Name: David R. Shelton
Submission Date: August 25, 2003
Version of Assignment: GSEC Practical Assignment Version 1.4b, Option 1
Title:
Securing the Enterprise with OpenNetwork DirectorySmart: An Implementation Review

Abstract

This paper outlines the introduction of OpenNetwork's DirectorySmart into the enterprise infrastructure and application environment of a financial services Company. Though typically labeled as a web access control product, DirectorySmart provides a much broader framework to improve both the security of applications and productivity of its users. Historically, limited short-term methods have partially addressed the need for web- and client-server based user authentication and authorization. After analyzing the security risks and business benefits, the Company's strategic decision to purchase and implement DirectorySmart confirmed the need for a secure, centralized authentication and authorization management framework. This decision included two candidate applications for DirectorySmart integration: AMISC and PeopleSoft Time & Labor.

The Java-based AMISC application provided a good first "Proof of Concept" for DirectorySmart, since it served a small internal user base, was web-driven, and did not already contain extensive authentication and authorization code. DirectorySmart's role-based access control architecture fit well with the application, providing both URL-based and fine-grained access control features. From an implementation perspective, some changes in the DirectorySmart infrastructure layout were required; however, the integration proved successful. From this exercise, a number of security and business benefits were derived from improved access and password management, authentication, logging, and web server access, resulting in reduced development and administrative costs and increased user productivity. Going forward, the use of DirectorySmart is expected to expand within the Company to secure a variety of internal and external enterprise applications.

The History Lesson

Like many organizations, the Company's enterprise application development environments had expanded along with its business needs. Four years ago, client-server applications ruled the day, and enterprise web applications had yet to be developed. During this time, there was little need for authentication or authorization of internal or external users via the Web, since there was neither protected content nor personalized information that required security.

The first system involving external web user authentication was the Wholesale site. The system allowed brokers (i.e., external customers) to log in to access the full range of services offered by the site. The authentication mechanism used a technique of hashing and storing a user's password into a Microsoft SQL*Server database when a user first visited the site. Upon his return, the user password was rehashed and the value compared with the original stored password; if the values matched, the user was allowed entry into the application. The hashing technique used Macromedia's ColdFusion Hash function, which is based on Ron Rivest's MD5 algorithm, and takes a variable-length string and converts it into a fixed 32-byte indecipherable string (Horvath). While this method was relatively simple to implement, it did not inherently have the ability to place additional security constraints—minimum character length, periodic password change, account lockout after a certain number of attempts, expiration, login history, etc. Depending on the type of user who logged in—broker, Wholesale personnel, administrator—the application would grant the appropriate level of access within the application. This "role-based" access control was developed specifically for the application.

Along a different development path, another group developed a method for authentication and application-level access for a handful of its database-driven applications. Though not specifically a web-based model, it involved encoding a user's password (using reverse hex) and storing it in a relational database. While not truly secure, it provided a convenient method for application-specific logins.

As these development efforts came together, a new updated model was developed for user registration, authentication, and authorization. Part of the process included using Lucigenic Crypt, a third-party cryptographic COM object that incorporated the Blowfish encryption algorithm that could be called to encrypt/decrypt selected information, while storage of such information would continue to be SQL*Server-based.

As additional large-scale applications continue to be added to the enterprise, each one has brought its own unique method of authentication and authorization, depending upon whether the application was internal- or external-facing, web-based or client-server, COTS (Commercial Off-The-Shelf) or internally developed.

The General Problems

By further examining elements of the existing application architecture, a number of problems became evident. In some cases, these were outright security vulnerabilities, and in others, opportunities for improvement. These were:

Lack of unified identity and access management model for applications – By not having a common framework for identity management and access to applications and services, there was no consistency in the method for creating, maintaining, securing, and terminating user identities. Without this framework, it became increasingly difficult to enforce a standard set of policies for accounts and associated access.

Additional development – From an application development perspective, additional resources were required to build the security as each application was created. Each application required some form of authentication and role-based access control. Also, additional work was needed to develop administrative screens to manage users and to track user access to a given application.

Additional administration – Administratively, as the number of applications grew, additional time and effort was required to provision (i.e., grant access), manage, monitor, and deprovision (i.e., revoke access) for each application for each user. Proper execution of these access management functions is an essential element of a security program; for a good summary of access management principles, see SANS Security Essentials with CISSP CBK Version 2.1 (Cole et al., p.392). Deprovisioning became increasingly difficult, which is discussed below.

