on the subject that is available at: http://java.sun.com/javase/technologies/security/ and http://java.sun.com/javaee/technologies/management/

3.6.1 JCA and JCE

The Java Cryptography Architecture (JCA) contains a "provider" architecture and a set of APIs for digital signatures, message digests (hashes), certificates and certificate validation, encryption (symmetric/asymmetric block/stream ciphers), key generation and management, and secure random number generation, etc. These APIs allow developers to integrate security into their application code in a consistent and manageable way. The architecture was designed to provide:

- **Implementation independence.** Applications do not need to implement security algorithms. Rather, they can request security services from the Java platform. Security services are implemented in providers which are plugged into the Java platform via a standard interface. An application may rely on multiple independent providers for security functionality.

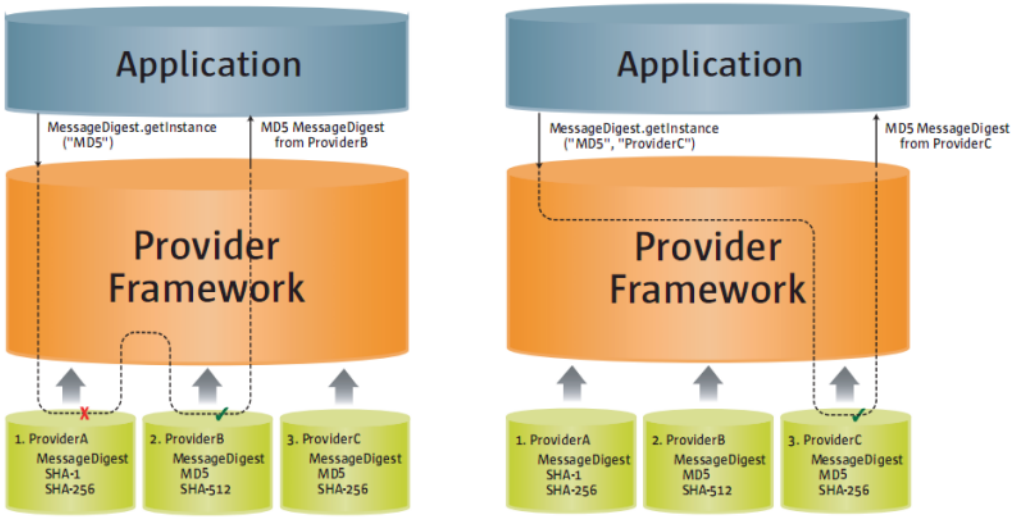
- **Implementation interoperability.** Providers are interoperable across applications. E.g. an application is not bound to a specific provider, and a provider is not bound to a specific application.
- **Algorithm extensibility**. The Java platform includes a number of providers that implement common algorithms and supports the installation of custom providers to implement custom and future algorithms.

> **Note:** Prior to JDK 1.4, the Java Cryptography Extension (JCE) was an unbundled product, and as such, the JCA and JCE were regularly referred to as separate, distinct components. Since the JCE is now bundled into the JDK, and the JCA and JCE share the same architecture, the JCE should now be thought of as part of the JCA.

3.6.1.1 Cryptographic Service Provider Architecture

Figure 3–6 illustrates the concept of JCA providers. Each provider contains an instance of the `java.security.Provider` class which contains the provider's name and lists all of the security services/algorithms it implements. When an instance of a particular algorithm is needed, the JCA framework consults the provider's database, and if a suitable match is found, the instance is created.

Figure 3–6 JCA Provider Architecture



The example on the left illustrates a request for message digest algorithm MD5. The JCA framework searches all providers until one is found that is registered under that name. On the right, a request was made for MD5 explicitly calling out Provider C.

The provider architecture makes it easy for end-users to add additional providers. In addition to the providers included in the JDK, many third party provider implementations are already available.

3.6.2 JAAS

The Java Authentication and Authorization Service (JAAS) was introduced as an optional package (extension) to the Java 2 SDK, Standard Edition (J2SDK), v1.3. JAAS was integrated into the J2SDK 1.4. JAAS provides two main capabilities:

authentication of users, and authorization to ensure users have permission to perform requested actions.

JAAS authentication is performed in a pluggable fashion. This permits applications to remain independent from underlying authentication technologies. New or updated authentication technologies can be plugged under an application without requiring modifications to the application itself. Applications enable the authentication process by instantiating a `LoginContext` object, which in turn references a `Configuration` to determine the authentication technology, or `LoginModule`(s), to be used in performing the authentication. Typical `LoginModules` may prompt for and verify a username and password. Others may use other forms of credentials.

Upon successful authentication a `Subject` object is created, which represents the authenticated party. At this point the JAAS authorization component works in conjunction with the core Java SE access control model to protect access to sensitive resources. Access control decisions are based on `Principles` assigned to the `Subject`, and the application code the `Subject` is attempting to execute.

3.6.3 JSSE, JGSS, and Java SASL

The Java Secure Socket Extension (JSSE) enables secure Internet communications. It provides a framework and an implementation for a Java version of the SSL and TLS protocols and includes functionality for data encryption, server authentication, message integrity, and optional client authentication. Using JSSE, developers can provide for the secure passage of data between a client and a server running any application protocol, such as Hypertext Transfer Protocol (HTTP), Telnet, or FTP, over TCP/IP.

JSSE provides support for several cryptographic algorithms commonly used in cipher suites, including RSA, RC4, DES, 3DES, AES, Diffie-Hellman, and DSA. It uses an extensible "provider" architecture pattern that supports implementation and algorithm independence, as previously described for JCA.

JSSE was previously an optional package to the Java 2 SDK, Standard Edition (J2SDK), v1.3. JSSE was integrated into the Java Standard Edition Development Kit starting with J2SDK 1.4.

The Java Generic Security Services (JGSS) API is another API that can be used for secure communications in a Java environment. It is similar to JSSE, but is a token-based API that is required to support token-based protocols such as Kerberos. It allows the developer to selectively encrypt certain messages, as opposed to all or none.

Simple Authentication and Security Layer, (SASL), is an Internet standard (RFC 2222) that specifies a protocol for authentication and optional establishment of a security layer between client and server applications. It is used by protocols, such as LDAP and the Internet Message Access Protocol (IMAP) to enable pluggable authentication.

The Java SASL API defines classes and interfaces for applications that use SASL mechanisms. It allows applications to select the mechanism to use based on desired security features. SASL mechanisms also follow the JCA provider architecture. Therefore, the Java SASL API also allows developers to use their own, custom SASL mechanisms.

3.7 Regulatory & Governance Standards

This section includes some common regulatory and management standards that pertain to security.

3.7.1 Information Technology Infrastructure Library

The **Information Technology Infrastructure Library (ITIL)** is a set of concepts, best practices, processes, and policies around IT Service Management. Enterprises have recognized that IT Services are crucial, strategic, organizational assets and therefore enterprises must invest appropriate levels of resource into the support, delivery, and management of these critical IT Services and the IT systems that underpin them.

ITIL consists of a series of books giving guidance at each stage of the IT Service lifecycle, from the initial definition and analysis of business requirements in Service Strategy and Service Design, through migration into the live environment within Service Transition, to live operation and improvement in Service Operation and Continual Service Improvement. ITIL includes a book on Security Management designed to promote adequate information security.

More information on ITIL can be found at: http://www.itil-officialsite.com

3.7.2 Control Objectives for Information and Related Technology

Control Objectives for Information and related Technology (COBIT) is an IT governance framework and supporting toolset that allows managers to bridge the gap between control requirements, technical issues, and business risks. COBIT enables clear policy development and good practice for IT control throughout organizations. COBIT emphasizes regulatory compliance, helps organizations increase the value attained from IT, enables alignment, and simplifies implementation of the COBIT framework. COBIT processes include security issues and training.

More information on COBIT can be found at: http://www.isaca.org/

3.7.3 Committee of Sponsoring Organizations

The Committee of Sponsoring Organizations of the Treadway Commission (COSO) is dedicated to providing guidance on critical aspects of organizational governance, business ethics, internal control, enterprise risk management, fraud, and financial reporting. It is a voluntary private-sector organization established to address fraudulent corporate financial reporting in response to the 1997 U.S. Foreign Corrupt Practices Act (FCPA).

COSO defines a framework for describing and analyzing the internal control system implemented in an organization. The framework now consists of eight components which include internal environment, objective setting, event identification, risk assessment, risk response, control activities, information & communication, and monitoring. IT security plays a role in several components including those related to risk management, information security, and monitoring.

Information about COS can be found at http://www.coso.org/.

3.7.4 International Organization for Standards and the International Electrotechnical Commission

The International Organization for Standards (ISO) and the International Electrotechnical Commission (IEC) have jointly published a series of standards pertaining to Information Security Management Systems (ISMS). The ISO/IEC 27000-series (ISO 27k), aka ISMS Family of Standards, provide best practice recommendations on information security management, risks, and controls. There are several standards already published and many more currently under development. ISO/IEC 27001 is perhaps the most well known standard in the series.

ISO/IEC 27001, published in 2005, specifies the requirements for establishing, implementing, operating, monitoring, reviewing, maintaining and improving a documented Information Security Management System within the context of the organization's overall business risks. It specifies requirements for the implementation of security controls customized to the needs of individual organizations or parts thereof[2]. Organizations that claim to have adopted ISO/IEC 27001 can be formally audited and certified compliant with the standard.

The full listing of ISO/IEC 2700-series standards can be found on the ISO web site.

3.7.5 Sarbanes-Oxley

Sarbanes-Oxley (SOX) is a United States federal law as a reaction to a number of major corporate and accounting scandals. The legislation set new or enhanced standards for all U.S. public company boards, management, and public accounting firms.

Sarbanes-Oxley contains 11 titles that describe specific mandates and requirements for financial reporting.

The text of the law can be found at:
http://frwebgate.access.gpo.gov/cgibin/getdoc.cgi?dbname=107_cong_bills&docid=f:h3763enr.tst.pdf

3.7.6 Payment Card Industry Data Security Standards

The **Payment Card Industry Data Security Standards (PCI DSS)** is a set of security requirements around management, policies, procedures, network architecture, software design, and other critical protective measures.

Table 3–1 PCI DSS Requirements

Control Objectives	PCI DSS Requirements
Build and Maintain a Secure Network	■ Install and maintain a firewall configuration to protect cardholder data ■ Do not use vendor-supplied defaults for system passwords and other security parameters
Protect Cardholder Data	■ Protect stored cardholder data ■ Encrypt transmission of cardholder data across open, public networks
Maintain a Vulnerability Management Program	■ Use and regularly update anti-virus software on all systems commonly affected by malware ■ Develop and maintain secure systems and applications
Implement Strong Access Control Measures	■ Restrict access to cardholder data by business need-to-know ■ Assign a unique ID to each person with computer access ■ Restrict physical access to cardholder data
Regularly Monitor and Test Networks	■ Track and monitor all access to network resources and cardholder data ■ Regularly test security systems and processes
Maintain an Information Security Policy	■ Maintain a policy that addresses information security

[2] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=42103

The standard assists enterprises that process card payments to prevent credit card fraud through increased controls around data and its exposure to compromise. The standard applies to all organizations which hold, process, or pass cardholder information from any card branded with the logo of one of the card brands.

Enterprises require a security framework that not only assists in implementing these requirements but also provides the ability to audit and attest to compliance.

More information on PCI DSS can be found at:
https://www.pcisecuritystandards.org/security_standards/pci_dss.shtml

4

Conceptual View

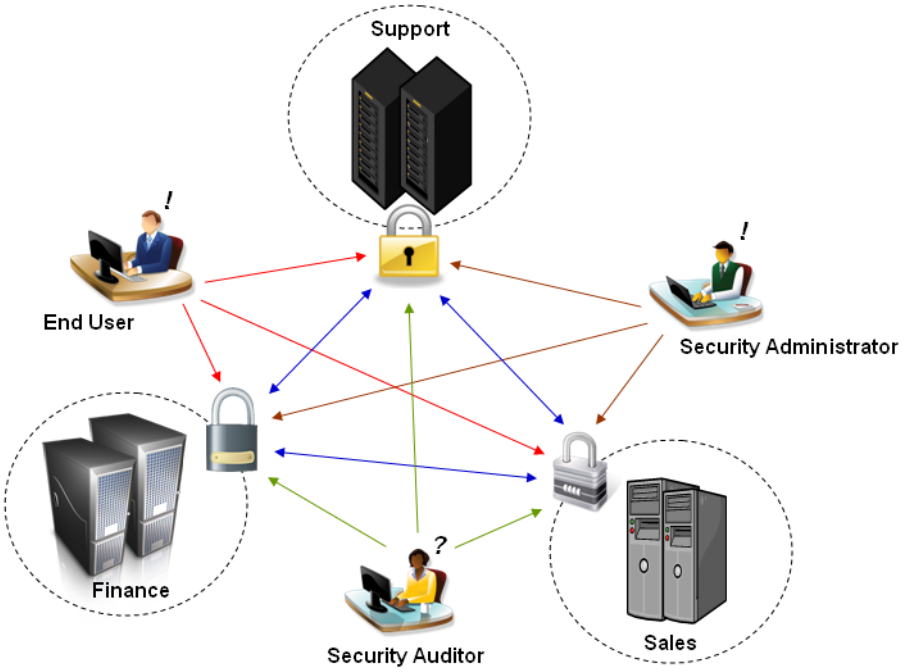
This chapter of the reference architecture describes the need for a common security framework and conceptually introduces such a framework. It builds on the capabilities and standards described in the previous chapters and provides context for the next chapter which presents a logical view.

4.1 Need for a Common Security Framework

The second chapter of the reference architecture described a number of concepts and capabilities that a secure computing environment must provide. Many of these concepts have been around for a relatively long time, and have been addressed over the years in a number of ways. Therefore, providing these capabilities is not new, and is not necessarily difficult. The real challenge is providing them in a way that supports business agility, improves IT responsiveness, and enables an organization to know what security measures are in place.

For example, Figure 4–1 illustrates the problems one can encounter when security is maintained in application silos. In this example there are three systems, each with its own security implementation and management capabilities. End users must log into each system and remember passwords for each system. The password policies may be different across systems including the requirements for character usage, change intervals, and uniqueness. Administrators need to configure each system separately with user credentials, attributes, access privileges, roles, policies, etc. - a task that can be very tedious and error-prone. Likewise, auditors need to access security configurations on each system and aggregate data across systems in order to obtain an understanding of security across IT.

Figure 4–1 Security Silos



Security silos also create problems for application integration, which is of great importance to modern computing solutions that leverage business processes, SOA Services, and enterprise portals. In these situations security is often considered a hindrance to productivity and agility. Since bridging security incompatibilities is often quite time consuming and technically challenging, the decision to disable security for system interactions is routinely made. This can severely compromise security, particularly where threats from inside occur, e.g. employees and contractors that have the ability and means to steal or misuse unprotected resources.

The unfortunate reality is that many organizations have to deal with multiple security silos. The problem is a natural extension of application silos, where each application was built as a self-contained, standalone entity. Many legacy systems and packaged applications were initially designed in this manner. Fortunately the industry is trending toward greater openness and embracing interoperability between applications and common security infrastructure.

4.1.1 Purpose of a Security Framework

In order to provide security in a consistent manner, a common set of infrastructure, e.g. a security framework, must be used. The purpose of this framework is to rationalize security across the enterprise by:

- Establishing a master set of security data that reflect the policies, IT resources, participants and their attributes across the entire domain of security
- Mapping organizational structures, computing resources, and users to roles in a way that clearly depicts access privileges for the organization
- Maintaining fine-grained access rules based on roles that have been established for the organization
- Propagating the master security data to individual applications and systems that enforce security

- Detecting changes to security data residing on systems that have not been propagated from the master source of record, and sending alerts regarding these inconsistencies
- Providing common security services, such as authentication, authorization, credential mapping, auditing, etc. that solutions can leverage going forward in place of custom-developed and proprietary functions
- Facilitating interoperability between systems and trust between security domains by acting as a trusted authority and brokering credentials as needed
- Centrally managing security policies for SOA Service interactions

The security framework should provide these types of capabilities as a value-add to the existing infrastructure. The intent is not to discard the capabilities built into current applications, but rather to provide a common foundation that enhances security across the enterprise. Security enforcement can still be performed locally, but security data should be modeled and managed holistically.

Figure 4–2 Security Unified via a Common Framework

A security framework can greatly rationalize the way security functions are performed by offering a single point of management, a unified view of security data and policies, a common authentication scheme (SSO), and a means to promote system interoperability in a secure and consistent manner.

As depicted in Figure 4–2, the security framework greatly simplifies usage, administration, auditing, and integration. It does not eliminate the security capabilities provided by the individual computing platforms, but rather works in conjunction with them.

4.2 Architecture Principles

The following sections contain useful architecture principles that pertain to security.

4.2.1 Defense in Depth

Principle	**Defense in Depth**
Statement	Failure of a single component of the security architecture must not compromise the entire IT environment.
Rationale	The environment must be protected by multiple independent and reinforcing controls such that a single failure will have minimal impact.
Implications	■ Multiple security perimeters must exist between public networks and protected resources. ■ Access between perimeters must be restricted such that only traffic from known systems, ports, and protocols can pass. ■ Solutions must be designed such that access to protected resources requires an attacker to breach more than one application or system. ■ Each computing and database platform must be able to inject security controls into the processing chain.

4.2.2 Least Privilege

Principle	**Least Privilege**
Statement	Users and other consumers of resources must operate using the least set of privileges necessary to complete the job.
Rationale	Security risks increase with the amount of access a user or resource consumer is granted. Risks can stem from misuse of privilege, unintentional destructive actions, compromised accounts and systems, etc.
Implications	■ By default, users/consumers should be denied access so that being able to access data or functionality must be a conscious decision. ■ Users should be granted access to functions and data based on their roles and/or attributes, i.e., based on needs rather than enabled by default. ■ A review of a user's access rights should be performed when the user's attributes change, such as when the user changes roles. Similarly, access rights must be revoked when a user leaves to organization. ■ System access to data stores, other systems, and networks should be granted only to the extent required for proper operation.

4.2.3 Security as a Service

Principle	**Security as a Service**
Statement	Business solutions must be designed to consume common security services where possible as opposed to implementing custom security logic and replicating copies of security data.

Rationale	Security services allow multiple solutions to share common security logic, features, policies, and identity information. This provides a more secure environment by eliminating redundancies and associated risks. It also enables more effective management of security in the IT environment.
Implications	■ Security logic must be externalized as much as possible, i.e., developers must not hand-code security logic into business solutions. ■ Security enforcement, decisions, and management must be performed by dedicated, shared services and infrastructure. ■ Security services must leverage open standards for interface protocols and message formats where possible in order to promote interoperability. ■ The availability and performance characteristics of security services must meet or exceed the specifications required to support the business solutions.

Principle	**Authentication as a Service**
Statement	Business solutions must leverage a shared authentication service where feasible.
Rationale	A service-based approach improves consistency, auditability, and maintainability, and therefore offers greater security than multiple disparate authentication mechanisms.
Implications	All implications listed for "Security as a Service", plus: ■ The service must support a variety of authentication methods, such as passwords, certificates, tokens, etc., and be extensible to support new and custom-developed methods. ■ The service must provide the ability to associate protected resources with certain authentication methods in order to control how each resource is protected. ■ The service must be capable of providing identity validation via multiple identity stores for security domains that have more than one store for users and credentials. ■ The service must support strong and/or multi-factor authentication as well as methods that thwart keyloggers and man-in-the-middle attacks. The service must be configurable and adaptable to meet the security needs of the organization. ■ The service must provide a cookie/artifact/token that enables single sign-on to other solutions that are configured for the same form of authentication. ■ The service must produce an audit log of authentication events that occur, (successful or not), and changes made to the service.

Principle	**Authorization as a Service**
Statement	Business solutions must leverage a shared authorization service where feasible.
Rationale	A service-based approach improves consistency, auditability, and maintainability, and therefore offers greater security than multiple disparate authorization mechanisms.

Implications	All implications listed for "Security as a Service", plus: ■ Security infrastructure must provide the ability to perform policy decisions and enforcement locally, and policy administration centrally. ■ The service must provide the ability to define access policies (rules) that control operations on resources based on user factors such as identity, group membership, role assignments, attributes, etc. ■ The service should support both static and dynamic group memberships and role assignments. ■ The service must be extensible to support new and custom-developed authorization mechanisms. ■ The service must produce an audit log of authorization events that occur and changes made to the service.

4.2.4 Federation

Principle	**SSO and Identity Federation Across Domains**
Statement	Security infrastructure must provide identity mapping, credential mapping, and identity propagation necessary to support federation across security domains.
Rationale	Application integration is not limited to security and web cookie domains. Infrastructure should be available to support integration between companies, business units, and divisions that are not part of the same domain. An infrastructure-based approach is preferred over having each solution attempt to solve the problem in its own way.
Implications	■ The solution must be based on industry standards such as WS-Trust, WS-Federation, and SAML. ■ Authentication associated with identity federation must be based on common authentication services. ■ A trust structure must be defined and adhered to. The structure indicates which systems trust each other based on strength of protocols, algorithms, procedures used, etc. It clearly identifies trust relationships, and lack thereof; requirements to maintain the relationship, and means for untrusted systems to interact.

4.2.5 Web Service Security

Principle	**Secure Web Services**
Statement	End-to-end system security and adherence to other security principles must not be compromised by the use of Web Services.
Rationale	Adopting SOA and Web Services must not negatively impact system security, negate the use of infrastructure-based security services, or deviate from the use of industry standards over proprietary interfaces.

Implications	■ Security infrastructure must protect Web Services end-to-end regardless of the number of intermediaries that exist between the consumer and provider. ■ Security infrastructure that provides Web Service Security (WSS) must leverage open security and interoperability standards, such as those based on WS-Security and WS-I Basic Security Profile. ■ WSS must be fulfilled and enforced using security infrastructure, e.g. common security services that provide authentication, federation, and authorization. It should not be hard-coded into service consumer or service provider business logic. ■ WSS must be declared using security policies that are (or can be) represented using WS-SecurityPolicy documents. ■ Security infrastructure must provide a means to discover security policies for Web Services. ■ Security infrastructure must provide the means to change security policies at run-time without requiring a service outage or code revision.

4.2.6 Secure Management of Security Information

Principle	**Secure Management of Security Information**
Statement	Security information such as users, credentials, groups, roles, attributes, must be managed centrally (holistically) across the entire organization in a secure and auditable manner.
Rationale	Government regulations require organizations to attest to information access rights in various forms, e.g. access to PII, access to financial information. Security information must be managed in a way that not only makes attestation and auditing possible, but reasonable to perform on a routine basis. Ideally, users and privileges should be provisioned and managed using a common, secure, and complete set of infrastructure and processes. This reduces the likelihood of orphaned accounts, outdated configurations, and forgotten privileges. Even if multiple identity and attribute stores are deployed, the system to manage identity should be connected and able to cope with such implementation details.

Implications	■ There must be a set of well-defined and managed processes to control the provisioning of security information such as users, credentials, attributes, groups, role assignments, and access privileges. This includes the creation, update, and deletion of such information. The process must include steps for approvals as deemed necessary. ■ Security infrastructure must support the dissemination and synchronization of security data to the various persistent stores, whether they are part of the common security framework or embedded within individual applications and systems. ■ Security infrastructure must detect and send alerts when security data in any individual persistent store has been changed using an out-of-band method and no longer matches the master version established via provisioning processes. ■ Security infrastructure must provide the means to accurately report on access rights for users, groups, and roles, and access privileges assigned to all managed resources. ■ Security information must be kept confidential and protected from manipulation at all times. It must be encrypted while in transit and while persisted. ■ An audit log must be kept of all additions and changes made to security information. ■ Security infrastructure must allow for the delegation of administration duties pertaining to security information. ■ Security infrastructure must provide the ability to statically and dynamically map organizational structures to role hierarchies. ■ Security infrastructure must be able to map passwords across various systems and enforce password creation rules. ■ Where multiple credential stores exist, security infrastructure must provide the ability to synchronize like data across stores despite technology differences. It must also offer an aggregate view that represents the set of all unique attributes across stores. ■ Security infrastructure must provide the ability to manage certificates used for public key (PKI) encryption, signatures, and non-repudiation.

4.2.7 Active Threat Detection and Analysis

Principle	**Active Threat Detection and Analysis**
Statement	Security infrastructure must be capable of detecting abnormal behavior and adapting accordingly to protect vulnerable resources.
Rationale	Despite best efforts, attackers can sometimes manage to gain access to protected resources. Security infrastructure can limit damages by detecting abnormal behavior and/or denying access to suspicious users, devices, and locations.

Implications	■ Security infrastructure must monitor for fraudulent use and abnormal behavior and take appropriate measures, e.g. send alerts and suspend accounts. ■ Security infrastructure must provide the means to assess threats based on a number of user-defined factors and model security based on the assessed risk. ■ Security infrastructure must provide the ability to record and block activity from suspicious IP addresses, locations, machines, etc.

4.2.8 Auditing

Principle	**Secure, Complete Audit Trail**
Statement	The security system must be able to identify when changes have been made to data within the organization, what changes have been made, and by whom.
Rationale	Legal and regulatory concerns require organizations to maintain a complete and secure audit trail. Infrastructure must be able to gather audit records from various sources into a secure repository where they can be collectively monitored and reviewed.
Implications	■ Security infrastructure must provide a means to collect audit records from applications, processes, SOA Services and infrastructure into a common store for correlation and analysis. ■ Infrastructure must protect the confidentiality, integrity, and availability of audit records, even from privileged users such as DBAs, architects, and developers. ■ Security infrastructure must provide a means to query audit records, generate reports, and analyze events by type, person, time, etc. ■ Audit records must adhere to a consistent record template that includes elements such as: identification of resource issuing the audit record, the identity of the user making the request, type of request, result (status), and time of day.

4.2.9 Data Security

Principle	**Data Security**
Statement	The confidentiality, integrity, and availability of data must be ensured at all times.
Rationale	Data are a valuable commodity and there are tremendous risks to the company in terms of lost revenue and trust when security is compromised.

Implications	■ The organization must comply with PCI DSS control objectives for cardholder and other PII data. Control objectives are listed in Section 3.7.6, "Payment Card Industry Data Security Standards" ■ Data must be classified according to its sensitivity level and subsequently protected from disclosure, alteration, and destruction. ■ Rules for transporting, persisting, and maintaining each class of data must be defined and adhered to ■ The database must be able to control access to information based on user and data security classifications. ■ Sensitive data and PII must be protected from all users, including privileged database users. ■ Confidential data must be persisted in encrypted form, both within the database and on backup media. ■ Changes to data, database structures, configurations, and security settings must be logged in a secure audit repository.

4.2.10 System Availability

Principle	**System Availability**
Statement	Solutions, systems, applications, services, etc., must be adequately protected to ensure their intended degree of availability, but not overly constrained by security measures to unnecessarily impede normal operations.
Rationale	It is possible to very easily and completely secure a resource by preventing all to access it. Too much security (or bad security) has the effect of preventing legitimate access to systems and data. Security measures, and the overhead of using the system, should not outweigh their usefulness.
Implications	■ Risks must be assessed in order to properly determine the security measures necessary for a system. ■ The costs (money, time, and inconvenience) of securing a system must be weighed against the likelihood and impact of a security breach.

4.3 Security Framework & Services

To define a framework that meets the security architecture principles one might consider the environment to be comprised of basically two parts. One part consists of the platforms that comprise business solutions, such as applications, business processes, SOA Services, databases, portals, etc. These all provide security services to some extent. Some platforms are relatively closed, self-contained, and isolated in terms of a security model, while others are more open to leverage external security services and security data stores.

The second part of the environment consists of the shared infrastructure that is dedicated to providing security services, managing security data, and orchestrating activities related to security provisioning, synchronization, and auditing.

The cardinality between these two parts is one-to-many, e.g. the shared infrastructure that comprises a framework supports many platforms and platform types.

Figure 4–3 Layers of a Security Framework

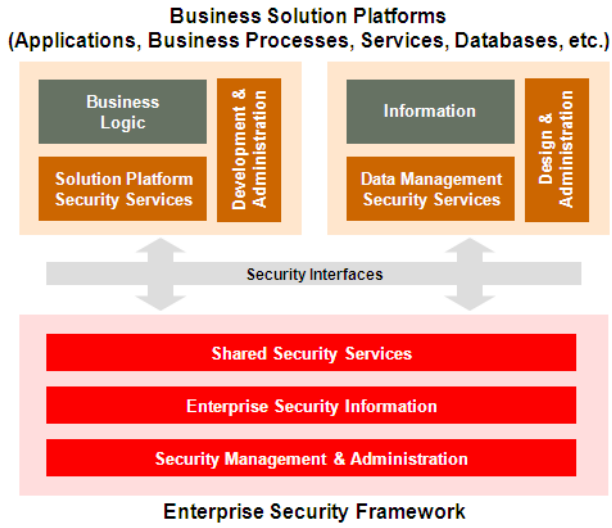


Figure 4–3 illustrates one type of solution platform consisting of business logic, written by software developers and security services that are provided by the platform. It provides support for the inclusion of security at development time as well as the configuration and administration of security for the solution at deployment and run-time. This platform can represent an application server domain or equivalent.

Likewise, another platform consists of information, modeled and designed by information architects. It also offers security services that can be configured and administered locally. This platform can represent a database or content management system.

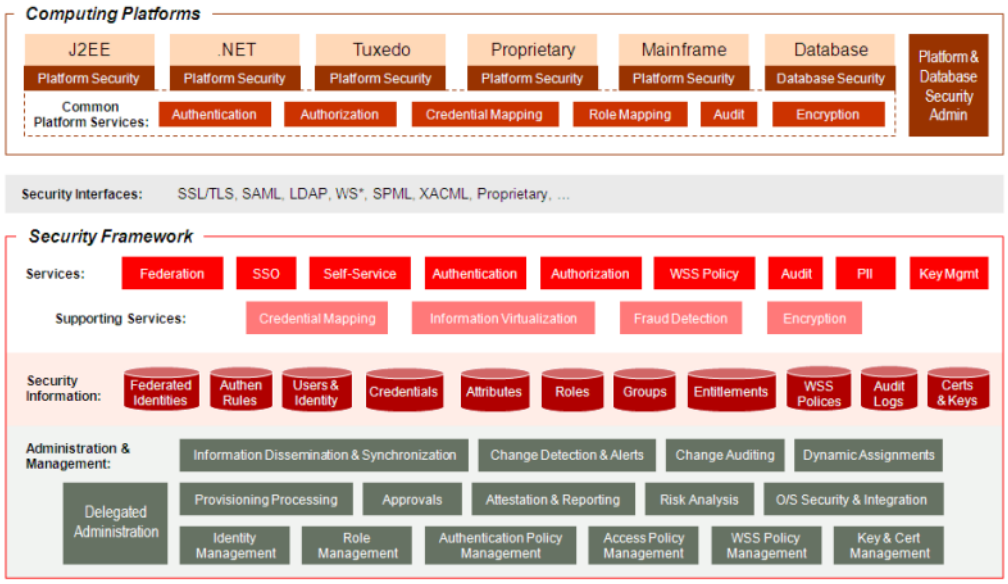
Solution platforms are not all created equal; therefore the type of services offered, standards supported, "out-of-the-box" capabilities provided, and features will vary from platform to platform. Generally speaking, mature platforms will offer a more robust set of services than nascent platforms - simply because security capabilities are often incorporated into the platform over time. For instance, J2EE platforms and relational databases provide far more capabilities now than when they were first introduced.

The security framework is based on infrastructure components that provide a collection of security services across the enterprise. The framework provides integration with the solution platforms via a set of interfaces that are primarily based on open standards. Proprietary interfaces may be added to support specific solution platforms when necessary. The framework includes a master version of all security-related information, and provides administrative capabilities to manage the enterprise security services and information.

4.3.1 Detailed View

The diagram in Figure 4–4 expands on this concept by including some examples for each of the layers. A number of common computing platforms are illustrated such as J2EE, .NET, proprietary, Tuxedo, mainframe, and database platforms. Each platform provides some form of security and a means to administer it.

Figure 4–4 Security Framework Conceptual Model



Though security services will vary across platforms, the following services are commonly provided:

- **Authentication** can be supported in a number of ways. Legacy platforms often included all functionality and data needed to authenticate users within the platform itself. In other words they were a closed, standalone security domain. Today most platforms will leverage external sources to perform authentication, either by deferring authentication to an external service, or by validating credentials against an external identity store.

  The use of an external authentication service is preferred as it provides the greatest versatility in terms of authentication methods, rules, options, and strengths. It also better supports capabilities such as SSO, centralized management, auditing, and attestation.

- **Authorization** can also be managed and performed locally, or via an external service or data store. The use of an external service is preferred for the same reasons stated above. Authorization includes coarse- and fine-grained access control. Most platforms provide greater built-in support for coarse-grained access control since fine-grained control requires greater infrastructure capabilities.

- **Role Mapping** is the ability to determine which roles a user is associated with. It is a necessary step for role-based access control. Computing platforms do this by obtaining a list of roles the user is associated with, either via a locally managed store or via a remote (common) store. The use of a common role store is recommended in order to promote information sharing, centralized management and modeling, auditing, etc.

- **Credential Mapping** is often performed in conjunction with application integration. Platforms that support credential mapping do so by determining what type of credential the remote system requires and creating that type of credential based on the identity of the user of the current session. It often involves using the same type of platform on the calling and receiving end in order to identify credential requirements. Credential mapping across platform types is often accomplished using a security federation service or integration intermediary such as a service bus or message bus.

- **Auditing** involves collecting and logging information about the activities performed on the system. For security purposes it aids in the detection and tracking of fraudulent activities. It also offers a way to determine what changes have been made to the security configuration, when they were made, and by whom. Local audit services will sometimes support the propagation of audit logs to a common service, which enables the roll-up and reporting of information at the domain level.
- **Encryption** is often handled by the platform transparently to the user. Each platform will often manage its own key and credential store, although some might support the use of a common store. In addition, a common encryption/decryption service could be useful in order to provide end-to-end encryption.

Local security interfaces within the computing platforms will often use native language semantics and message objects. These local security services are generally built for speed in order to limit the overhead associated with securing applications.

The trade-off of using remote security services is the lag time required to communicate requests and responses over the network. To rectify this, many service implementations include some form of shared and coordinated capability between the computing platforms and the security framework. Security information is either pushed to the platforms or cached by the platforms. This allows local decision making based on remote/centralized security information. By default, local platforms can rely on remote security services. They avoid remote service calls when sufficient data is present to make the decision locally.

As depicted in the conceptual view, computing platforms are supported by a common security framework. The framework provides three essential needs:

1. **A master version of security information** that represents the users, resources, privileges, and security policies of the domain.
2. **A set of services** that act on the security information to control access to resources and implement security policies.
3. **A complement of management and administrative capabilities** required to maintain the security information, control security services, and report on the current state of security across the domain.

4.3.1.1 Security Information

The security information presented in the conceptual view represents a common set of data that most organizations will have in one form or another. It is not an exhaustive set. Other types of data exist and may play a part in any organization's security model.

Security information will likely include:

- **Users and Identity.** User information will describe the end user in terms that make the user unique to the organization. Data elements may include name, employee id, SSN, and other data to establish a unique identity. Identity information, in this context, refers to the identity a user presents to the system in order to authenticate.
- **Credentials.** A user may have more than one form of credentials. The most common form is a static password. Other forms include tokens, smart cards, and encoded biometric data.
- **Attributes.** Certain attributes about the user may be maintained by the security infrastructure in order to help determine for which resources and operations a user should have privileges. Examples include job classification, work locations, and work group affiliations. These can also be used to assign users to roles, which

in turn are granted access privileges. Attributes may also include personal data, associated with identities, which need to be managed and controlled based on specific policies, (see Section 1.2.7, "Privacy and PII").