Error-prone deprovisioning – When the number of independent authentication and authorization database stores (sometimes referred to as "silos") for applications grows, additional time and effort is required in order to ensure that when a user leaves the organization, all of his access privileges are properly removed. With separate systems, the administrator is faced with three unpleasant alternatives. The first involves tracking each user's authorized applications throughout his employment. When the user leaves, the tracking log is consulted, and the administrator goes into each of the authorized applications and removes the user's access. The second method doesn't involve any tracking; however, when a user leaves, every application is reviewed to ensure that the individual's access is revoked. The third method involves removing the user from default key system entry points all users possess (e.g., Microsoft network login, e-mail), which prevents the user from accessing other enterprise applications. With each of these methods, it should be evident that errors can and will occur, resulting in incomplete deprovisioning of users, and therefore a potential vulnerability of unauthorized access. In the case of this Company, a combination of these methods was used with the addition of periodic audits of accounts to check for inactivity (e.g., not having logged in for 60 days) and then taking appropriate action.

Separate authentication and authorization stores – Aside from the provisioning management headaches of separate authentication and authorization silos, having these reside in different locations makes it increasingly difficult to manage. Specifically, the infrastructure needs to be designed so that each of these silos is protected in a like manner. This means that they would need to reside in the same location on the network (or at least on an equivalently secured network segment). The hosts on which these silos reside would also need to be protected similarly. If these silos are accessed through a firewall, more "holes" would have to be created in order for the communication to occur between the application and its respective silo. The bottom line is that as the number of authentication and authorization stores grows, the overall security burden (and therefore risk) increases.

Homegrown web application security – As discussed in the History section, the development communities chose to develop in-house methods for authentication and authorization. While this appeared to be a simple, cost-effective approach to securing web applications, there were definite drawbacks. First, one of the authentication schemes used simple encoding of the password, which was extremely easy to defeat. Second, there was a chance that some coding error could be made that would allow inadvertent access. This could be said of a commercial web access control product as well; however, a mature product with hundreds of customers and literally millions of user accounts and no security breaches empirically suggests that this is a strong web application security approach. Third, the homegrown solution involved a single key (discussed separately below) which made the application more vulnerable.

Vulnerable symmetric key – Blowfish is proving to be an excellent multi-purpose cryptographic algorithm (Schneier), and is utilized by a number of products, including DirectorySmart. Since it utilizes a symmetric key, it is essential to protect the key used to encrypt and decrypt information. However, a strong cryptographic algorithm is not a panacea and does not guarantee a secure application (Schneier). In this case, the key resided directly on the web servers in a separate file and was referenced by the application. This meant that if the web server was compromised, it would be a simple matter to find the key and be able to decrypt the password and other sensitive information. As the web application usage grew, the same key was shared among all of the web servers.

No centralized logging or reporting – With these homegrown applications, there was neither a common requirement nor method for recording key security events, which is essential in the proper securing of an application. Events related to user access, such as login successes, login failures, password changes, access to particular application functions, etc., would not be consistently logged across applications. Further, security-administrative events, such as user creation, deletion, role creation, role mapping, etc. were also not recorded. As such, the applications tended to have only a subset of the desired features and data necessary to perform proper auditing of an application's usage, much less a reporting facility to present the data in a meaningful way.

The Business Needs

With the explosive growth of the Company, combined with the increased pace of development, it became clear that a unified approach to identity and access management was required. With this in mind, the following technical requirements to support the business needs were identified:

Enterprise directory-oriented – Create a centralized data repository of essential identity information, accessible by open standards-based methods, i.e., LDAP (Lightweight Directory Access Protocol).

Centralized authentication/Single Sign-On (SSO) – Provide a common underlying method for applications to authenticate users, providing a path towards Single Sign-On (or at least reduced sign-on), with flexibility to support stronger forms of authentication: X.509 certificates, smart cards, or tokens.

Low implementation and operational costs – Deploy the product easily and provide a straightforward development environment.

Leverage Active Directory – Utilize and extend existing Active directory infrastructure, using existing user accounts within the domain.

Java/J2EE and Microsoft support – Provide an application and infrastructure environment that supports the frameworks of Java/J2EE (Java 2 Enterprise Environment) and Microsoft-based applications.

Fully-integrated security – Support policy management for multiple applications, which can be configured and enforced correctly through an integrated set of tools.

Auditing and reporting – Provide the ability to audit and report on all transactions related to the directory, particularly for security events (logon success/failure, logoff, attribute changes, etc.).

Self-service – Allow users to perform their own basic account management, such as password changes, resets, or recoveries.

Self-registration – Allow external users to self-register into a central directory for authorized applications.

Delegated administration – Provide the facility to distribute selected user management administrative tasks to various business units, using an easy-to-use interface.

The Decision

Over the course of the previous two years, the Security group had analyzed a number of web access management solutions, including Netegrity SiteMinder, Oblix NetPoint, RSA ClearTrust, Entrust GetAccess, and OpenNetwork DirectorySmart, with a recommendation to purchase DirectorySmart. Earlier this year, the Security group of the Company led a renewed effort to organize and implement a web access control solution that would address a number of the existing problems and provide a solution to the business needs. Based on a current review of the business requirements vis-à-vis product features, DirectorySmart once again emerged as the recommended product of choice.