- **Groups.** Users are generally assigned to groups based on organizational structures. Groups may be organized into a hierarchy according to the organization and access may be granted to resources based on group membership.
- **Roles.** Roles represent logical functions that users perform as part of their job. They differ from groups in that all persons reporting to a manager do not necessarily have the same job title or job functions. Even persons with the same job classification might perform different functions. Roles provide a more accurate representation of resource and privilege needs.
- **Entitlements.** Entitlements (aka authorization rules) represent the policy statement of who has what privileges to what resources and under what conditions. They are used to make authorization decisions aka access control. The 'who' part can be represented by user identities, groups, or roles. The privileges are combinations of read, write, update, etc. Resources can be URLs, EJBs, SOA Services, entities, and any other object that is protected by security logic. Conditions can include time of day, physical location, and other types of measurable contextual data. Entitlements in this reference refer to both coarse-grained and fine-grained policy statements.
- **Authentication Rules.** Since not all resources have equal privacy and protection requirements, it is likely that not all resources will be protected by the same authentication mechanisms. Certain highly sensitive systems may require more stringent proofs of identity. In fact, these proofs may be dynamically altered in order to reduce the ability for hackers to cheat the system. In addition, user identities and credentials may exist in multiple physical stores. Authentication rules are a way to represent the forms of authentication required and the flow of authentication attempts between stores that are used to protect a given set of resources.
- **Federated Identities.** In order to support interoperability of systems between security domains, it is necessary to establish a mapping between identities in the local domain and those in other domains. This store represents those mappings which are used by federation services.
- **WSS Policies.** The standard way to secure Web Services is through the use of WS-Security standards, and the standard way to represent WS-Security requirements is through WS-SecurityPolicy. WSS Policies store is where security policies of Web Services are stored so they can be obtained by service consumers and enforced by service providers.
- **Audit Logs.** Audit logs depicted here represent the information gathered from all elements of the security framework. They include changes made to the security information, use of the security services, errors, retries, and other types of information that might be needed to attest to the state of security over time and monitor possible fraudulent activity. Audit logs may also include information gathered and forwarded from various computing platforms.
- **Certificates and Keys.** An organization may decide to centrally manage a set of common encryption keys and certificates. It may also want to issue keys and act as its own internal certificate authority (CA).

4.3.1.2 Security Services

Security information and common functions supported by the information are exposed to computing platforms via a collection of security services. These services improve the robustness and consistency of common security functions.

Security services should not be confused with Web Services or SOA Services (as defined in *ORA Foundation*). Like SOA Services, security services offer a means to rationalize security functions and data and offer these capabilities via an industry-standard interface. Security services may or may not be fully described by a standard service contract, mediated, and discoverable through ordinary service discovery mechanisms. Therefore security services can be made into SOA Services by adhering to the appropriate rules for service engineering. However, many security services are only meant to be accessed by other forms of infrastructure, not directly by business solutions. So the business value associated with service description and discoverability may be minimal.

Security services differ from Web Services in that not all interfaces are WS-compliant. Many security services rely on specialized standards, proprietary protocols, and specific message exchange patterns in order to function properly. It is more important that they are consumed using an appropriate standard interface protocol (where supported) than that all services use a common protocol such as SOAP.

The security services presented in the conceptual view represent a common set of services that most organizations will need. It is not an exhaustive set. Other types of services may exist and may play a part in an organization's security framework.

Security services include:

- **Authentication.** This service provides multiple forms of authentication based on the rules established for a set of resources. It relies on identity, credential and authentication rules information provided by the framework.
- **Single Sign-on.** This service provides a form of proof that computing platforms will recognize and honor as evidence of authentication. Proof may be in the form of a token, cookie, or assertion, depending on the SSO implementation used. The SSO service maybe combined with the authentication service such that SSO proof is automatically provided upon successful authentication.
- **Federation.** This service offers a way to propagate identity across security and cookie domains. It involves the mapping of identities and credentials from the current domain to what is required by the remote domain.
- **Authorization.** The authorization service represents a standard approach to all forms of access control. It leverages the identity, group, role, and entitlement policy information in order to make access decisions. It represents the policy decision, retrieval, information, and administration points.
- **Self-Service.** This service provides a means for end users (that are not administrators) to self-register and maintain their own security information for which they have access rights. Common maintenance functions include password resets, address changes, and contact number changes. This service may interact with administrative processes where approvals are needed and synchronization across information stores must be orchestrated.
- **WSS Policy.** This service provides a means to access WSS Policies for Web Services and validate and enforce policy requirements on service requests.
- **Audit.** The audit service offers a standard means to collect audit data from various sources such as computing platforms, infrastructure, and security framework components. It also offers a means to view and report on audit data collected.

- **PII.** The PII service provides access to personal information governed by privacy rules. It can be used by business solutions in order to process transactions and execute business processes, or it can be used by security services in order to evaluate access control decisions.
- **Key Management.** This service provides the ability to register and resolve public keys that are used for encryption. It supports the maintenance of PKI at the organization or domain level.

In addition the supporting services include:

- **Credential Mapping.** This service supports other services that provide identity federation by supplying the type of credential needed for a particular resource. It is mostly used by federation services.
- **Information Virtualization.** Since security information such as identities, credentials, attributes, etc. may be persisted in a number of physical stores, and each store may include some amount of unique data, this service provides a virtual view, or aggregation, of all data pertaining to a specific entity. The virtualization service can be used by other services that perform authentication and authorization in order to execute against a complete data set.
- **Fraud Detection.** This service operates behind the scenes monitoring other services with the intent of detecting possible fraudulent activity. It is mostly associated with the authentication service since that is where hacking will first be evident. It can monitor events such as failed logins and various inconsistent behavior as it occurs. It can send alerts and suspend accounts and sessions when suspicious activity is detected.
- **Encryption.** As a supporting service, encryption capabilities are often built into the infrastructure that provides other forms of services. Encryption can also be offered as a primary service for organizations that want the ability to encrypt data from end-to-end using a single message-level encryption and decryption service.

4.3.1.3 Identity Management & Administration

The key to establishing and maintaining an effective security framework is in the capabilities pertaining to management and administration. If these capabilities are lacking, then visibility and integrity of security information will naturally degrade over time. Many important capabilities are depicted in the framework and introduced below.

Security management and administration capabilities include:

- **Delegated Administration.** The ability to partition administrative capabilities such that users can perform administrative function based on their role and group/department within the organization. Although security information and services are centrally managed, administration can be partitioned and delegated to provide a certain amount of control to groups and/or individuals within the organization.
- **Identity Management.** The management of user identities, credentials, and attributes.
- **Role Management.** The creation of roles and role hierarchies based on users or groups. These may be developed from scratch or generated based on one or more existing organizational structures.
- **Authentication Policy Management.** The management of policies that dictate how authentication must be performed in order to access a particular resource. This includes the type(s) of authentication, which may involve single-factor or

multi-factor methods. If multiple unsynchronized and unvirtualized identity stores exist, it may also define the flow of authentication attempts across a set of identity stores.

- **Access Policy Management.** The management of policies that determine access rights to resources. Policies represent the assignment of access privileges on resources to roles, groups, and user attributes.
- **WSS Policy Management.** The management of policies that secure Web Services. These determine the type of identity token required, the message parts that need encryption, and if messages or message parts must be digitally signed. They also define the type of signature and encryption algorithm to use.
- **Key and Certificate Management.** The management of encryption keys and digital signatures. This may include functions related to operating a private certificate authority.
- **Provisioning Processing.** Identity management, such as the creation of a user account, is usually a multi-step process. It involves the initiation of management activities; the determination of type of user, attributes, group assignments, role assignments, etc.; obtaining certain approvals; the creation of users in various data stores, and the dissemination of information to data stores maintained by individual applications. This sequence of work is controlled and managed by provisioning processes.
- **Approvals.** The framework must provide a means for change approval, a way to inject approval steps into various management functions, and a way for approvers to interact with provisioning processes.
- **Attestation and Reporting.** It must be possible to know at any point in time how the security system is configured. Government regulations require companies to attest to privacy of information. An organization must be able to report on access policies to sensitive information and resources and be certain that they are accurate and correct.
- **Risk Analysis.** This function involves the processing of fraud detection logs and/or audit logs in order to recognize potential risks to the systems. It may involve automated processing, manual analysis, or some combination of both. The output of risk analysis is an indication of the likelihood of malicious activity, which can influence ongoing authentication and access policy management.
- **O/S Security & Integration.** This capability allows the information maintained as part of the application security framework to be used by, and/or synchronized with, security provided by operating systems. Doing so provides consistency between the identities and credentials used for application security and those used to log into PCs, UNIX systems, mainframes, etc.
- **Information Dissemination and Synchronization.** As users are provisioned and security information is created, updated, and removed, it is important to ensure that all data stores, applications, and systems that maintain the information are kept synchronized. This involves knowing where the data are kept, having the ability to integrate with them and send updates to them, and having the ability to map and transform fields as needed in order to conform to each endpoint's unique schema.
- **Change Detection and Alerts.** In conjunction with the synchronization process, it is important for the synchronizing party to know when changes have been made to a specific local security data store. If an organization has adopted a provisioning process, then this "out-of-band" change should be treated as a potential security

breach. The local change should be retracted, if possible, and security personnel alerted of the local change.

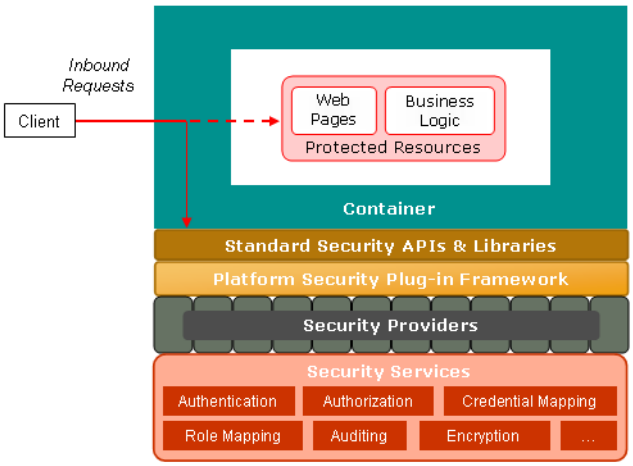
- **Change Auditing.** All management operations should be recorded in order to provide a means to retrace steps that were taken and know who performed them and when they were done.
- **Dynamic Assignments.** Many of the information management features provided by the framework are initiated by request or performed via manual actions, modeling, etc. Information otherwise remains unchanged (static). Dynamic assignments allow for security information to change, or provisioning processes to initiate based on events that occur within the organization. This most frequently pertains to group and role assignments. The security framework might provide the capability to dynamically change these assignments based on changes to organization structures and user attributes, provided of course that the triggering events are properly administered and secured. Doing so allows the system to react to these ongoing changes without waiting for groups and roles to be manually remodeled, potentially after every organization and user change is made.

Identity management and administration capabilities can be offered as services that are invoked by various processes within the organization. For example, user provisioning can be offered as a service that is invoked by an HR user onboarding process. In order to ensure proper use, identity management services may leverage common security services such as authentication and authorization.

4.3.1.4 Computing Platform

Computing platforms should insulate business logic and provide certain security services. Platforms such as J2EE offer protection in the form of container-managed security. This concept, as illustrated in Figure 4–5, creates a logical barrier around protected resources so that requests from users and other systems are intercepted before reaching them. Security checks such as authentication and authorization can be performed first to provide assurance that requests come from valid and authorized sources.

Figure 4–5 Computing Platform Security



The computing platform shown in Figure 4–5 also introduces the concept of a plug-in security framework. The purpose is to provide a means to decouple security services from the actual providers of that service. This allows providers to be selected and configured based on the needs of the solution. The computing platform provides the

container, framework, a set of default providers, and standard interfaces through which security services are accessed. Additional providers can be introduced as needed to replace, extend, or augment the default capabilities.

4.3.1.5 Database Platform

The final security barrier for an organization consists of the security capabilities provided by the database. Information stored in the database - whether it is transactional data, warehoused data, content, or security information - must be protected. It is often considered one of the most valuable commodities an organization possesses. As such one might consider security infrastructure, as described earlier in this chapter, as a way to protect the neighborhood, while database security is how one protects the jewels.

Figure 4–6 Database Platform Defense in Depth

Four important aspects of database security highlighted here are perimeter protection in the form of a smart firewall, auditing & monitoring, access control, and encryption & masking.

The SQL Firewall represents the ability to inspect SQL commands issued to the database and reject commands that are considered malicious. This layer of security helps defeat SQL Injection attacks by only allowing commands through that have been deemed normal for the application. As a firewall, this layer prevents rejected requests from ever reaching the database. Further access controls are provided at another layer in order consider the context of the request.

Auditing & Monitoring includes auditing user actions, administrative actions, data changes, and configuration settings. This protects the information from threats and exposure from privileged users as well as ordinary users. It enables the database to be rolled back in the event undesired changes are made and tracks which users performed which operations. It also includes data restoration in the event data is lost entirely.

One must ensure that access to the database is controlled and limited to those that need it. User accounts and role assignments must be maintained for the database much like they are maintained for other IT assets such as applications. Administrative accounts must be strictly controlled to ensure that information is safe both from malicious hacking as well as accidental damage. Access can be controlled along a number of dimensions including user roles, attributes, locations, times, request parameters, etc.

Encryption is important within the database as it protects sensitive information from view, even to those that have system access. Ideally, sensitive data fields are encrypted automatically while they are persisted. This makes it easy to manage and less likely to be forgotten during the solution development process. Fields can also be masked depending on data classification levels. In addition, fields can be permanently masked, or transposed, when copies of production data need to be used for development or testing purposes.

The next chapter describes how the features and capabilities of the framework are logically applied to common security scenarios.

5

Logical View

The Logical View of the architecture describes the various components of the security architecture and common interactions. Given the large number of components that comprise the architecture, the Logical View is broken down into several scenarios. Each scenario highlights a group of components and a set of common operations that they support.

Components and interactions depicted in the Logical View are not specific to any product or set of products.

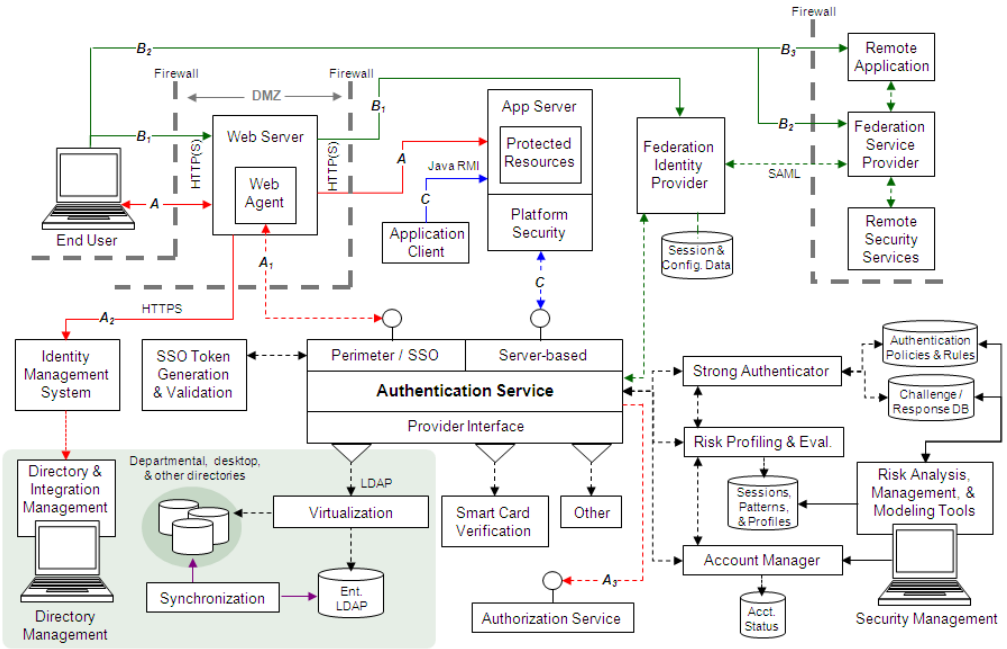
5.1 Common Scenarios

The following sections describe the logical architecture by breaking it down into a series of common scenarios that illustrate parts of the architecture.

5.1.1 Authentication and Identity Federation

The diagram in Figure 5–1 depicts components of the authentication service, SSO, identity federation, and access-related fraud detection.

Figure 5–1 Authentication, SSO, Identity Federation, and Fraud Detection



The authentication service is designed to support multiple types of authentication. The diagram depicts mechanisms that include LDAP, smart card, and others. Authentication mechanisms "plug into" the service via a published provider interface. The service can be extended to provide stronger forms of authentication such as challenge/response interactions based on pre-configured questions and answers. This allows the service to evolve to meet the organization's security needs without changing the service itself.

Since identity information for a particular mode of authentication might be persisted in more than one identity store, a virtualization service is used to logically combine these stores into a single virtual view. For example, employees may be maintained in a company LDAP while contractors and partners are maintained in a database. Likewise, multiple departmental LDAPs and databases may be used. The number and types of physical stores can be managed by the virtualization service in a way that simplifies authentication.

In addition to virtualization, the architecture includes a service to synchronize identity sources. This is necessary when similar data are stored in more than one directory or database. The service ensures that data are kept consistent when changes are made. A directory management component is also shown to support management of the enterprise LDAP directory(s) as well as configuration of the virtualization and synchronization services.

The diagram illustrates three forms of authentication. The lines marked as interaction A pertain to authenticating a user at the perimeter. The lines marked as interaction B illustrate identity federation across security domains. The lines marked as interaction C depict authentication of an application client. Dotted lines represent interactions behind the scenes between security components.

Perimeter authentication, as described previously in Section 2.1.2.1, involves the use of a web agent, authentication service, and SSO token generation and validation service. When a user attempts to access a protected resource, the web agent intercepts the request and checks for the presence of a security token. If a token is not found, the agent redirects the user to the authentication service where the user is prompted to authenticate (A1). If authentication is successful, then a SSO token is created and returned.

Interaction A also depicts interaction with the identity management system (A2). This occurs when the user's password needs to be reset or changed. The authentication service can detect such conditions and redirect the user to a self-service management interface.

In addition, interaction A3 illustrates the connection between authentication and authorization services. The system can be configured to perform authorization checks once authentication completes. Access to protected resources will only be granted if both authentication and authorization are successful. Authorization is described in more detail in the next section.

If all security checks succeed, then interaction A continues with the token being presented to the application server hosting the protected resource. The application server can either validate it based on a digital signature, or use a service the server trusts to perform validation. The application server, like most computing platforms, has security capabilities built in, as represented here by platform security. It handles security checks on behalf of the application resources under its control. It may also perform authorizations (not shown). The user can now access other protected resources within the cookie domain by presenting the SSO token as proof of authenticated identity.

Interaction B illustrates a request made by the user to a web resource in another security domain. In this example the remote domain does not have the ability to authenticate users of the local domain. Instead, it trusts an identity provider to vouch for user authenticity and pass along an assertion of identity. The request (B1) is forwarded to the local federation identity provider in order to generate a SAML assertion that is trusted by the remote service provider. Since the user has already authenticated in the local domain, e.g. an SSO token exists in the user's session, the identity provider can use the identity contained within it to establish a remote identity. The local identity can be mapped to a different identity that is known by the remote domain. Had the request not contained a token, then the user would either be authenticated by the federation identity provider or redirected to the local authentication service where authentication and SSO token generation would occur.

Once authenticated, the identity provider redirects the request, along with the federated assertion to the service provider (B2). The service provider validates the assertion and uses its own local security services to create a session token to be used for further (direct) interaction with the user (B3). This token is placed in a cookie that corresponds to the remote resource's cookie domain.

Interaction C illustrates what happens when one application interacts with another. Here perimeter authentication is not performed. Instead the client presents identity and credentials that are authenticated via a server-based authentication interface. Both perimeter and server-based authentication can leverage the capabilities of the authentication service.

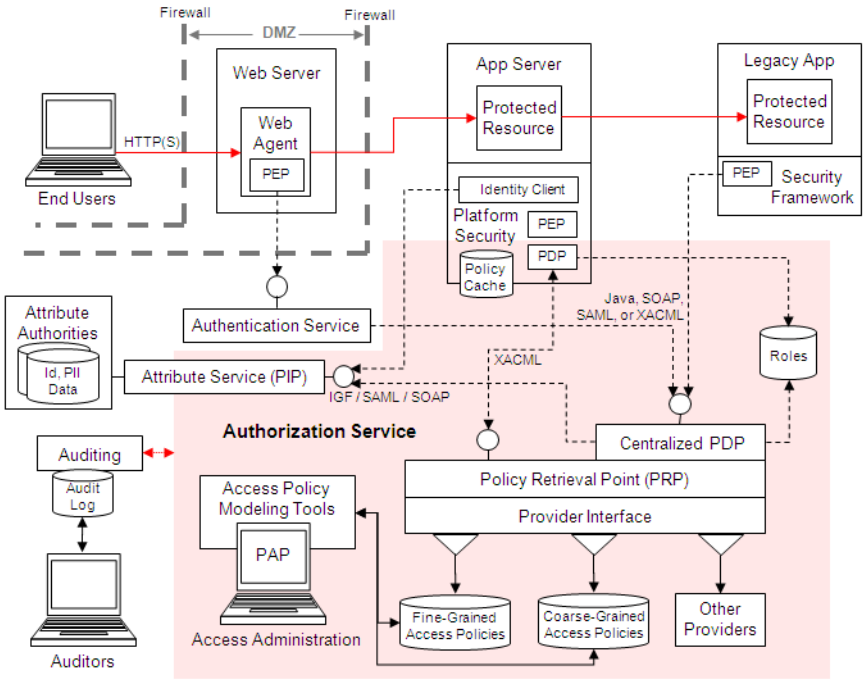
Finally, a key concern of any authentication service, particularly one that handles end users, is fraud detection. Since this service is the first line of defense against fraud, it is important to monitor authentication requests to detect unusual behavior. The fraud detection server/service is fed data pertaining to authentication, such as: user identity, device identity, location, and time of day. It logs such data and uses rules to detect unusual occurrences that might indicate fraud, such as users logging in from different locations in a short period of time using the same identity, or the use of many different computers for the same user.

The strength of authentication may be automatically adjusted based on specific risk factors. For example, if a user attempts to log in from a new device or from a suspicious location, then security questions may be asked in addition to the normal user id and password. Authentication rules can be established to determine which factors trigger stronger authentication requirements. The security framework supports the modeling and administration of fraud detection algorithms, as well as the configuration of authentication rules. Authentication rules provide a means to not only combat fraud (by adjusting authentication strength or disabling accounts), but also to define the types and factors of authentication needed for particular resources. The latter capability underscores the notion that not all resources have the same degree of confidentiality, and not all forms of authentication have the same degree of protection. Rules can stipulate multi-factor authentication schemes for resources of higher sensitivity.

5.1.2 Authorization and Access Policy Administration

The diagram in Figure 5–2 depicts the components that combine to make up the authorization service, and some of the interactions commonly associated with it.

Figure 5–2 Authorization and Access Policy Administration



The authorization service is much more distributed in nature than other types of services. This is due to the "need for speed" with respect to making access control decisions. Unlike authentication, where (ideally) decisions are handled once, or once in a fairly long while, authorization may happen with every system interaction. In fact it can happen multiple times with each user request depending on the chain of events that occurs as a result of the request.

In our example, authorization happens three times, once by the web agent at the perimeter, once by the application server, and once by a legacy application. Authorization at the perimeter was mentioned in the authentication scenario, where the authentication service invokes coarse-grained authorization of an authenticated user request. This restricts traffic through the firewall to the application server and acts as a first line of defense.

The application server may choose to perform additional coarse- or fine-grained authorizations. These checks can be more specific, e.g. based on specific operations, parameters, and user attributes involved in the request. In this scenario the application server makes a request of a legacy application, which performs its own authorization checks.

The components of the authorization service follow the model used to present XACML, previously described in Section 3.2.4. Here we see policy enforcement being handled by each system's security platform. The security platforms may or may not provide the ability to make policy-based decisions. In this case the application server uses an embedded policy decision point (PDP), while the other platforms leverage a shared, or centralized PDP.

Access to the remote PDP can be handled using a number of protocols. Common interfaces may include Java, .NET, and SOAP. A SAML authorization query may be used to standardize access based on the XACML specifications. In a pure Java environment this specification may be implemented using the "Open AZ" authorization API. Centralized PDPs may be deployed in clusters along with the resources they protect, or in close proximity, in order to minimize network overhead.

PDPs get policy information from the policy retrieval point (PRP), represented here as part of the authorization service. Policies are retrieved from policy stores. PDPs may get policy information pushed to them when they are initiated or when policy changes are made. In this case the PDP caches policies in order to accelerate the decision process. The authorization service also offers a provider interface that supports multiple authorization schemes. In this example the service supports both coarse-grained and fine-grained access control.

The authorization service also offers a provider interface that supports multiple authorization schemes. In this example the service supports both coarse-grained and fine-grained access control. The provider pushes access policies to the PDPs so they can make decisions locally.

Access policies are modeled and managed by the policy administrators. Policy modeling involves combining roles, attributes, operations, and resources together. Most of these data are obtained from the identity management system, which is described in Section 5.1.4. Access policies are provisioned back through the identity management system in order to ensure approvals are granted and all data stores are updated properly.

PDPs may require additional information, such as user attributes, etc., in order to make access decisions. The policy information point (PIP) provides this sort of information. In order to simplify integration between information sources and all the PDPs, an attribute service is used. The PIP gets requests from the PDPs, often using SAML attribute requests, and obtains the needed attributes in a number of ways, which may include direct access to identity stores, databases, and data services.

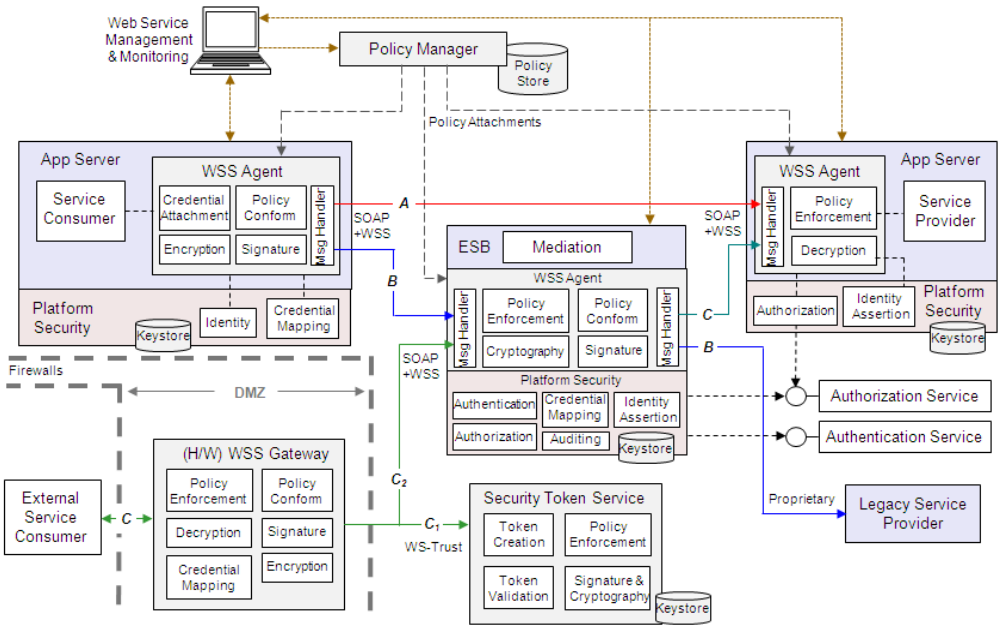
PIPs may also get requests for personal information directly from application code. These requests are necessary in order to support ordinary user interactions, such as displaying and updating addresses, phone numbers, job titles, etc. In order to standardize the retrieval of such information, the attribute service is designed to follow the IGF standards, (described in Section 3.5).

Auditing is provided in order to track events such as access requests, access decisions, and policy changes. Components such as PDPs and PAPs must log activity that can be reviewed by an auditor. Other components may generate audit log entries as well. Individual connections to the auditing component are omitted in order to avoid over-complicating the diagram.

5.1.3 Web Service Security

The diagram in Figure 5–3 depicts components that can be used to establish and maintain security for Web Services. Components include two service consumers, a service provider, an enterprise service bus (ESB) to mediate service requests, a Web Service Security (WSS) gateway, a security token service (STS), and a policy manager.

Figure 5–3 Web Service Security



Security applied to Web Services is defined using WSS policies, which are WS-SecurityPolicy XML documents or an equivalent. Policies dictate the method used to represent user identity and authenticity, as well as requirements for encryption and digital signatures applied to the message or message parts. These policies are authored and assigned to Web Service providers, mediators, and clients by the policy manager.

Policies may be assigned to more than one Web Service. They are discovered and assigned at design-time. The preferred method of client-side security compliance is through the use of WSS agent infrastructure to detect and adhere to policies automatically at run-time. This allows changes to be made without re-coding and re-deployment. The agent interacts with the application server's security platform in order to create the proper identity assertions and to sign and encrypt messages on behalf of the client. This concept is further discussed in Section 5.1.6.

Likewise, policy validation is handled automatically by a WSS agent on the service provider side. Service requests are validated against the security policy, decrypted, and identity is asserted. The service provider's security platform can perform authorization against the caller's identity. Interaction A indicates this basic pattern, with dotted lines used to represent infrastructure interactions.

Interaction B in Figure 5–3 represents a case where security used by the service provider is not supported by the client. In this case an ESB is used to mediate one form of security with the other. The same policy discovery and validation take place, this time it happens between the client and ESB. The ESB then interacts directly with the service provider, which in this example uses a protocol other than SOAP.

Interaction C represents an external client accessing a Web Service. In this case a WSS gateway is positioned in the DMZ. It validates the XML request in order to eliminate mal-formed requests, and/or requests that might represent denial of service (DoS) attacks such as high volume traffic and message sizes that are deemed too large to support. Satisfactory requests are passed to the ESB, where routing and security mediation take place.

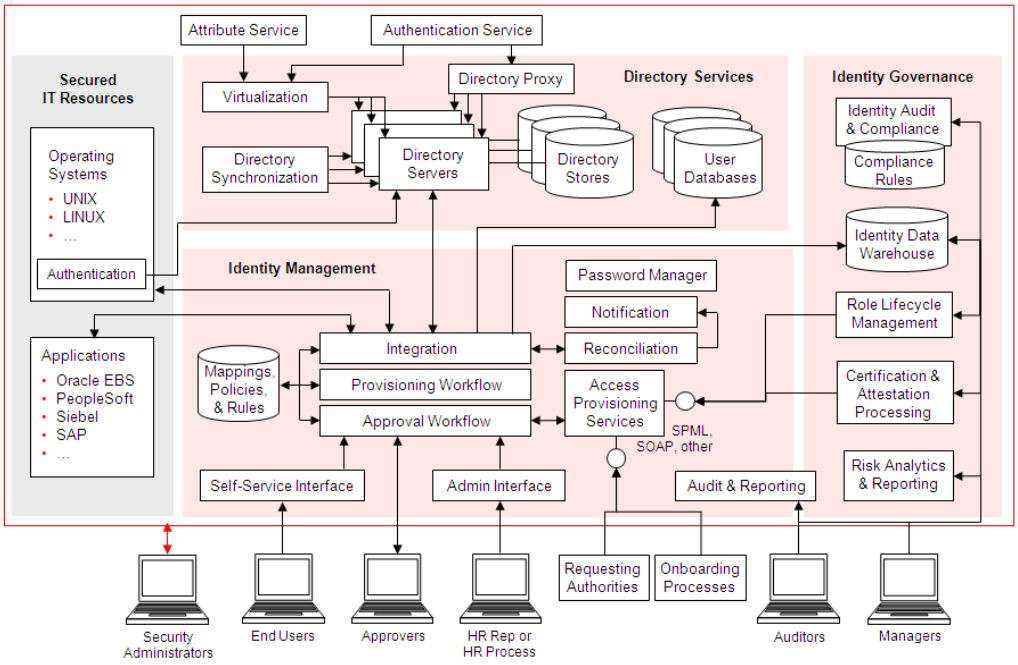
In this scenario the external client happens to be from another security domain. Service providers in the local domain do not have a direct trust relationship with external

clients. The architecture includes a security token service that mediates identity and trust. It accepts external client requests (C1) following the WS-Trust specification, performs its own authenticity validation, and generates WSS tokens that the local domain will accept. Interactions (C2) are then possible with the local ESB.

5.1.4 Identity Management & Governance

The diagram in Figure 5–4 illustrates the interaction of significant components that comprise an identity management and governance system.

Figure 5–4 Identity Management and Governance



This aspect of the security framework establishes a deliberate, consistent, and cohesive set of security information, which in turn secures the IT assets of the organization. It does this by providing proper methods to provision, administer, and audit security that ensure consistency between types of information, stores of information, and uses of information.

Identity management starts with a provisioning request, typically either from an end user, from an administrator, a requesting authority (see SPML) an HR representative, or an HR onboarding process. The system provides a number of entry points that accommodate the most efficient ways to initiate provisioning. Some entry points are represented as user interfaces, while others are represented as common provisioning services.

A self-service interface allows end users to make requests and handle some tasks, such as password resets and personal information changes, completely on their own. It also provides an interface for persons such as HR representatives to provision users manually. Another administrative interface can provide access to security administrators that need less interaction and prefer a more efficient data entry point.

These interfaces are entry points to the provisioning and approval process engines. The process engines orchestrate the activities associated with each type of provisioning request, either for creation, update, or removal of identity information.

They are controlled by provisioning rules, which are maintained by security administrators. All work performed by the processes is logged for auditing purposes.

Approval processes interact with the various users configured to grant approval for each type of request. This not only helps to ensure that security changes are approved, but also serves as a method to stipulate and ensure each type of change has the proper forms of approval - that ad hoc changes are not supported.

The heart of the system is the component that manages provisioning workflows. It controls the flow of information and initiation of approvals. Since provisioning can be performed to many different directories, databases, systems, and applications, the process engine relies on an integration layer that handles individual system interactions and necessary data transformations. Integration typically involves deploying some form of adapter for each endpoint that the provisioning process will manage.

Since security data is often managed by individual applications, particularly legacy or packaged applications, the provisioning system must integrate and synchronize with these applications as well. This provides a means for users to have a common identity and password across a greater number of applications. It also provides a means to manage roles and access privileges across applications - even those that maintain their own security constructs.

Likewise, identities and passwords can be synchronized with access mechanisms used by operating systems. This enables a user to log into a Windows, UNIX, LINUX, or mainframe system with the same credentials used to access applications.

The provisioning system must not only ensure that changes are pushed to the managed endpoints, but also that any changes made locally at each endpoint are detected. Otherwise there would be no way to ensure that information is consistent across the organization and that all security data and configurations have been approved. The architecture includes a reconciliation component to detect local changes and attempt to eliminate them. It also includes a notification component that will alert security administrators when such changes are detected. These situations will also be logged so that auditors will know when and where inconsistencies in the system existed.

An organization may have more than one directory system. In these cases it may be necessary to either synchronize data across directories or create an aggregate view of security data that can be consumed by applications that need information stored in multiple directories. Directory integration can provide a permanent virtual view across multiple directories, or can act as a temporary solution while directories are merged into a single system. This virtual view can be leveraged by the authentication and attribute services, offering an abstraction layer between those services and various underlying directories.

Directories may also be strategically placed for geographic load distribution. For instance, directories may be deployed in each continent where large numbers of users exist, with user accounts partitioned accordingly. Likewise, redundant directories may be deployed for fault-tolerance and scalability. In either case proxies play a key role in routing requests. They also will serve as a filter to prevent inappropriate access to directory servers as well as a method to prevent DoS attacks.

Working in conjunction with identity management is identity governance. This portion of the architecture is designed to support activities related to ensuring that roles and privileges are properly assigned and maintained.

At the heart of identity governance is an identity warehouse. This is a data store designed for intelligence-like queries where managers and auditors can view the security landscape in its entirety. It is populated by either online integrations or offline

ETL-like processes. The identity warehouse is flexible to allow various types of information to be stored in a way that supports the security audits of the organization.

Components of identity governance include identity audit and compliance, and risk analytics and reporting. These components are used to help identify potential risks due to conflicts of interest and/or privileges that may violate internal compliance rules. In addition, governance includes processes for certification and attestation, which is important for ensuring compliance with government regulations. These processes feed into identity management provisioning in order to institute changes that need to be made. Finally, governance includes role lifecycle management, which provides for proper creation, administration and versioning of role assignments. Role lifecycle management processes also invoke identity management provisioning services.

Overall, identity management must support the ability of an auditor to determine who has access to what resources and under which conditions. This is accomplished by the components described in this section, which provide:

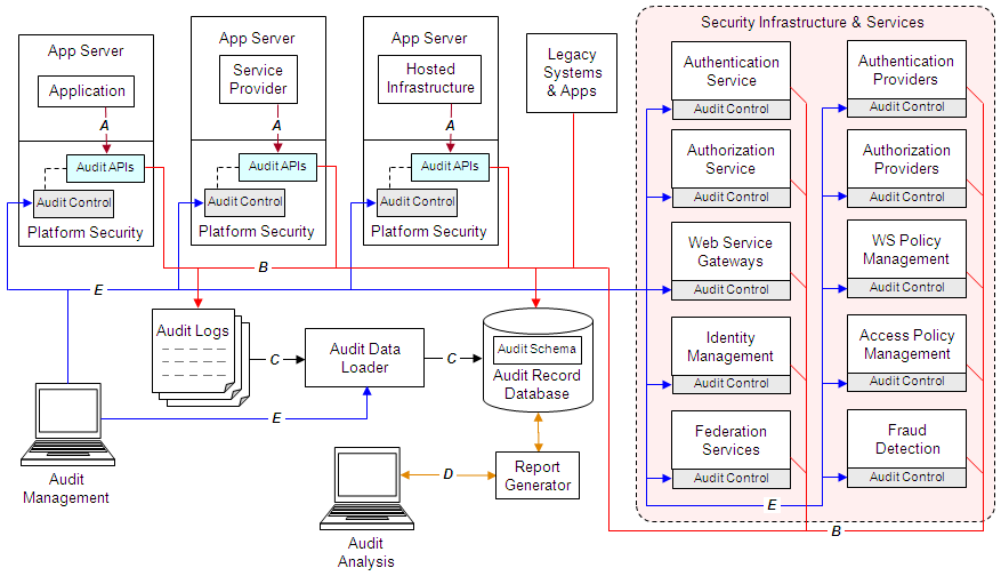
1. A set of well-described and consistently executed processes that control the configuration of security across the organization
2. A means to detect and log out-of-band changes, eliminate inconsistencies, and notify security personnel when changes are detected.
3. An audit log of all changes made to security, including the change made, user that made the change, and date and time of change.
4. A reporting mechanism that supports the query of current information and changes made to access policies, role assignments, users, etc.