Among the reasons for the recommendation of DirectorySmart, its strengths included the following:

All-LDAP approach – DirectorySmart is among the few web access control products that relies solely on an LDAP directory for storage of the identity and access management information. (Originally, DirectorySmart was the first product to hold this claim; other vendors have since followed suit with this capability.)

Strong integration with Active Directory – OpenNetwork continues to enjoy a strong working relationship with Microsoft. This support has translated into a product that not only supports an Active Directory-based LDAP implementation, but also embraces it. This has resulted in the product's achieving top scores in performance and scalability. Bruce Weiner, President of Mindcraft, an independent testing vendor, stated that "DirectorySmart 4.7 with Microsoft Windows 2000 Active Directory… sets the TCO/Performance and Annual TCO/User standards against which other Web access control and identity management solutions will be measured" (OpenNetwork).

Balanced application environment support – Among the products reviewed, DirectorySmart provides balanced application support for both the Java/J2EE and Microsoft environments. This is particularly evident in the API support—Java, C/C++, and COM are supported. The other products reviewed favored a Java-centric environment.

Parsimonious solution – When compared with the other products, the DirectorySmart architecture presented the simplest design to implement, with the fewest "moving parts." The other major web access control solutions (e.g., Netegrity IdentityMinder/SiteMinder, Oblix NetPoint) required separate repositories and servers to manage identity and access control information.

Costs – In comparing the costs of DirectorySmart with the other major vendors, the price per user was quite favorable, considering the amount of functionality that it offered. Two of the top leading vendors required purchasing dual licenses (or a single license at effectively double the cost), one for the identity management and another for the access control.

After presenting this information to executive Information Technology decision makers, the product was approved for purchase and implementation.

The next decision involved which applications would be suitable candidates for integration with DirectorySmart. Over twenty applications were identified supporting specific business units or the entire enterprise. Interestingly, these applications could be categorized across a number of dimensions: current versus future; web-based versus client-server; small versus large user base; external versus internal; Java/BEA versus IIS/ColdFusion; and internally developed versus off-the-shelf.

Out of this discussion, two candidates emerged: the AMISC (an acronym of one of the Company's smaller business units) application, and PeopleSoft's Time & Labor. The AMISC application was chosen because the Loan Servicing business unit development team was seeking out a centralized directory-based access control solution, knowing that future applications could leverage the work done on AMISC. Additionally, the existing application served a small internal user base of ten individuals, was Java-based running on BEA WebLogic—one of the Company's strategic development platforms—contained only minimal authentication code, and had a requirement for role-based access control.

In contrast, PeopleSoft Time & Labor was chosen because it was a COTS product with an enterprise focus. The application had only been deployed to a pilot group of users, and OpenNetwork offered a connector specifically for PeopleSoft to ease the integration. Since every employee ultimately would have to use Time & Labor (for recording employee time and attendance), having DirectorySmart integrated into this application would be a major milestone with broad enterprise visibility.

The Analysis

Since AMISC was the first application to take advantage of the benefits of DirectorySmart, this application will be analyzed more closely to understand some of its security deficiencies and opportunities for improvement. The AMISC application was developed for one of the Company's subsidiaries that specialized in selling and servicing optional products to the Company's customers, primarily in the form of insurance policies. This application manages the day-to-day information flow related to these policies, augmenting the functions of a separate mainframe application that handles the sales component of the policies from outbound telemarketing efforts. The primary reason for creating the application was that the mainframe application was limited in flexibility, particularly with the servicing of the policies.

The Java-based application uses a BEA WebLogic application server to provide the Java Runtime Environment, and most importantly, uses the BEA Web Server to handle the presentation layer functions (i.e., serving up web pages). This point will be revisited in the implementation considerations. The AMISC application and data rely on a SQL*Server back-end for storage of information and stored procedures.

The authentication functions are handled by a Java Server Page, login.jsp, working in conjunction with a SQL*Server stored procedure. The jsp captures the User ID and password and passes it to the database stored procedure. The procedure encodes the password (using reverse hex), compares it with the encoded password value stored in a SQL table (LOOKUP). If the values match, the stored procedure sends a message back to the program that the user is successfully authenticated.

Concerning authorization, the current application has no provision for restricting certain functions to different classes of users, even though they can presently be separated into Processors and Managers; all users have the same functionality. This presents an operational risk, since Processors could update a value in a field, a process that only Managers should be able to perform.

The DirectorySmart Solution

Since a decision was made to implement DirectorySmart as a web access control solution, a review of its core components is in order to illustrate how it addresses the shortcomings of the original AMISC application and other previous web application security methods.