Finally, the entire identity management system must itself leverage roles and access policies in order to grant access to its capabilities appropriately. The system must support delegation of duties based on job functions and organizational boundaries. In this way centralized management does not impose limitations on who can administer the system. Administration can be federated across multiple groups with privileges delegated to individuals within the groups that are trusted to perform certain functions.

5.1.5 Auditing

Figure 5–5 depicts a framework to support auditing of security events in the solution environment. It is comprised primarily of: solutions hosted by platforms that capture audit events, systems such as security infrastructure that capture events, event storage, audit management, and audit analysis.

Figure 5–5 Auditing Logical Framework



Audit events can be generated by a number of sources which include applications, SOA Services, infrastructure components, legacy systems, etc. The diagram depicts some of these sources being hosted by application servers, while others (legacy systems) are not. In the case of application server hosting, platform security provides APIs to accept and capture audit events reported by the hosted applications, as indicated by flows (A). Platform security generates events of its own based on user activity such as failed logins, access decisions, etc. It also provides an audit control mechanism to support the management of event capture. This allows remote management of the type of events being captured.

Components listed in the Security Infrastructure & Services area represent logical security components described in previous sections of this document. These components may or may not be hosted by an application server. The illustration is not intended to imply hosting type but rather to illustrate a common theme of audit control and event capture. Each infrastructure component generates audit events based on activity in the environment, either initiated by end users or security administrators.

Audit events are stored in a common database, (via flows B). This provides the ability to search for events, analyze activity, and generate reports based on events collected across a number of systems, applications, SOA Services, and infrastructure. Systems that do not support event logging directly to a database, or are not capable of capturing events in a format compatible with the audit schema, may log events to flat files. An audit data loading component is included to support transformation and loading of events from files into the audit database, (marked as flows C).

A key capability of the audit framework is the ability to centrally manage auditing across the entire security domain; (flows E). This allows an administrator to configure the type of audit records to capture based on current security needs. Configuration is performed and managed centrally, and events across the domain are generated in a consistent manner.

The audit framework includes components to support security reporting and analysis; (interaction D). The report generator allows security personnel to generate reports based on specific types of activities such as users created and deleted, user transactions, authentication and authorization failures, policy violations, etc. Queries

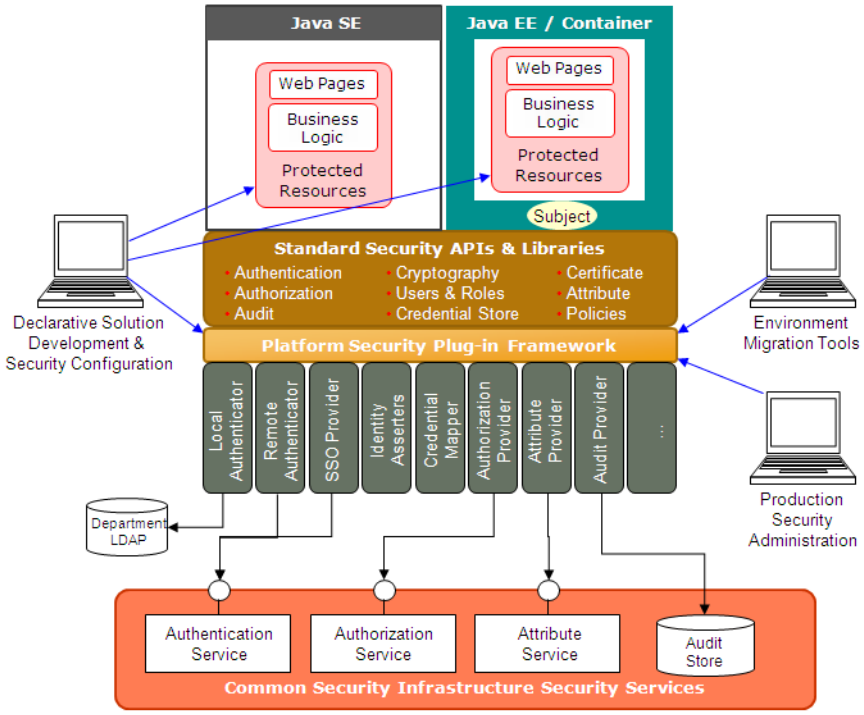
may be initiated based on user identity, time of day, event types, and other attributes of the common event record.

In order for event capture, management, and reporting to work seamlessly across multiple audit sources, each component must support a common event taxonomy and record structure. The common taxonomy enables consistency of event collection. For instance, if the capture of successful authentication events is enabled, then all components must recognize what type of event that is and what data to capture. Likewise, the common record structure ensures that events correlated and reported across multiple components will contain similar and consistent data elements.

5.1.6 Platform Security Framework

Figure 5–6 illustrates the use of the security framework that connects Java SE and Java EE platforms to common security services.

Figure 5–6 Platform Security Framework



The security framework illustrates a number of key components required by computing platforms, such as:

- Security providers that act as clients to common security services. The providers surface security services to protection logic, such as the Java EE container, via standards-based APIs.
- A plug-in framework that supports the use of various providers of security services. The diagram above illustrates the choice of multiple authentication providers and shows examples of other types of providers commonly required by business solutions. Providers plug into the framework by implementing common provider interfaces. The backing security mechanisms used by the providers are not exposed directly to the framework thus allowing the security system to evolve and expand without code changes to the platform itself or the applications it supports.

- Standard security APIs and libraries to reduce the complexity of integrating security into an application (or container). They provide a full range of capabilities without the need to locate and integrate 3rd party libraries into each solution deployment.
- Declarative security controls that allow access restrictions for pages, page flows, EJBs, Web Services, etc., to be declared at design-time. Developers can annotate code with protections directly from the IDE. This eliminates the need to develop security code manually, and instead leverages the platform and container to interpret security declarations.
- Migration tools to support the lifecycle of a solution from development environment, through test environments, and into production. These tools are used to migrate security configurations and enable the solution to integrate with security services that exist in each environment.
- Production security administration allowing administrators to modify security in the production environment. It also enables multiple solution platforms to be administered from a single administrative platform. This helps reduce administrative overhead and promotes consistency across solution deployments.

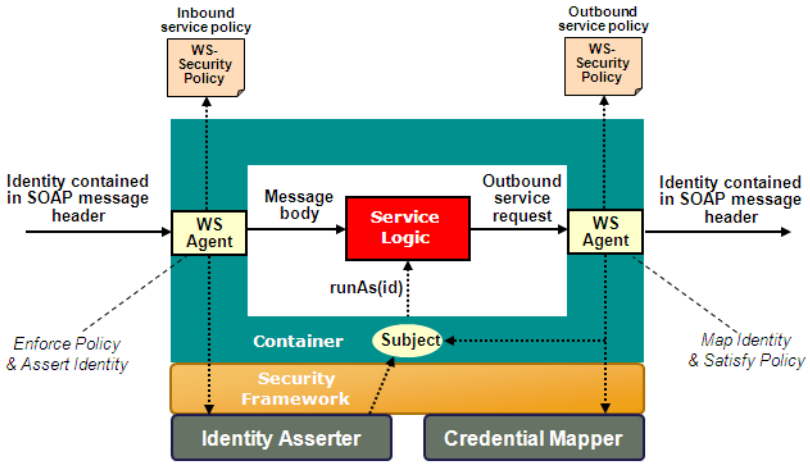
In a Java EE environment the container and security framework work together to secure protected resources. Client requests are intercepted by the container and passed to the platform security framework where they can be authenticated and/or authorized. Based on this example, authentication can be performed against the local authenticator provider, which uses a local departmental LDAP, or against the shared authentication service. The choice is based on a configuration made by the platform security administrator. The example also shows a provider to support SSO, so authentication can be avoided if an SSO cookie or token is detected in the request.

If authentication is successful, the identity asserter creates a Subject, which is used to identify the user. Actions performed by the user will be associated with the Subject so that the caller's identity will be known. If outbound requests are made by the business logic, identity can be included by the container based on the Subject.

5.1.6.1 Container-Managed Web Service Security

Figure 5–7 depicts Web Service agents working in conjunction with the container and platform security framework to provide security for Web Services. The agent at the receiving end is responsible for enforcing WS security policies, verifying digital signatures, and decrypting incoming messages.

Figure 5–7 Container-managed Web Service Security



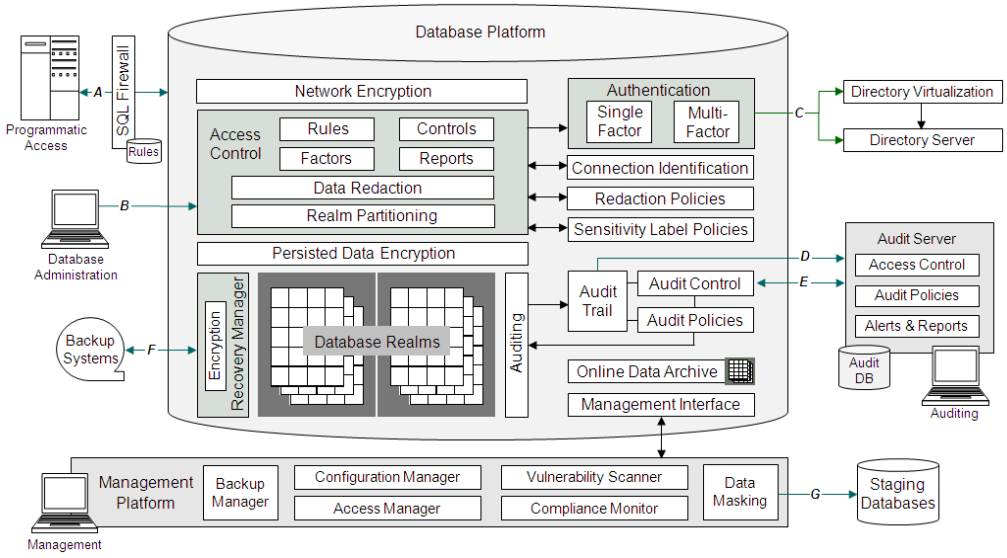
Identity is propagated between resources and is surfaced to business logic without the need for custom coding. As shown, service logic is fully supported by the container. It does not need to process inbound message headers or inject identity into outbound requests. When an inbound request is made, the container intercepts the request and saves the caller's identity. Service logic is "run as" the caller, which is also known as impersonation (though the user did not log into the service logic directly, it impersonates the user and thereby inherits the user's entitlements). This supports access control based on user identity.

If the Web Service invokes another Web Service as part of its processing logic, identity is automatically injected into the outbound request. A credential mapper creates the appropriate type of credential according to the outbound service policy. This relieves the service logic of the need to process and propagate identity, thus simplifying service development. The agent on the sending end signs and encrypts the outbound message and inserts the proper credential type based on the policy of the Web Service being called.

5.1.7 Database Security

Security within the database platform must be quite comprehensive due to the tremendous value information has to the organization, and the many common threats. Figure 5–8 illustrates a number of components within and adjacent to the database.

Figure 5–8 Database Security



Starting with ordinary programmatic access, (interaction A), the database is protected by a SQL firewall. This component inspects SQL commands and blocks commands that are not permitted by firewall rules. This form of protection helps to eliminate malicious access such as SQL-injection or any clients that attempt commands that have not been previously approved.

The database must support encryption of data over programmatic interfaces. It must also incorporate a mechanism that controls access over these interfaces as well as administrative interfaces, (interaction B). Access control is role-based and rule-based. It limits operations based on the user's role as well as rules, such as IP address, authentication mechanism used, and program name. In addition, it limits access to columns and rows based on user and data classification rules and redaction policies. It also limits DBA access to realms, or portions of the database, in order to support the principle of least privilege. Reports provide insight into account management, roles, privileges, and access attempts.

Access Control leverages identity management and authentication mechanisms based on internal or external directories and security services, (interaction C). Authentication can be in the form of simple, single factor authentication, i.e. user id and password, or multi-factor authentication, which involves the use of tokens and/or certificates. Policies and clearances used to evaluate classification labels can also be managed externally and shared across multiple databases.

The database includes many important auditing features, including:

- Logging of transactions performed on persisted data.
- Connection identification; support for passing and tracking end user identity (in addition to system identity used for establishing and pooling connections) on operations performed over programmatic interfaces.
- Logging of administrative operations such as changes to tables, indexes, security settings, and data.
- Support for centralized audit repository and remote audit policy management. Audit data is transmitted to a secure centralized store where reports can be run and alerts triggered. Local database audit policies can be managed locally or globally. (Interactions D & E).

- Access control to protect audit log entries and policies from unwanted viewing or modification.

The database also has the ability to maintain online historical archives in order to fulfill data retention requirements, support internal audits, and enable human error correction.

Information within the database can be persisted in encrypted form. That way it is not viewable in the clear, even to DBAs with full access privileges. Encryption is handled automatically, (transparently to the client), based on predefined rules. Data can also be transferred to backup media in encrypted form, (interaction F). This, along with network encryption, completes the end-to-end data protection path and enables secure archiving, even if backup media is lost or stolen.

The database can be managed as a standalone entity or as part of a collection of resources. A management interface allows remote management via a management platform. The management platform can be used to manage multiple databases (individually or collectively).

The database management platform includes components used to track and manage database configuration, compare it to pre-defined configuration policies, scan for vulnerabilities based on configuration and patch levels, and provide compliance metrics. It also controls backup and recovery operations with standard network attached storage (NAS) devices using Network Data Management Protocol (NDMP).

Finally, in order to enable testing of applications, data are extracted from production databases and loaded into staging areas. The database management system includes a masking component in order to transpose actual values into false values while maintaining rules of integrity, (interaction G).

6

Product Mapping View

ORA Security presents an abstract reference architecture for security and mappings to Oracle products. The logical architecture, defined in the previous chapter, is intended to show components derived from a conceptual model. Components of the logical model are not intended to represent a 1:1 direct mapping to specific products or product components.

This approach allows a customer to define an architecture that best serves their needs and select products based on fit and merit to implement the architecture. An implementation of this architecture may contain a mixture of products from Oracle and other sources.

This section describes how Oracle products fit together to realize the security architecture defined in the previous chapter.

6.1 Products Included

There are a number of products and product options from Oracle that can be used individually to satisfy specific security needs, or used in combination to establish a complete security framework. Note that product names, features, and options may evolve and change over time. Please consult your local sales representative or visit http://www.oracle.com for the latest product information.

The following products are included in the product mapping view:

- **Oracle Access Manager (OAM)** - OAM provides an identity management and access control system that is shared by all applications. It offers a centralized and automated single sign-on (SSO) solution for managing who has access to what information across IT infrastructure.
- **Oracle Adaptive Access Manager (OAAM)** - OAAM provides superior protection for businesses and their customers through strong yet easy-to-deploy multifactor authentication and proactive, real-time fraud prevention.
- **Oracle Identity Manager (OIM)** - OIM is a user provisioning and administration solution that automates the process of adding, updating, and deleting user accounts from applications and directories; and improves regulatory compliance by providing granular reports that attest to who has access to what resources.
- **Oracle Identity Analytics (OIA)** - (formerly Sun Role Manager) provides enterprises with the ability to define and manage roles and automate critical identity-based controls. In addition to managing the role lifecycle, OIA provides key identity governance capabilities including auditing, reporting, attestation, certification, and analytics.

- **Oracle Identity Federation (OIF)** - OIF provides a multi-protocol federation server that works along with existing identity and web access management systems to securely share identities across security domains in order to facilitate interaction with vendors, customers, and business partners.
- **Oracle Internet Directory (OID)** - OID serves as the central user repository for Oracle Identity Management, providing a scalable, secure, high-performance LDAP data store for enterprise applications and Web Services.
- **Oracle Virtual Directory (OVD)** - OVD virtually aggregates identity information from multiple sources and presents a real-time unified view without storing or copying the identity data itself.
- **Oracle Directory Server Enterprise Edition (ODSEE)** - (formerly Sun Directory Server Enterprise Edition) is a high-performance directory service that supports data distribution, and maximum availability and scalability. ODSEE includes its own built in database, proxy server, and supports integration with Microsoft Active Directory.
- **Oracle Web Services Manager (OWSM)** - Oracle Web Services Manager is a J2EE application designed to define and implement Web Services security in heterogeneous environments, provide tools to manage Web Services based on service-level agreements, and allow the user to monitor run-time activity in graphical charts. Oracle Web Services Manager is delivered both as a standalone product and as part of the Oracle SOA Suite.
- **Oracle Security Token Service (STS)** - STS establishes a trust relationship between online partners through Web Services. STS provides both standard and proprietary security token issuance, validation, and exchange.
- **Oracle Entitlements Server (OES)** - OES externalizes and centralizes fine-grained authorization policies for enterprise applications and Web Services.
- **Oracle Platform Security Services (OPSS)** - OPSS provides security APIs for Java SE and Java EE platforms that insulate application developers from underlying identity systems and allows them to focus on business logic. OPSS is an enterprise-grade, standards-based, and portable security platform used by all Oracle Fusion Middleware 11g components and Oracle Fusion applications. As the security platform used by Oracle for its own products, OPSS is now also available to customers of Oracle Fusion Middleware for writing custom applications.
- **Oracle JDeveloper** - is a complete IDE for Java solution development. It provides visual and declarative tools for JSF, JSP, EJB, ADF, Oracle Fusion Middleware, and Oracle application development. Oracle JDeveloper provides wizards that allow OPSS services to be invoked from the development environment. It is also integrated with OWSM to enable Web Service policy attachment at design-time.
- **Oracle Database (ODB)** - Oracle Database delivers industry leading performance, scalability, security and reliability. It provides comprehensive features to easily manage the most demanding transaction processing, business intelligence, security, and content management applications. Oracle Database includes features such as transparent data encryption, Enterprise User Security (EUS), Oracle Database Vault, and data masking.
- **Oracle Advanced Security (OAS)** - OAS combines network encryption, database encryption and strong authentication together to help customers address privacy and compliance requirements. With OAS, customers can transparently encrypt all application data or specific sensitive columns, such as credit cards, social security numbers, or PII.