The core architectural components of DirectorySmart are as follows:

LDAP Directory (Directory) – This directory is the central repository for the DirectorySmart identity management elements (e.g., users, roles) and configuration management elements. The Directory houses virtually of the information associated with the DirectorySmart process. Note that the Directory does not need to be dedicated solely to DirectorySmart functions; in fact, it is possible to integrate an existing production LDAP directory used for generalized directory functions and extend the schema to accommodate the DirectorySmart objects, classes and attributes. In addition to Microsoft Active Directory, iPlanet Directory Server, IBM SecureWay Directory Server, and Novell NDS eDirectory Server are supported.

User Management (UM) – The User Management component is the central configuration console. Through this web-based interface, four major DirectorySmart object types are managed: Web Services, Organizations, Roles, and Users. The term “Web Services” is not to be confused with the more recently coined term “Web services” (small “s”) that describes an application that provides a Web API, supporting application-to-application communication using XML and the Web (Manes, p.27). In DirectorySmart parlance, it simply refers to an application protected by DirectorySmart. Users are the entities (sometimes referred to as principals) used to access applications and infrastructure mediated by DirectorySmart (Manes, p.233). Organizations are the method for organizing and managing Users, Roles, and Web Services. They can be set up in a flat structure, or into an organizational hierarchy (useful for delegated administration), using the optional Organizational Hierarchy (OH) component. Roles provide a way to grant a set of Users authorizations to Web Services, and as a result, provide a method to enforce a consistent policy (OpenNetwork, p.26). Additionally, Roles can provide a way to control a User’s access to specific functions within a Web Service through fine-grained access, discussed below.

Web Access Control Agent (WAC) – The WAC performs a vital function, serving to authorize and log initial and subsequent user requests to protected Web Services. It mediates the user login process, checking and creating credentials in the form of an encrypted browser session cookie. The Web Access Control Agent is a web-server plug-in filter; a variety of popular web server platforms are supported, including Microsoft IIS, iPlanet Web Server, IBM HTTP Server, RedHat Stronghold, Oracle Web Server, and IBM/Lotus Domino.

Self Service – The Self Service module is a Web Service that performs two essential functions in user identity management. The first, as the name suggests, is the self-service function, i.e., the ability of a user to review and update selected attributes about himself. User attributes such as phone number, extension, and e-mail can be entered and updated by the user as needed (OpenNetwork, p.1). The second function involves password management, and is the ability for a user to perform password updates, resets, or recoveries. In the first case, if a user wishes (or is required) to change a password, a screen is presented with three familiar fields common to most password update processes: Current Password, New Password, and Confirm Password. Once the user properly fills out the Current Password and enters in the new password in the second and third fields, the password is successfully changed. DirectorySmart supports user-defined Password Challenge Prompts and Responses. Provided that a user fills in these fields (e.g., Challenge Prompt: “What is your birthplace?”; Challenge Response: “Los Angeles”), this can be used in conjunction with password resets and/or password recovery if a user forgets a password. In such a case, a user would go to a page, enter his User ID, Challenge Prompt, and Challenge Response; for password resets, the password is randomly generated and presented to the user either on the screen or sent via e-mail; for forgotten passwords, the current password is displayed on the screen or sent via e-mail (OpenNetwork, pp.48,57-66).

Self-Registration – This module allows users to participate in DirectorySmart services after performing a self-registration process. The module supports two modes: Activation and Registration. In the case of Activation, dormant users are predefined in the Directory. When a dormant user visits the self-registration page, he is required to put in his valid User ID (which was already preloaded by the administrator) and a new password to activate his identity. In the second case, Registration requires that the user not pre-exist in the Directory. By default, the Registration mode will validate a user against an external database (of valid PIN numbers, for example), and upon the user’s successful entering of a User ID and password, a User object will be created in the Directory (OpenNetwork, pp.1-6)

Menu of Services (MOS) – The Menu of Services is an optional Web Service that provides a dynamic portal with some customization and personalization capabilities. Though not being used by the Company, the MOS provides an easy way to deploy a dynamic (i.e., processed in real-time) web-based menu of available applications (Web Services), based on a user’s Role and/or Organization (OpenNetwork, p.1).

Log Collector/Reporting – The Log Collector is a servlet-based application that is responsible for logging access, failed access, and audit events. It works in conjunction with the WAC to record events. Access and failed access events are generated each time a user accesses or, alternatively, fails to access, any page protected by a Web Service. Audit events are generated each time someone makes a change (e.g., add, modify, delete) to an object within the Directory. For added integrity and privacy, these events can be signed and encrypted. The events themselves are written to a relational database, such as Oracle, DB2, or SQL*Server. The Log Collector supports three modes of logging, depending on the degree to which an event is required to be verified that it was successfully written to the database. The most stringent form of logging (called “strict” mode) requires the Log Collector to confirm that an event was recorded to the database; if it is unsuccessful, it signals to the WAC to not allow any further access to any protected web applications (OpenNetwork, pp.1,38-39). To the report on these activities, a set of predefined reports built in Crystal Reports come with the product, and allow reporting on all three types of events, providing various levels of granularity (e.g., Organizations, Web Services, Users) (OpenNetwork, p.1).

DirectorySmart APIs – The DirectorySmart APIs are a set of Application Program Interfaces, available in C/C++, Java, and COM. The APIs allow direct access and control to the DirectorySmart-related objects in the Directory. The APIs can work in conjunction with Web Services to provide fine-grained access control.

In addition to DirectorySmart’s core components, two important features related to identity and access management must be reviewed, since they pervade much of the discussion and the security improvements offered by DirectorySmart: Authentication and Authorization.

Authentication – As a concise definition, authentication is “the process of confirming the correctness of the claimed identity” (Cole et al., p.A-127). In DirectorySmart, this is presentation of valid credentials to establish the identity of a principal. When a user requests a web page that is defined as a Web Service and one that requires a valid Role to access the page, the WAC will check to see if the user (and his browser) contains a valid credential in the form of a cookie. If true, then optionally, the credential is compared against the minimum authentication level the Web Service is set to. Four authentication levels, in ascending order of strength are supported: Basic (User ID and Password), Cert Only (User Certificates from a valid Certificate Authority), Cert & Basic (Both a User Certificate and User ID and Password), and SecureID (RSA SecureID Server). If a cookie does not exist, a standard or custom login page is presented to the user. When the user enters the proper authentication attributes (such as User ID and Password, in the simplest case), the WAC will query the Directory to perform the authentication. If successful, the WAC will create the encrypted cookie.

Authorization – Authorization is the process of granting specific access rights within a given system or application to a user (directly or through other indirect methods), allowing the user to perform the specific functions the authorization permits. DirectorySmart accomplishes authorization to applications and their components through two methods. The first method controls authorizations via URL-based security. Specifically, if a given web application has a sub-function (perhaps a particular web page of an application) represented by a particular URL, one can assign a Web service to the particular URL, and grant users access to it either directly or via a Role. The second method of authorization defined by DirectorySmart is known as fine-grained access control. Within a given Web Service, a particular function can be defined by giving it a Function

Name (e.g., “Purchase Goods”), a Function Code (e.g., Purch_goods), and Function Values (e.g., Full,Limited,OneItem). After the function is defined within the Web Service, the APIs can be used to query the function value for a given User/Role, thereby granting proper access based on the returned function value.

The Implementation

Now that the major DirectorySmart features and functions have been reviewed, this provides a backdrop to the implementation of DirectorySmart and the AMISC application within the Company’s infrastructure and outlines the challenges and considerations encountered.

Active Directory – Before the installation of DirectorySmart, the Active Directory needs to be sufficiently prepared. This would include extending the directory schema, setting up a Certificate Server to generate a certificate to facilitate SSL-based LDAP connections via Port 636 to the directory, and setting up the initial container to hold the DirectorySmart objects. (For more detailed information on these functions, please consult the DirectorySmart “Active Directory Pre-Installation Requirements Guide, Version 4.8.2.1” by OpenNetwork.) Microsoft’s Active Directory requires LDAP over SSL-based communications when performing password resets and creating Users under the User Manager (UM).

As part of the Company’s well-defined Active Directory structure, it had followed Microsoft’s suggestions for setting up a Dedicated Forest Root Domain, sometimes known as an “empty root” (Microsoft). This model worked well for the Company, since it managed semi-autonomous business entities, each with its own child tree. This had some minor implications on the installation. Since schema extensions apply at the Active Directory forest top level, applying the schema update to the empty root domain was not an issue. However, since the DirectorySmart installation was not inherently designed with subordinate trees in mind, it didn’t understand how to install the DirectorySmart objects within the desired target domain. In this instance, the Company worked with OpenNetwork to manually install pieces of the DirectorySmart directory object configuration to allow the installation to finish.

Web Access Control (WAC) – The classic DirectorySmart configuration involves the installation of WAC agents on the web servers hosting the web applications, requiring protection by DirectorySmart. During the early discovery phase of the project, it was revealed that the Company’s web infrastructure did not only include Microsoft IIS web servers, but also contained extensive use of BEA’s Web Server. Interestingly, this revealed a few faulty assumptions. First, since the BEA WebLogic application server platform was prevalent in the enterprise as well as IIS, it was assumed that IIS was being used in conjunction with BEA WebLogic, with minimal use of BEA Web Server. Second, since DirectorySmart supported the three major “flavors” of web servers (IIS, iPlanet, Apache), any new variant would be supported. Lastly, since OpenNetwork created