- **Oracle Database Firewall (ODF)** - ODF is the first line of defense for both Oracle and non-Oracle databases. It monitors database activity on the network to help prevent unauthorized access, SQL injections, and other forms of attack. ODF uses positive (white list) and negative (black list) security models to validate SQL commands before they can reach the database.
- **Oracle Database Vault (ODV)** - ODV addresses common regulatory compliance requirements and reduces the risk of insider threats by preventing highly privileged users (DBA) from accessing application data, enforcing separation of duty, and providing controls over who, when, where, and how applications, data and databases can be accessed.
- **Oracle Label Security (OLS)** - OLS helps organizations address security and compliance requirements using sensitivity labels such as confidential and sensitive. Sensitivity labels can be assigned to users in the form of label authorizations and associated with operations and objects inside the database using data labels.
- **Oracle Data Masking (ODM)** - ODM helps organizations comply with privacy and confidentiality laws by masking sensitive or confidential data in development, test, or staging environments. It uses an irreversible process to replace sensitive data with realistic-looking but scrubbed data based on masking rules and ensures that the original data cannot be retrieved, recovered, or restored.
- **Oracle Total Recall (OTR)** - OTR provides a secure, efficient, easy-to-use, and application-transparent solution for long-term storage and auditing of historical data. OTR stores historical data in secure, tamper-proof databases while keeping it accessible to existing applications.
- **Oracle Audit Vault (OAV)** - OAV automates the collection and consolidation of audit data. It provides a secure and highly scalable audit warehouse, enabling simplified reporting, analysis, and threat detection on audit data. In addition, database audit settings are centrally managed and monitored from within OAV, reducing IT security cost.
- **Oracle Secure Backup (OSB)** - OSB provides an integrated, efficient, easy-to-use, policy-based backup solution that encrypts data to tape to safeguard against the misuse of sensitive data in the event that backup tapes are lost or stolen.
- **Oracle Virtual Private Database (VPD)** - is a feature of Oracle Database that supports complex rules to protect database tables. It is used when other forms of role- and rule-based access controls are not sufficient. VPD enables the redaction of data, either rows or columns, based on established policies and the evaluation of run-time parameters.
- **Oracle Enterprise Manager (OEM)** - OEM is a family of integrated management products designed to manage Oracle environments. OEM enables centralized management functionality for the complete Oracle IT infrastructure, including systems running Oracle and non-Oracle technologies.
- **Configuration Management Pack (CMP)** - CMP forms the centerpiece of OEM's ability to manage assets, configurations, compliance, and automate IT processes. It captures and centralizes the information about all hardware and software resources in the enterprise. CMP combines discovery, vulnerability scanning, compliance benchmarking, patch advisory, and central management of database configuration to detect and prevent configuration drift or unauthorized configuration changes

6.2 Product Mapping

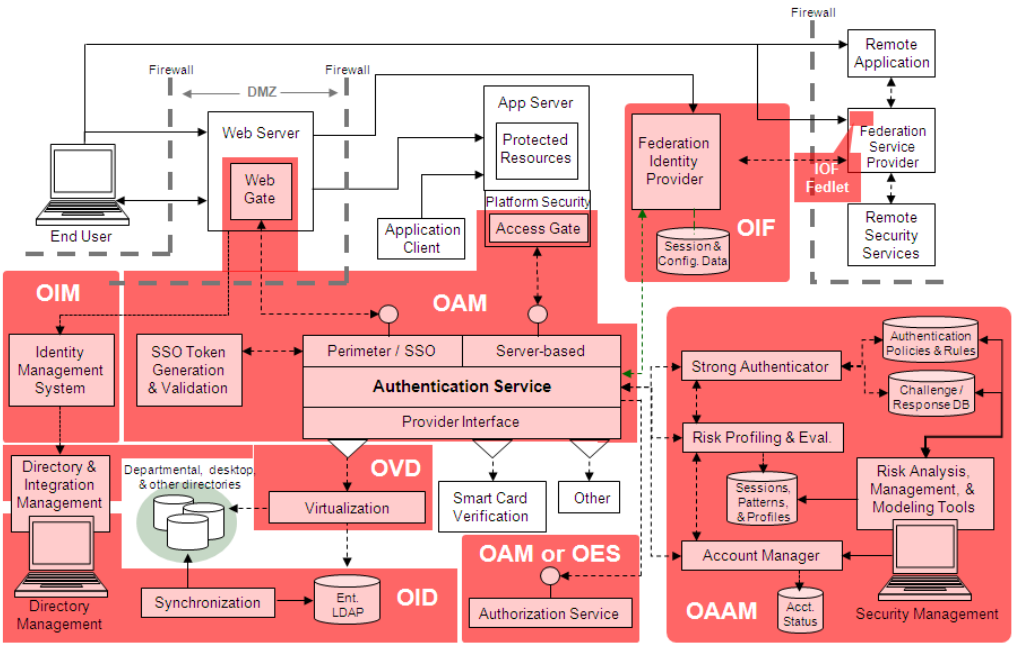
The following sections illustrate the mapping of Oracle products based on the scenarios presented in the Logical View. Some products have multiple "significant" capabilities and therefore may appear in more than one scenario. Some product capabilities may not be shown due to space considerations.

Note that the diagrams below are meant to depict a "generalized" security architecture. The components and interactions are logical representations; not actual product architecture diagrams. Please consult Oracle product documentation for a complete list of features, architecture documentation,product usage, and integration capabilities.

6.2.1 Authentication, Identity Federation, and Risk Management

Figure 6–1 depicts Oracle products as they relate to components of the authentication service, identity federation, and fraud detection / risk management.

Figure 6–1 Authentication, Identity Federation, and Risk Management Product Mapping



The following Oracle products play a significant role in this scenario:

- **OAM** provides the authentication service which includes perimeter-based authentication, SSO, and server-based authentication. It includes a provider interface that supports multiple authentication providers. OAM also provides coarse-grained authorization capabilities.
- **OVD** provides the virtualization of multiple identity stores into a single aggregate view with a standard LDAP interface. This is an optional component in the architecture, shown here to illustrate a use case with multiple identity stores.
- **OID** is an identity store that complies with LDAP specifications. It uses Oracle Database (ODB) as the physical persistence mechanism. ODB provides enterprise class performance, security, and availability to OID. OID supports two unique database security features: Oracle Database Vault (enforcing separation of duties for database administrators) and Oracle Transparent Encryption (automatically encrypts data when it is written to disk and decrypts it when it is read back to the

authorized user). OID (and OVD) include Oracle Directory Services Manager (ODSM), a web-based administration user interface for server configuration.

- **OIF** is an identity federation solution enabling browser-based, cross-domain single sign-on using industry standards such as SAML, Liberty ID-FF, WS-Federation, and Microsoft Windows CardSpace. OIF integrates with OAM to establish a local identity and it supports account linking between the identity provider and service provider in order to map users between domains. OIF can also act as the federation service provider. As a service provider it integrates with authentication and authorization systems in order to manage access to services.

  The OIF Fedlet is a small web archive designed to be embedded into a service provider's Java EE application. It enables the OIF-based identity provider to establish a federation without the need to have OIF installed as the remote service provider.

- **OAAM** provides real-time fraud detection and prevention. It also includes components to support multi-factor authentication and strong authentication, including a client interface that allows password entry in a way that circumvents keyloggers and similar viruses.

- **OIM** is the system used to manage identity information, (see Section 6.2.4). OIM includes a self-service interface that allows users to change their passwords as needed.

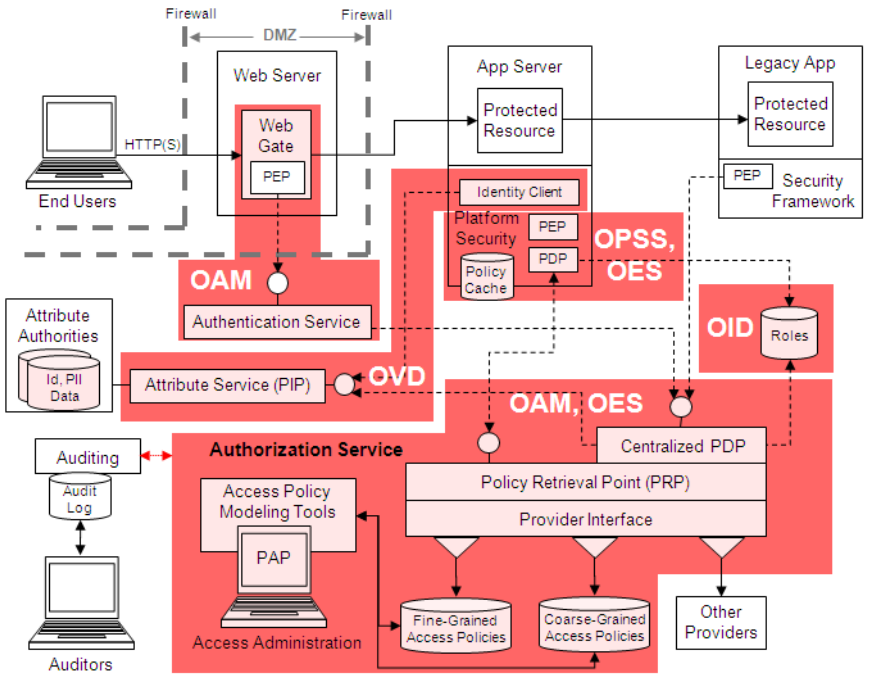
- **OES** is a fine-grained access control solution, described in more detail in the next section. It integrates with OAM to authorize user requests following successful authentication.

- **All Products** offer administrative and auditing capabilities.

6.2.2 Authorization and Access Policy Administration

Figure 6–2 depicts Oracle products as they relate to the components that combine to make up the authorization service, and some of the interactions commonly associated with it.

Figure 6–2 Authorization and Access Policy Administration Product Mapping



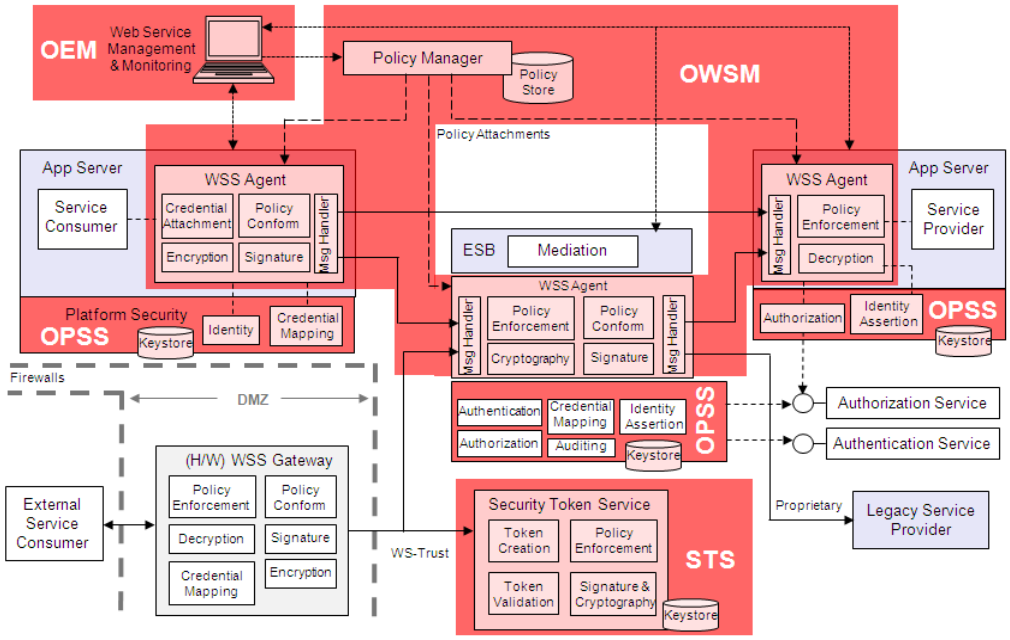
The following Oracle products play a significant role in this scenario:

- **OAM** provides the authentication service, as described in the previous scenario. OAM also performs coarse-grained authorization to resources based on user roles and access policies. In addition, OAM integrates with OES to provide fine-grained authorization and entitlements.
- **OES** is a fine-grained authorization engine that simplifies the management of complex entitlement policies. The authorization engine includes both local and centralized PDPs. OES integrates with OPSS (and other security platforms) to enable the use of local PEPs and PDPs. Policy administration is centralized, providing a broad perspective of access privileges, yet delegated, enabling multiple stakeholders to maintain the policies that affect them.
- **OID** is the directory where roles are stored. It may also be the backing store for attribute authorities.
- **OVD,** along with the OVD Provider library for ArisID, and the ArisID library, comprise a complete set of libraries that can be used to implement an IGF-compliant attribute service.
- **OPSS** is a standards-based Java framework of plug-in security services and APIs. It provides the platform security for Oracle WebLogic Server. OPSS includes authorization providers and supports the OES fine-grained authorization provider.
- **Both OAM & OES** provide the ability to author and manage access policies, and audit policy changes and authorization decisions. Policies can also be authored using Oracle's Authorization Policy Manager tool and administered with Enterprise Manager.

6.2.3 Web Service Security

Figure 6–3 illustrates how OWSM maps to the components that provide Web Service Security.

Figure 6–3 Web Service Security Product Mapping



The following Oracle products play a significant role in this scenario:

- **OWSM** is a run-time framework for security policy creation, management, and governance. Policies are created, attached to services, and enforced at various points in the messaging life cycle. OWSM includes a policy manager and Web Service security agents. Both the policy manager and agents run on Oracle WebLogic Server (OWLS).

  Agents can be on the service requester side (client) and/or the service provider side. Agents are installed in the OWLS Web Service interceptors. A request made to a Web Service is intercepted by an OWSM agent that enforces security policies defined in the OWSM policy manager.

  Since OWSM and Oracle Service Bus (OSB) both run on Oracle WebLogic Server, OWSM agents can be used to secure OSB proxy and business services. This provides a common, universal policy and enforcement model for WSS. OWSM is also integrated with Oracle JDeveloper to provide declarative policy attachment at development time.

- **OEM** Fusion Middleware Control can be used to manage Web Service policies, enforce policies at run-time, and monitor the performance of Web Services.

- **OPSS** works in conjunction with OWSM and the WebLogic container. It provides the plug-in security framework (described in 6.2.6). OPSS enables OWSM to perform credential mapping and identity assertion, which is necessary in order to propagate and assert identity from client to service. It also handles authentication and authorization of service requests as needed.

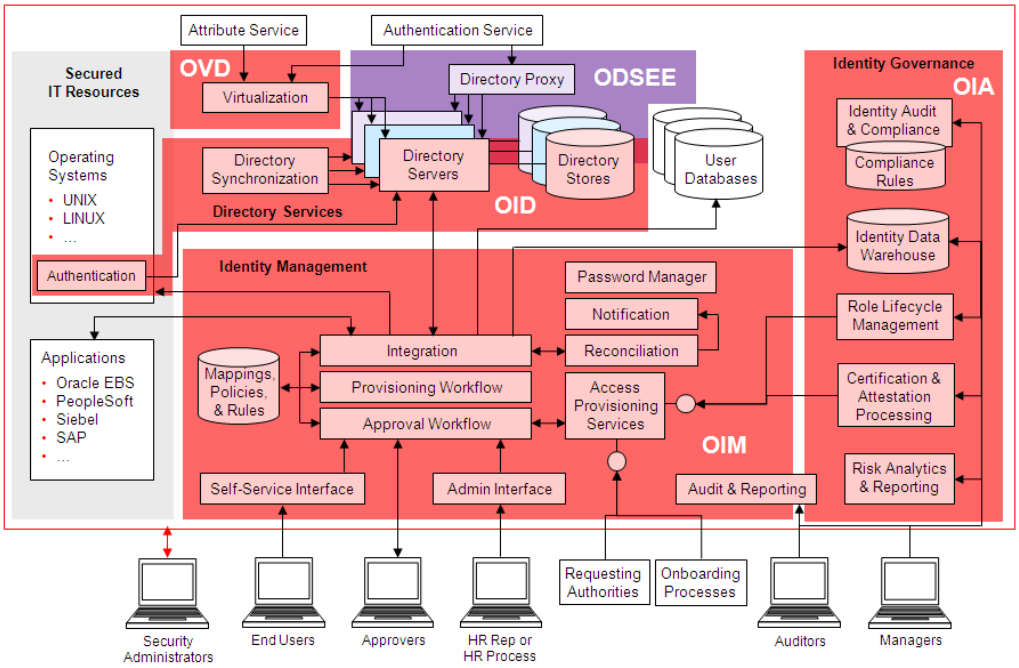
- **STS** implements the Web Services Trust language (WS-Trust) to broker a trust relationship between two parties in a Web Service exchange. STS supports many

common token formats including UserName, X.509, SAML (1.1 and 2.0), Kerberos, and Oracle OpenSSO.

6.2.4 Identity Management & Governance

Figure 6–4 depicts Oracle products that relate to components for identity management and governance.

Figure 6–4 Identity Management & Governance Product Mapping



The following Oracle products play a significant role in this scenario:

- **OIM** is a comprehensive identity management system that includes provisioning, approval workflow, integration with identity stores and applications, reconciliation, notification, password policy management, reporting, auditing, and delegated administration. It integrates with many different brands of directory servers, including Oracle's OID and ODSEE, as well as various operating systems and popular packaged applications.
- **OIA** provides the analytics, certification, attestation, compliance, and identity data warehousing capabilities required for proper identity governance. OIA integrates with OIM in order to initiate changes that need to be made as a result of identity governance, and capture changes being made to the security environment.
- **OID**, as previously mentioned, is an identity store that complies with LDAP specifications. It provides the ability to synchronize with other identity stores. In addition it includes Pluggable Authentication Modules (PAM) for most UNIX and LINUX distributions to support authentication services for operating systems.
- **ODSEE** provides a high performance directory server, directory administration, integration with Microsoft Windows, and a directory proxy server. ODSEE can be deployed in various distributed directory environments.
- **OVD** provides a virtual view of identity by aggregating information from multiple identity stores. OVD offers a virtual LDAP interface designed for authentication and attribute services.