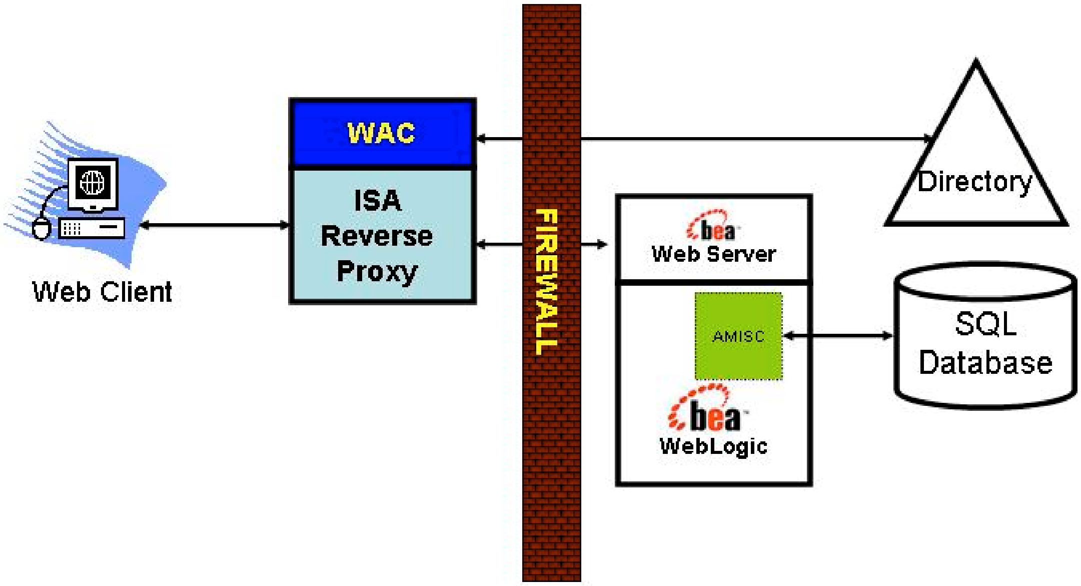
ApplicationShield, a separate OpenNetwork product that connected DirectorySmart to BEA WebLogic, it was thought that there should be a way to install the WAC on a BEA Web Server. Unfortunately, this was not the case—one cannot install a WAC on a BEA Web Server.

This left the implementation team with two options: either use IIS in place of BEA Web Server, or use a reverse proxy. Since the first option required using BEA's IIS plug-in, this was ruled out due to past negative experiences with stability.

Reverse Proxy – The reverse proxy solution provides an elegant alternative to installing a WAC directly on a web server. In this context, a reverse proxy is a service that takes HTTP/S requests from the "outside" and proxies them to the "inside," which is the reverse of a standard forward web proxy. First, a reverse proxy was set up using Microsoft ISA (Internet Security & Acceleration) Server. Microsoft refers to its reverse proxy capability as publishing web sites. At a high-level, ISA needs to be configured with the appropriate Destination Sets for each of the internal web servers, and then Publishing Rules are created that link the external URLs to the internal web servers (Microsoft). Once ISA is configured, the WAC can then be installed on ISA. When an authorized user requests a given URL, ISA receives it and sends it to the WAC; from there DirectorySmart applies its security-filtering techniques, allowing the valid request to go through.

Using a reverse proxy solution carries with it some additional benefits and considerations. This method does not require the installation of WACs on individual web servers, which can impose some administrative and performance overhead. Upgrading a WAC is now simplified, since only the WAC on the ISA server needs to be upgraded. This is also useful in situations where a firewall sits between the ISA server and internal web servers. Rather than opening up ports from the outside to web servers on the inside, only the reverse proxy is allowed to communicate directly with internal hosts (See Figure 1). Another advantage is that client connections are terminated at the ISA server, instead of directly with the web server. A corresponding proxy connection is created from the ISA server to the target web server; this makes it more difficult to directly attack the target web server. The potential downside is that for internally-based applications, one would need to use alternative methods to secure the BEA Web Servers, since one could attempt to directly access them, circumventing the ISA reverse proxy. This was controlled by restricting the network access to the BEA Web Servers so that only the ISA server could communicate with them for web requests.

Figure 1: WAC-ISA Installation with Firewall



Application Server – Ever since DirectorySmart version 4.7, a Java-based application server is required for DirectorySmart in order to run components such as User Management, Configuration, Self-Service, Log Collectors, and the Menu of Services (MOS). Given the Company's extensive use of BEA WebLogic, it was the obvious choice. For the initial implementation, the application server was used to house both the DirectorySmart items as well as the AMISC code. A future implementation will likely separate the AMISC application and the DirectorySmart component into two separate application servers.

Application Development – The development team decided that the best way to leverage the DirectorySmart APIs was to create a handful of encapsulated methods of the DirectorySmart API calls, such as getRoles() and getAllPermissions(). This would provide a layer of abstraction to the APIs and had two benefits: provide a common set of methods to Company developers, and prevent reworking of future applications if the DirectorySmart API changes were to occur. The other major discussion was the use of fine-grained access control. As of this writing, the application development team had not finished building function codes and function values to incorporate into the AMISC Web Service for fine-grained access control; however, they specified a high-level approach by bundling a number of application permissions into a reasonable set of function codes, rather than granulating a single permission (e.g., Display Button 1) to a single function code.

Other DirectorySmart infrastructure – Aside from these considerations, the implementation of the SQL*Server Database (for the Log Collectors and AMISC application) generally went “according to the book.” For the initial implementation, the default log settings were used, logging successful and failed access and audit events, with the logging mode set to Recoverable, which ensured that events were written to the database (or a backup text log file if the database is unavailable for later processing).

The Results

As the dust of the initial implementation settles, a number of security and business benefits will become apparent:

Security Improvements

Improved consistent provisioning of users – By moving towards a common directory-based identity, it becomes easier to provision and manage users, since there will be fewer separate security administration silos to manage. This helps to ensure that provisioning and deprovisioning will occur in a rapid and consistent manner.

Password management – From a security perspective, password resets and recoveries are sometimes seen as security weaknesses, since they offer opportunities for unauthorized individuals to gain access to systems by guessing people’s answers to predefined, easy challenge questions, leading to a limited number of easy answers (e.g., Question: “What is your favorite color?” Answer: “Blue”). However, in the case of DirectorySmart, the challenge question is completely open-ended to the user, as then will be the answer. When this is combined with its password reset and password recovery (i.e., “Forgot password?”) functions, this reduces the risk of unauthorized resets and recoveries. This added functionality can reduce the occurrence of the “Post-It Note” phenomenon, where users write their password down and stick it in a visible location by their workstation.

Common access control model – By implementing a centralized access control product, this creates a model for managing consistent access control policies, such as password policies, role-based security, and more granular permissions. This is in sharp contrast to the variety of methods that came about as a result of independent development methods to handle access control.

Improved authentication/authorization – The implementation clearly brought about improved authentication capabilities, particularly for AMISC, given that the former method used an encoding scheme for password storage with no authorization abilities. The new method requires DirectorySmart to mediate the authentication, securely pass the authentication information to the Directory, and then deliver to the client a strong encrypted cookie-based credential. Further, with DirectorySmart's URL-based and fine-grained access control, it provided a secure method for authorization to functions within the application.

Logging and auditing – The AMISC application did not have any mechanism for logging access activity or the ability to audit application administration events. With the Log Collector enabled, all accesses, failed accesses, and inactive or locked accounts were logged, available for later reporting. Further, any event related to directory changes, such as modifying an application Role, would be logged as well, providing an audit trail of administrative changes.

Improved defense in-depth – This term is used throughout the information security profession, and the SANS Security Essentials course devotes the entire Section II, and specifically Chapter 7 to this topic. As one reviews the post-implementation web security model, it is clear that additional layers of security have been added, reducing the exposure and providing increased defense in-depth. Specifically, the access to web content is controlled by a web access control agent, securely tied to a directory repository, as opposed to unrestricted access to the web server. This is further secured by the use of a reverse proxy, discussed earlier, which now makes it possible for n-tier web applications to be securely accessible by external participants. A final example would be the addition of event and audit logging, providing essential security information for monitoring and managing access and directory administration.

Business Improvements

Improved development – One of the successes of this initiative was that the developers themselves realized the value of DirectorySmart from an application development perspective. Specifically, they were no longer required to build security components into each one of their applications, thus freeing them to focus on other development efforts. In addition, they could utilize a common set of APIs and encapsulated methods.

Reduced administrative costs - Effective self-service tools, particularly password management, will improve user productivity and reduce administrative costs. A recent sampling of the Company's Help Desk indicated that the number of calls for password resets had reached 250-300 calls per day. By giving users an easy-to-use tool to reset their login (and their directory-enabled application) password, this will easily reduce the call volume and lower Help Desk-related support costs. For security administrators, this will reduce the amount of time spent on access management functions, since there are fewer separate access management environments to administer.

Increased user productivity – Another benefit of effective password management tools and single sign-on technology is that users themselves become more productive, since they are less likely to forget a password (since there are fewer passwords to remember), and if they do forget, a password reset/recovery mechanism gets them back to work quickly, instead of requiring the Help Desk to reset their passwords. Another direct benefit to new users is that with a centralized identity, it takes less time for security administrators to create their user account to give them access to all of their authorized applications, since the applications are tied to the singular identity. This translates into users being able to access their applications and hence become more productive sooner.