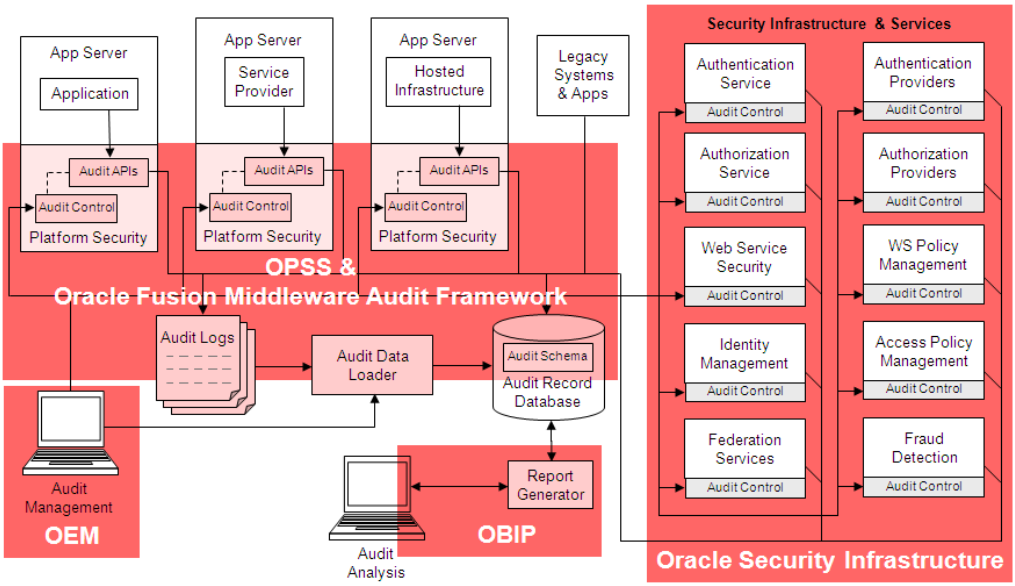
- **All Products** provide administrative capabilities and auditing of administrative functions.

Oracle also provides application-level access control governance to model and enforce segregation of duties using Oracle Application Access Controls Governor (not pictured).

6.2.5 Audit Framework

Figure 6–5 illustrates how Oracle products map to the audit framework. Oracle security infrastructure products are described in previous sections of this chapter and therefore are not called out in this section. They act as generators of audit events along with other forms of infrastructure, systems, applications, and SOA Services.

Figure 6–5 Audit Framework Product Mapping



The following Oracle products play a significant role in this scenario:

- Oracle Fusion Middleware Audit Framework is a centralized audit framework for the middleware family of products. The framework supports auditing of events handled by OPSS, Oracle Web Services, Oracle's JavaEE applications, and managed system components. The framework includes a database schema, data loading, event definitions, and other building blocks of the audit system.
- **OPSS**, described in Section 6.2.6, provides the platform security functions to support auditing. These include APIs for hosted applications to report audit events, and audit control for remote audit management.
- **OEM** is used to manage auditing configurations across the middleware environment.
- **Oracle Business Intelligence Publisher (OBIP)** can be added to the framework to support reporting. OBIP includes a large number of pre-defined reports designed specifically for the audit database.

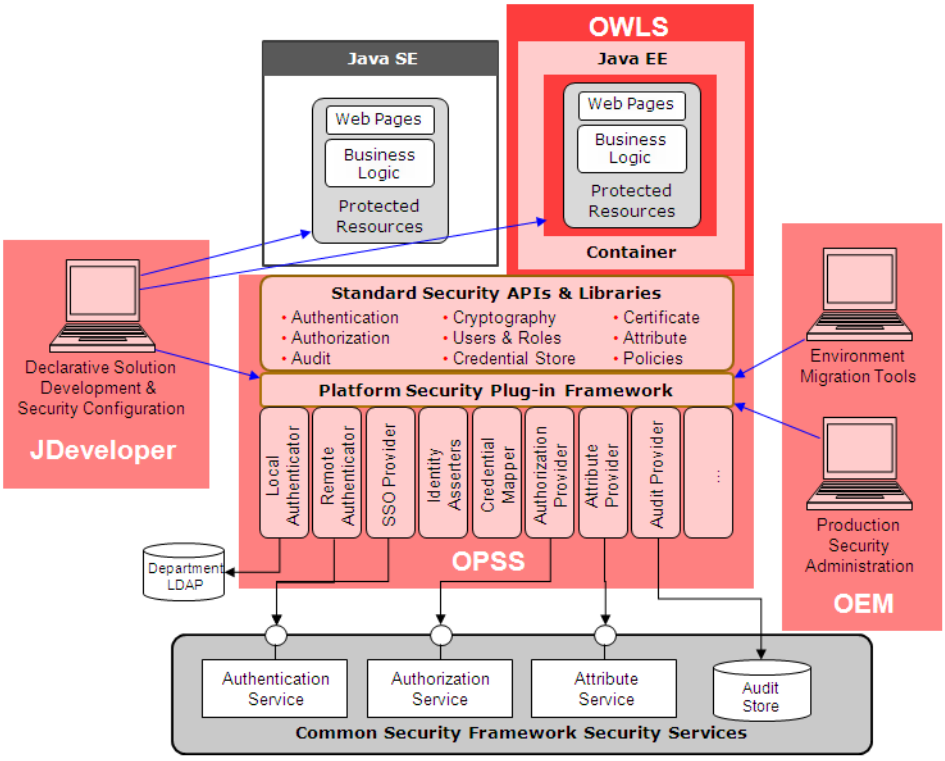
6.2.6 Platform Security Framework

Oracle WebLogic Server (OWLS) is the de facto application server for Oracle Fusion Middleware 11g. With the release of Oracle Fusion Middleware 11gR1, Oracle combined BEA's internal security framework used in products like WebLogic Server and Oracle Entitlements Server (OES), with Oracle Fusion Middleware's security platform, Java Platform Security (JPS), into one complete security framework known as Oracle Platform Security Services (OPSS).

OPSS builds upon Java SE and Java EE security and provides an abstraction layer in the form of standard Java APIs, based on the Java Community Process, that insulate developers from security and identity management implementation details.

As shown in Figure 6–6, OPSS provides a layered architecture. Java applications use APIs to access security features and functionality. These APIs abstract the details of underlying identity management and security systems, allowing the management of security providers and security information to be handled separately from the applications that use it.

Figure 6–6 Oracle Platform Security Services



OPSS's security provider architecture allows it to support different security and identity systems without changing the APIs provided to applications. OPSS Also provides a number of important features such as:

- Authentication providers including the WebLogic Default Authenticator, external LDAP stores, and database systems to host data for enterprise applications
- WebLogic Identity Assertion providers that support certificate authentication using X.509 certificates, SPNEGO tokens, SAML assertions, and CORBA Common Secure Interoperability version 2 (CSIv2) identity assertion
- A Java policy provider that supports code-based and subject-based authorization

- Support for role-based access control (RBAC), Java Authorization and Authentication Services (JAAS), and Java Authorization Contract for Containers (JACC)
- Java EE container security in permission-based (JACC) mode and in resource-based (non-JACC) mode
- Support for application roles (logical roles specific to an application), role hierarchies, and the mapping of application roles to enterprise groups (enterprise roles) maintained at the security domain level
- User and Role API framework, which allows applications to access identity information (users and roles) in a uniform and portable manner regardless of the particular underlying identity repository
- Oracle Security Developer Tools (OSDT) which includes a set of Java-based cryptographic libraries supporting XML signature, XML encryption, XML Key Management Specification (XKMS), SAML, WS-Security, and other non-XML standards such as Secure / Multipurpose Internet Mail Extensions (S/MIME) and Online Certificate Status Protocol (OCSP)
- A Credential Store Framework (CSF), a set of APIs that applications can use to create, read, update, and manage credentials securely
- A common audit framework for Oracle Fusion Middleware products
- Integration with **Oracle JDeveloper**, which allows application designers to model security into the application
- Integration with **OEM** to support security administration

> **Note:** The actual list of providers and features may vary depending on product version.

In addition to OPSS, Oracle provides the following platform security services:

- Oracle's Authorization API framework, a public project known as "OpenAZ", which provides a XACML-based interface for pluggable PEPs and PDPs. The Java XACML Authorization API draft was submitted to the OASIS XACML Technical Committee in July 2009 (http://lists.oasis-open.org/archives/xacml/200907/msg00020.html).
- Identity Governance Framework (IGF), which includes the ArisID library with Identity Beans, designed to provide a single API for developers to access a variety of identity information following the IGF specification.
- Oracle Web Services Manager (OWSM) policy manager and Web Service Security interceptors, further described in Section 6.2.3.

6.2.6.1 The Application Lifecycle

OPSS provides support for all the phases of an application's lifecycle. OPSS is integrated with Oracle JDeveloper, which allows an application designer to model security into the application through a set of wizards. Oracle JDeveloper also provides an authorization editor that allows developers to create authorization policies for ADF taskflows and pages without writing any code.

Typically a developer deploys an application to a WebLogic Server domain embedded in JDeveloper. The developer can then deploy the application to a remote WebLogic Server domain using Oracle Enterprise Manager Fusion Middleware Control

(FMWControl). OPSS is integrated with FMWControl to allow application security policies and credentials migration to be configured during application deployment. Post deployment, an administrator uses FMWControl, or WebLogic Scripting Tool (WLST), to manage the application's security policies, e.g., edit authorization policies, or change audit policies. OPSS also provides a policy management API allowing programmatic control over authorization policies. All such changes are transparent to the application and do not require any application code change.

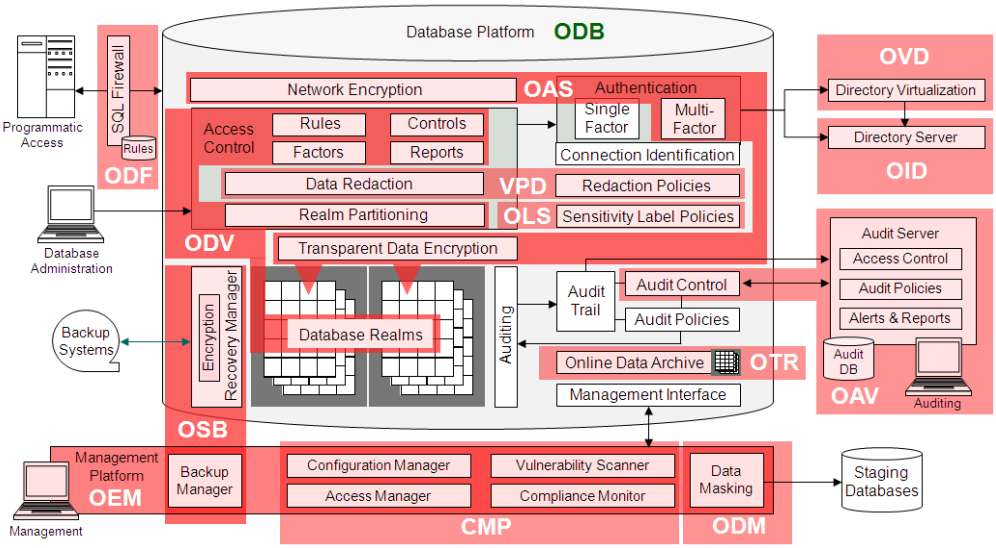
An application normally goes from development through test and staging environments before being put into production. OPSS supports this model by providing migration tools that move security policies from one domain to another.

From solution design, through testing stages, production deployment, and run-time operations management, OPSS supports application security. It interfaces with the tools used by designers, developers, security architects and administrators at each stage of the application lifecycle and provides a secure computing platform for business solutions.

6.2.7 Database Security

Oracle Database delivers industry leading performance, scalability, security and reliability on a choice of platforms. It provides comprehensive features to easily manage the most demanding transaction processing, business intelligence, and content management applications. Oracle offers a wide range of options to extend Oracle Database to meet specific security and regulatory requirements. Figure 6–7 illustrates the mapping of database security components to product options.

Figure 6–7 Oracle Database Security Options



Advanced security options for Oracle Database (ODB) include:

- **OAS** - provides industry standards-based data privacy, integrity, authentication, single sign-on, and access authorization in a variety of ways. OAS supports RC4, DES, 3DES, and AES based network encryption, and MD5 or SHA-1 hashing algorithms for data integrity. Authentication mechanisms include Kerberos, RADIUS (smart cards and token cards), SSL, and Entrust/PKI. OAS can also transparently encrypt persisted data on a per-column or per-tablespace basis.

- **ODF** - provides the SQL firewall capability to protect the database from inappropriate SQL commands.
- **ODV** - restricts access to specific areas in an Oracle database from any user, including users who have administrative access. ODV allow you to create realms composed of the database schemas or database objects that you want to secure. You can further secure the realm by creating rules, factors, identities, rule sets, and secure application roles. In addition, you can run reports on the activities these components monitor and protect.
- **OLS** - enables row-level access control. It controls access to the contents of a row by comparing that row's label with a user's label and privileges. Administrators can add selective row-restrictive policies to existing databases by means of the graphical interface provided by Oracle Enterprise Manager (OEM) Database Control.
- **VPD** - a feature of Oracle Database, allows security administrators to assign unique access policies to tables. Access policies can specify custom rules for specific rows and columns of a table. Data is redacted based on the evaluation of policies and run-time parameters.
- **ODM** - allows organizations to generate realistic and fully functional data with similar characteristics as the original data to replace sensitive or confidential information. ODM applies masking definitions to columns in a staging database. The results can be exported and used for development and testing functions.
- **OTR** - uses a Flashback Data Archive (FDA) for tracking historical data. An FDA is a logical container for managing historical information for specified tables. The FDA creates an internal history table for every tracked table. This makes it possible to automatically and transparently track all of the changes to any set of tables in an Oracle 11g database, and to easily query data in those tables as of any point in time or over any interval within the specified retention period, with minimal performance impact.
- **OSB** - provides centralized tape backup management for heterogeneous file systems and the Oracle database. OSB environments are centrally managed using a single console and common management interface across spectrum of servers and NAS devices. OSB delivers policy-based backup encryption supporting both host-based and LTO-4 tape drive encryption options.
- **OEM CMP** - simplifies the management of IT infrastructure by providing managed baseline configurations and configuration policies. CMP enforces compliance with real-time change detection for files, database objects, users and processes, and detects unauthorized and authorized configuration changes. It also alerts users to critical patches issued by Oracle and immediately identifies those systems across the enterprise that may require the new critical patch. CMP automatically assess all targets for violations, reports deviations, and offers remediation paths for fixing vulnerabilities.
- **OAV** - automates the consolidation of audit data into a secure repository which includes audit policies, access control, audit alerts, and reports. OAV provides centralized audit policy management.
- **OID and OVD** - (detailed in previous scenarios). ODB's Enterprise User Security (EUS) feature supports the use of externally managed identities via LDAP or other sources accessed via OVD. This allows access to all Oracle database instances across the organization to be managed using the same identity management infrastructure as applications and other resources.

7

Deployment View

This section provides an example of how the products mapped in the previous chapter might be deployed to physical hardware. A network topology format is used to illustrate where products are most likely to be deployed in terms of network tiers.

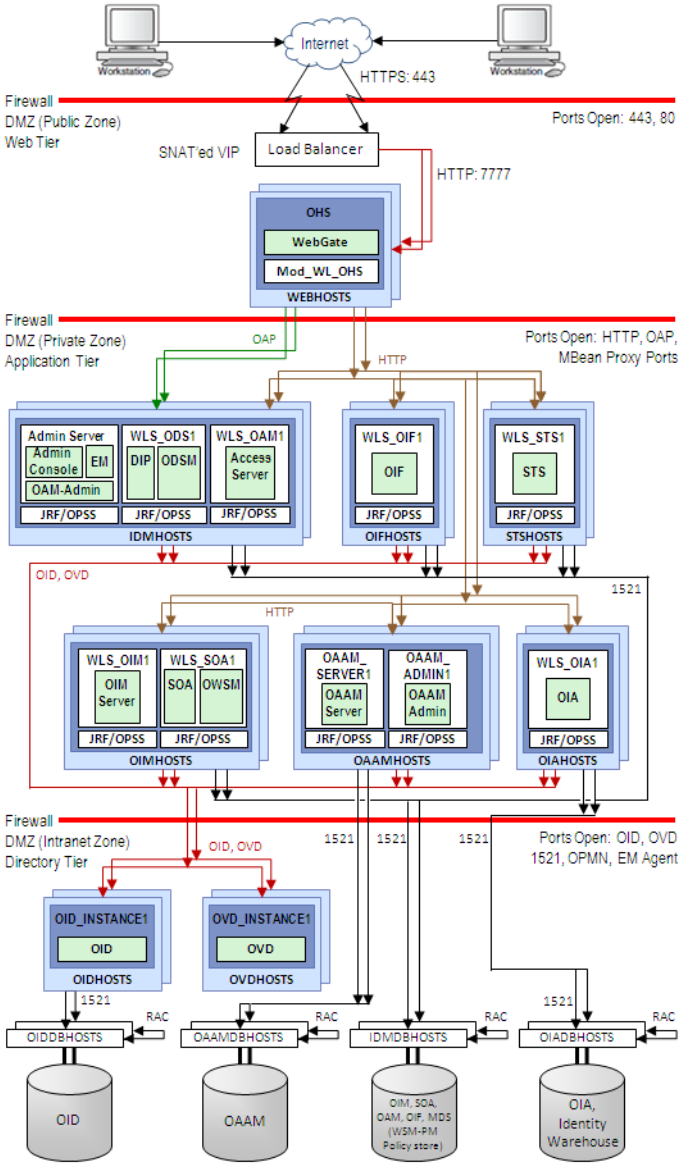
A number of factors influence the way products are deployed in an enterprise. For instance, load and high availability requirements will influence decisions about the number of physical machines to use for each product. Federation and disaster recovery concerns will influence the number of deployments and failover strategy to use. In addition, deployment configurations and options may vary depending on product versions. Given these and other variables it is not feasible to provide a single, definitive deployment for the products, or to guarantee the deployment scenarios presented in this chapter will work for a particular version. The scenarios presented here are meant to be used as an example. Please consult product documentation for further deployment information.

The Deployment View is divided into three diagrams. Figure 7–1 contains products pertaining to identity management. Figure 7–2 contains entitlements and Web Services management. Figure 7–3 illustrates how Oracle Database Security products and options may be deployed. Additional information on deployment can be found in the Oracle Fusion Middleware Enterprise Deployment Guide for Oracle Identity Management document.

7.1 OAM, OIM, OIF, OIA, STS, OID, and OVD Sample Deployment

Figure 7–1depicts a sample deployment for OAM, OIM, OIF, OIA, STS, OID, and OVD products.