The Future

With the initial groundwork of DirectorySmart laid, this will pave the way for future applications using its directory-based access model. Once AMISC is fully implemented, this will be followed by integrating further Loan Servicing web applications. As part of the original project scope, PeopleSoft is still planned as the second enterprise application to be integrated. The need for integration with DirectorySmart has been increased, since the Human Resources department is not only in the process of rolling out Time & Labor, but also planning on deploying other PeopleSoft-driven self-service functions, particularly in the area of Benefits. On the external front, there are a number of outward-facing web applications serving the existing customers that will need to be eventually retrofit with DirectorySmart authentication and authorization. For potential external customers, the self-registration features of DirectorySmart should prove helpful in having users create their own secure user identities as they review the Company's products and services.

Aside from the applications themselves, the self-service and password management functions of DirectorySmart need to be rolled out to the enterprise to fully achieve the value of these solutions. A number of the internal applications have a need for delegated administration, where the business units themselves will perform the administration. DirectorySmart, through its User Manager interface, will provide an easy way to delegate the administration of selected Users, Organizations, and Roles, allowing them only to manage the relevant objects within the Directory.

Concerning the future of the product itself, as of this writing, OpenNetwork very recently launched the next major release of DirectorySmart, now called the Universal Identity Platform, or UIdP (OpenNetwork). This new release extends the product to the Microsoft .Net platform; this allows major components, such as the Universal Identity Manager (formerly UM), to run directly on a Windows 2000 or 2003 Server. The API's are now also available for .Net. UIdP also expands the Role-Based Access Control capabilities of the product by providing Dynamic Roles. Perhaps the most major development of the new product is the Enterprise edition of UIdP, which adds Microsoft's Identity Integration Server (MIIS) and a Universal Business Process Manager (BPM) to provide workflow and provisioning services.

Within the context of these future trends, this fits in well with the Company's strategy and resultant continued rapid growth. The next major enterprise security initiative focuses on Automated Provisioning, a process that is designed to automate the provisioning, maintenance, and de-provisioning account processes, and will have a profound impact on improved user productivity. Having a solid enterprise directory in place is crucial to the success of such a large undertaking, one which this author is ready to face.

References

Horvath, Mary. "Understanding Encrypt, ToBase64, and Hash." 2000. URL: http://www.macromedia.com/devnet/server_archive/articles/understanding_encrypt.html (20 Aug. 2003).

Cole, Eric, Jason Fossen, Stephen Northcutt, and Hal Pomeranz. SANS Security Essentials with CISSP CBK Version 2.1. 2 vols. SANS Press, 2003. 392,A-127.

Schneier, Bruce. "Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish)." 1994. URL: http://www.counterpane.com/bfsverlag.html (25 Aug. 2003).

Schneier, Bruce. "Security Pitfalls in Cryptography." 2003. URL: http://www.counterpane.com/pitfalls.html (23 Aug. 2003).

OpenNetwork. "Independent Testing." 2003. URL: http://www.opennetwork.com/solutions/testing/ (20 Aug. 2003).

Manes, T. Anne. Web Services: A Manager's Guide. Boston: Addison Wesley, 2003. 27,233.

OpenNetwork. "DirectorySmart User Management User Guide. Version 4.8.2.1." 7 May 2003. 26,48,57-66.

OpenNetwork. "Self Service Guide. Version 4.8.2.1." 7 May 2003. 1.

OpenNetwork. "Self-Registration User Guide. Version 4.8.2.1." 7 May 2003. 1-6.

OpenNetwork. "Menu of Services Guide. Version 4.8.2.1." 7 May 2003. 1.

OpenNetwork. "Log Collector Guide. Version 4.8.2.1." 28 May 2003. 1,38-39.

OpenNetwork. "Crystal Reports Guide. Version 4.8.2.1." 7 May 2003. 1.

OpenNetwork. "Active Directory Pre-Installation Requirements Guide. Version 4.8.2.1." 6 Jun. 2003.

Microsoft. “Choosing a Regional or Dedicated Forest Root Domain.” Windows 2003 Deployment Kit. 2003. URL: http://www.microsoft.com/technet/prodtechnol/windowsserver2003/proddocs/deployguide/dssbc_logi_abak.asp (18 Aug. 2003). [Active link not possible—paste URL into browser to view reference.]

Microsoft. “HOW TO: Securely Publish Multiple Web Sites by Using ISA Server in Windows 2000.” Microsoft TechNet Knowledge Base Article 300435. 27 Oct. 2002. URL: http://support.microsoft.com/default.aspx?scid=kb;en-us;300435 (23 Aug. 2003).

OpenNetwork. “OpenNetwork Announces the General Availability of Universal Identity Platform 5.0 for End-to-End Identity Management.” Press Release. 19 Aug. 2003. URL: http://www.opennetwork.com/news/press/2003/2003-08-19_UIdP.php (25 Aug. 2003).