Figure 7–1 Deployment View, part 1



7.1.1 Web Tier

Figure 7–1 defines four computing zones separated by firewalls. The top-most zone is the public network, e.g. Internet. Internet traffic is permitted to pass through the public zone firewall using ports 80 (HTTP) and 443 (HTTPS). Traffic is load balanced to two web servers (WEBHOSTs) with secure network address translation in place to mask internal IP addresses.

The WEBHOSTs have Oracle HTTP Server, WebGate (an Oracle Access Manager component), and the mod_wl_ohs plug-in module installed. The mod_wl_ ohs plug-in module allows requests to be proxied from Oracle HTTP Server to a WebLogic Server running in the application tier.

WebGate in Oracle HTTP Server uses Oracle Access Protocol (OAP) to communicate with Oracle Access Manager running on the IDMHOSTs in the application tier.

WebGate and Oracle Access Manager are used to perform operations such as user authentication.

On the firewall protecting the web tier, only the HTTP ports are open (443 for HTTPS and 80 for HTTP). Oracle HTTP Server proxies requests for OIF, STS, and OIA.

7.1.1.1 Architecture Notes

- Communication from external clients does not go beyond the Load Balancing Router level.
- Direct communication between two firewalls at any one time is prohibited, e.g. communication that begins in one firewall zone must end in the next firewall zone.
- All communication between components across DMZs is restricted by port and protocol, according to firewall rules.
- Oracle HTTP Servers on the WEBHOSTs are configured with mod_wl_ohs, and proxy requests for the Oracle Enterprise Manager (EM), Oracle Directory Integration Platform (DIP), and Oracle Directory Services Manager (ODSM) J2EE applications deployed in WebLogic Server on the IDMHOSTs

7.1.2 Application Tier

The application tier is the tier where J2EE applications are deployed. Products such as Oracle Directory Integration Platform, Oracle Identity Federation, Oracle Identity Analytics, Oracle Adaptive Access Manager, Oracle Directory Services Manager, and Oracle Enterprise Manager Fusion Middleware Control are the key J2EE security-related components that can be deployed in this tier. Applications in this tier benefit from the high availability support of Oracle WebLogic Server.

The Identity Management applications in the application tier interact with the directory tier for reasons including:

- In some cases, they leverage OID and OVD for authentication and authorization operations.
- In some cases, they leverage the directory tier for enterprise identity information.
- In some cases, they leverage the directory tier (and sometimes the database in the data tier) for application metadata.
- Oracle Enterprise Manager Fusion Middleware Control and Oracle Directory Services Manager are administration tools that provide administrative functions to the components in the application tier as well as the directory tier.

In the application tier the IDMHOSTs have the WebLogic Server with the Administration Console, Oracle Enterprise Manager Fusion Middleware Control, Oracle Directory Integration Platform, Oracle Directory Services Manager, and Oracle Access Server installed. Both IDMHOSTs run the WebLogic Server Administration Servers and Managed Servers. Note that the administration server should be configured to be active-passive, that is, although it is installed on both nodes, only one instance is active at any time. If the active instance goes down, then the passive instance starts up and becomes the active instance. Oracle Access Server communicates with OVD in the directory tier to verify user information.

On the firewall protecting the application tier, the HTTP ports and OAP port are open. The OAP (Oracle Access Protocol) port is for the WebGate module running in Oracle HTTP Server in the web tier to communicate with Oracle Access Manager to perform operations such as user authentication.

The OIMHOSTs have OIM and Oracle SOA installed. OIM is the user provisioning application. Oracle SOA, deployed here, is used exclusively for providing workflow functionality for OIM.

The OIFHOSTs have the WebLogic Server with Oracle Identity Federation (OIF) installed. The STSHOSTs have the WebLogic Server with Secure Token Service (STS) installed. The OIAHOSTs have the WebLogic Server with Oracle Identity Analytics (OIA) installed. The OAAMHOSTs have the WebLogic Server with Oracle Adaptive Access Manager (OAAM) and Console installed.

7.1.2.1 Architecture Notes

- All applications running on Oracle WebLogic Server that require high availability should be clustered across multiple physical machines. Clustering in this diagram is implied by the use of multiple host icons.
- Oracle Enterprise Manager Fusion Middleware Control is integrated with OAM using the Oracle Platform Security Service (OPSS) agent.
- Additional application-tier load balancers (not shown) can be deployed to accommodate load balancing across clustered services.

7.1.3 Directory Tier

The directory tier is in the Intranet Zone. The directory tier is the deployment tier where all the LDAP services reside. This tier includes products such as Oracle Internet Directory (OID) and Oracle Virtual Directory(OVD). The directory tier also includes databases and metadata storage for security applications deployed in the application tier.

The directory tier is managed by directory administrators providing enterprise LDAP service support. In some cases, the directory tier and data tier may be managed by the same group of administrators. In many enterprises, however, database administrators own the data tier while directory administrators own the directory tier. The data and directory tiers have been combined in this document.

Typically protected by firewalls, applications above the directory tier access LDAP services through a designated LDAP host port. The standard LDAP port is 389 for the non-SSL port and 636 for the SSL port. LDAP services are often used for white pages lookup by clients such as email clients in the intranet.

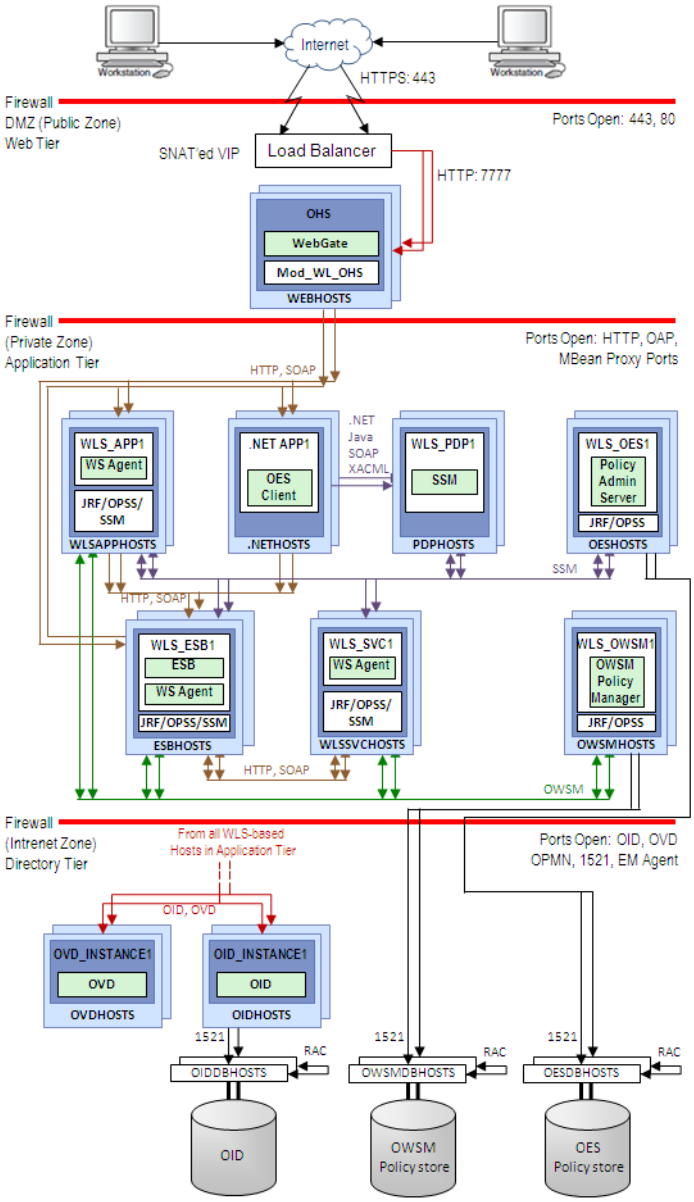
7.1.3.1 Architecture Notes

- Oracle Internet Directory relies on RDBMS as its backend data store.
- Oracle Virtual Directory provides virtualization support for other LDAP services or databases or both.

7.2 OWSM and OES Sample Deployment

Figure 7–2 depicts a sample deployment OWSM and OES products. It could be combined with the first diagram since it uses the same tiers. It is shown here separately in order to improve clarity and reduce complexity.

Figure 7–2 Deployment View, part 2



7.2.1 Web Tier

The web tier in this diagram matches that of Figure 7–1. OAP invocations to OAM have been omitted. Instead, HTTP requests to servers that host applications have been included.

7.2.2 Application Tier

The application tier now includes applications hosted by WLSAPPHOSTs and .NETHOSTs. WLSAPPHOSTs host an instance of WebLogic Server, which includes an embedded OWSM Web Services agent. The JRF/OPSS security framework includes an embedded OES Security Service Module (SSM). WLS_APP1 invokes Web Services hosted by WLS_SVC1 via the Oracle Service Bus running on the ESBHOSTs. OWSM agents, deployed on all servers that provide, consume, and mediate Web Services perform and validate Web Service security. WLS_APP1, WLS_ESB1, and WLS_SVC1

can perform entitlements validation and enforcement locally via their embedded SSMs.

.NETHOSTs are included to illustrate and example of using a centralized PDP to handle entitlements. The security service module hosted on the PDPHOSTs acts as the policy decision point for types of applications that are not yet supported by embedded SSMs. Centralized SSMs provide a number of APIs including .NET, SOAP, Java, and XACML.

WLS_OES acts as the policy administration point for access policies that define OES entitlements. Policies are pushed to all SSMs including embedded and centralized SSM.

WLS_OWSM runs the administrative processes associated with OWSM. The Policy Manager interacts with the OWSM agents and gateways via SOAP over HTTP.

7.2.2.1 Architecture Notes

- All applications running on Oracle WebLogic Server that require high availability, such as WLS_ESB1, WLS_APP1, and WLS_SVC1 should be clustered across multiple physical machines. Clustering is implied by the use of multiple host icons.
- Additional application-tier load balancers (not shown) can be deployed to accommodate load balancing across clustered services.

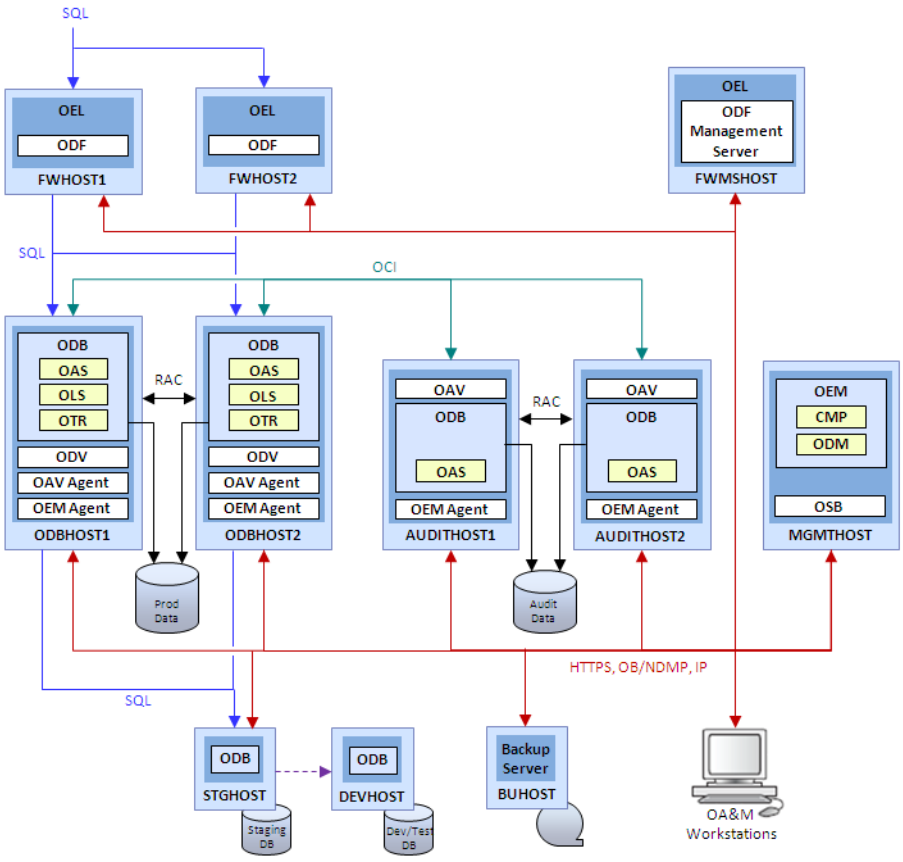
7.2.3 Directory Tier

The directory tier for OES and OWSM includes a database to store entitlements policies and Web Service security policies. This is accessed using standard JDBC connections over port 1521. OID and OVD are also included. These components are used by all WebLogic Server instances to support OPSS security services. They are also used by OES in the creation of entitlement policies.

7.3 Database Security Sample Deployment

Figure 7–3 shows a sample deployment of Oracle Database and related data security products.

Figure 7–3 Database Security Deployment



FWHOST1 and FWHOST2 are installed with Oracle Enterprise Linux (OEL) and Oracle Database Firewall (ODF). The ODF instances act as a firewall for incoming SQL traffic. Each instance can handle multiple downstream databases, and the instances are configured for high availability. SQL traffic must pass through the firewall boxes in order to reach the databases. FWMSHOST, also with the OEL operating system, contains the ODF Management Server.

ODBHOST1 and ODBHOST2 represent an Oracle Database Real Applications Cluster (RAC) deployed across two physical machines. The cluster is fault tolerant in that the database will still be available in the event of a single machine failure. Network communication between cluster nodes is denoted by the RAC connections.

The Oracle Database servers have Oracle Advanced Security (OAS), Oracle Label Security (OLS), and Oracle Total Recall (OTR) security options installed. In addition, Oracle Database Vault (ODV) is installed on the machine as well as auditing and management agents.

AUDITHOST1 and AUDITHOST2 represent machines where Oracle Audit Vault (OAV) is installed. OAV consists of a database to store audit records, an administration server and console, and agents that collect audit records from source systems. For scalability and availability, the OAV server can implement Oracle Database RAC. OAV may also use OAS database security features.

As a best practice, the OAV collection agents should be installed on each host system to be audited. The collection agent communicates with the Audit Vault server to receive some configuration information and to send audit data for storage. This communication channel is based on the Oracle Call Interface (OCI). Immediately

following installation, password-based authentication is used to secure this channel. Administrators can further secure this channel after installation by using the TCPS protocol to encrypt data.

MGMTHOST is a simple standalone installation of Oracle Enterprise Manager (OEM) along with the Configuration Management Pack (CMP) for database configuration, and Oracle Data Masking (ODM).

ODM operates on a copy of production data that is installed on a staging database (STGHOST). Once data masking has been completed, data can be copied to development and test systems, (DEVHOST), using a variety of copy methods (represented by the dashed line).

OEM also communicates with managed nodes via agents. Also installed on MGMTHOST is Oracle Secure Backup (OSB). OSB communicates with the database resource manager using OB/NDMP protocol. Backup data is encrypted at the production database server and then transferred to the backup server (BUHOST).

There are many ways that Oracle Database, Oracle Database Security, Oracle Audit Vault, and Oracle Enterprise Manager can be installed and configured. Please consult product documentation for detailed information on supported platforms and configuration options for specific products and versions.

8

Summary

Security is an important aspect of modern computing, particularly in highly distributed IT environments. Information is constantly being transmitted, processed, stored, backed up, moved, and analyzed. Web enablement and Web Services, both key trends to advance efficiency, agility, and connectedness, further extend corporate information out over public networks to remote work locations, customers, and partners. As the distribution and reach of IT systems and information increases, security must continually evolve to meet the needs and challenges this presents.

Not only is there a need to protect information and computing assets from malicious hacking, but there are also regulatory concerns to deal with. IT must be secure, and companies must be able to prove precisely what security exists. Therefore they must know the state of their IT security at all times.

The best way to meet the needs and challenges of IT security is to implement an architecture that is comprehensive, consistent, extensible, adaptable, standards-based, and easily managed. The architecture must provide visibility into the current state of configuration, as well as all changes that have been made over time. It must also be capable of detecting potential threats, presenting threat activity for evaluation, and notification of threats in real time.

ORA Security describes an architecture that is designed to meet these criteria using an extensible security framework and common security services. It presents architecture principles and advocates the use of components and standards that not only provide security in a consistent and extensible manner, but also make effective auditing, compliance, and attestation possible.

Oracle Identity Management and Oracle Database Security products can be used to implement any or all of the components outlined in the reference architecture. Oracle Identity Management is a comprehensive suite of integrated products that are best-in-class. These products enable customers to meet compliance efficiently, secure their critical applications and sensitive data, and lower operational costs. Oracle Database Security adds advanced security capabilities to the state-of-the-art Oracle Database. It helps organizations comply with privacy and audit regulations and reduce risks associated with insider threats.

A

Further Reading

ORA Security introduced a number of Oracle products that can be used independently or combined to implement a security architecture. A great deal of information is available on these products to describe their features, capabilities, benefits, architecture, and deployment options.

Please consult the Oracle Identity Management and Oracle Database Security web sites for the latest Oracle security product information. These sites contains links to product downloads as well as valuable technical information.

Identity Management product documentation can be found on the Oracle Documentation Library web site. Oracle Database product documentation can be found on the Oracle Database Documentation Library web site.

Also, check out the Oracle Identity Management Resource Library for technical white papers, analyst reports, seminars, webcasts, etc. In particular, the following technical information may be of interest:

- Oracle Access Manager, An Oracle White Paper
- Oracle Adaptive Access Manager, An Oracle White Paper
- Oracle Identity Manager Architecture, An Oracle White Paper
- Oracle Identity Analytics, An Oracle White Paper
- Oracle Identity Federation Technical White Paper
- Oracle Internet Directory Technical White Paper
- Oracle Virtual Directory Technical White Paper
- Oracle Directory Server Enterprise Edition documentation center
- Securing Service-Oriented Architectures (SOA) with Oracle Web Services Manager, An Oracle White Paper
- Oracle Entitlements Server, An Oracle White Paper
- Oracle Platform Security Services information web site
- Oracle Database Vault with Oracle Database 11g Release 2, An Oracle White Paper
- Oracle Audit Vault, An Oracle White Paper
- Oracle Total Recall with Oracle Database 11g Release 2, An Oracle White Paper
- Oracle Advanced Security with Oracle Database 11g Release 2, An Oracle White Paper
- Oracle Label Security with Oracle Database 11g Release 2, An Oracle White Paper
- Oracle Database Firewall web site

- Oracle Secure Backup Documentation
- Oracle Enterprise Manager 11g Configuration and Change Management