VACMAN® Enterprise
Secure Single Sign-On for e-Business
Requirements and Solutions

White Paper

VASCO

SECURITY BEYOND e-MAGINATION

VACMAN® Enterprise Secure Single Sign-On for e-Business Requirements and Solutions

Contents

VACMAN® Enterprise
Secure Single Sign-On for e-Business Requirements and Solutions

Introduction

As e-business applications continue to proliferate, users find themselves losing the war in the authentication department. Every application that they are introduced to requires its own authentication or login. One application does this because it needs to know the identity of the user, another because it simply doesn't trust any previous authentication. The evolution of computing has introduced many ways to accomplish authentication, some stronger (in the security sense) than others. The field of single sign-on, or SSO for short, attempts to address the "multiple login" issue, as well as some others that are intrinsically tied to it (such as encryption, password synchronization, access control, and session management). This paper explores these issues, discusses how the VASCO single sign-on solution, VACMAN® Enterprise, addresses them, and makes some recommendations on what to look for in SSO products in general.

Why is SSO Important?

The importance of SSO cannot be understated. A secure single sign-on solution will deliver benefits in three areas:

- Increased user productivity
- Increased security
- Reduced costs

Increased User Productivity

If you can accomplish in one login what today takes ten logins, you would have to agree that there is inherent efficiency in making this happen. You no longer have to remember ten accounts and ten passwords. You do not have to remember when you will be asked to change your password on system X, what policies govern password management, and who you need to contact when you forget your password.

Increased Security

Even more important than ease of use, and its related increase in productivity, is increased security. Writing down passwords and storing them in your desk drawer, under the keyboard, or even in a file on your computer or PDA (personal digital assistant) is a bad idea. A single sign-on system should manage this for you securely and efficiently.

Some claim that by implementing SSO the risk is greater than without it. All an attacker has to do is break one password and he has them all. This argument is weak. If the password used to protect my SSO identity is inherently weak (because it is static or because it has a weak policy engine behind it), then it is probable that my other passwords are of "equal quality." So breaking any one of them would probably mean that the attacker has now discovered the password for more than half of the other systems at the same time. The time it takes to break a weak password (same name as user name, company name, months, and so on) is short. The vulnerability of any security system is measured by its weakest link. What we need to do is strengthen that link, not throw away the system.

Reduced Costs

Increased productivity and increased security produce cost savings. The increase in productivity is easy to measure (see Table 1). Increased security can have an even greater impact on a company's balance sheet. Just ask yourself: whom would you rather do business with, a company that knows how to secure its assets, or one that does not?

Events	Time/cost
Time for average login	5 sec
Avg. no. of applications	8
No. of times each application is used	3/day
Time spent signing on	120 sec/day
Time saved with Automation	240 sec/day
Time spent signing on	360 sec/day (6 min/day, 3 hrs/mo.) [A]
Time spent changing a password	30 sec/month
Total time changing passwords	240 sec/month [B]
Time spent recovering a password	480 sec/month
Avg. no of lost passwords	0.5 per month
Total time recovering passwords	240 sec/month [C]
Total time per user (A+B+C)	158 min/month (2.63 hrs/month)
Avg. hourly rate	$15
Cost	$39.50/month or $474/year per user
Company with 10,000 users	$4.7M/year

Table 1. Productivity enhancement and cost savings benefit of a single sign-on solution. Based on an average employee salary of $31,200 per year. Your figures may vary.

Definition of SSO

Believe it or not, SSO has been around for quite some time. Early distributed systems, such as Kerberos and Lotus Notes, implemented SSO as a matter of practice; that was the way that you built applications for those environments. (Notes was actually one of the first systems to utilize a PKI, even though it wasn't called that at the time.) The model was designed around a centralized repository that stored user information (your name and a shared secret – your password).

The applications that you built utilized an API that contacted the repository, performed the authentication event, and returned a network credential to the user. The authentication would succeed if the user provided the correct user name and password. Subsequent application invocations used the credential for proof of identity and all was well. This **single network credential** model is extremely powerful **if** all applications adhere to it. In Kerberos terms you would say that all your applications are "Kerberized," or programmed to work in the Kerberos system.

Unfortunately, the situation today is that there are many disparate computing environments, each with its own security infrastructure. Most of them are password based, but there are some that utilize other means of authentication, such as one-time passwords (OTP) or digital certificates. What an SSO system is forced to do in this scenario is to provide a "meta authentication facility," where the user first authenticates to the SSO system. When the user attempts to login to a subsequent application, the SSO system retrieves the user's name and password for that application from the SSO system and performs the login to the target system. Behind the scenes in this "authenticate once, login everywhere" model the system performs multiple logins resulting possibly in **multiple network credentials,** one for each target. A point to note here is that the SSO system itself is most likely based on the single network credential model. You do not want to have to authenticate to the SSO system every time you want to retrieve a user name and password.

SSO Requirements

SSO would be fairly benign and quite trivial to implement if we could stop here. But a robust **enterprise-ready** SSO system also needs to address the following requirements:

- Ease of Use: the SSO system needs to be easy to use and as transparent as possible to the end user (integrated with the user's platform's native login).
- Encryption: the SSO system should *never* transmit passwords unencrypted, and all security information should be stored encrypted.
- Multiple Authentication Mechanisms: the SSO system should support multiple forms of authentication, such as tokens, certificates, and smart cards.
- User (Account) Management: how do I add users to the system? What do I need to do to make this work for all my target systems?
- Password Synchronization: does the SSO system require that passwords be synchronized? How is this accomplished? How can it be avoided?

- Authorization: can the SSO system provide access control to itself and to the targets that it supports?
- Session Management: can the SSO system tell me who is logged in and possibly even force them off?
- Infrastructure: the SSO system cannot contain a single point of failure
- Implementation: what are the challenges of deploying the SSO system? How do I get all my users into the SSO system?

Lets take a look at each of the above requirements in a bit more detail.

Ease of Use — True Single Sign-On

Beware of "Reduced sign-on" or "Single sign-on for the web" — that's all you'll get.

Why should we not expect users to login once and only once? Surprisingly, this is one of the more difficult requirements to satisfy. This requires that the SSO system be integrated into the operating system login mechanism. Unfortunately, like many applications, the platform login mechanism is different between operating systems. Windows NT uses GINA (Graphical Identification and Network Authentication), Windows 9X uses a Network Control Panel device. Some Unix systems use PAM (Pluggable Authentication Mechanism) while others just use a plain login program, and mainframes use OS/390 Security Server, ACF/2, or Top Secret. A good SSO system will have thought about these and addressed them with add-on modules that are OS specific. Beware of the terms "Reduced Sign-on" or "Single Sign-on for the Web." You will get exactly that and no more. Such products are not enterprise solutions, with functionality limited to specific areas and possibly giving rise to incompatible islands of (in) security.

Ease of use is also measured in the system's user interface. Most SSO products offer some form of graphical front-end that aggregates the sign-on activity, allowing for point and click activation and monitoring of the login sessions. Intuitively that seems acceptable, however there are certain environments where this change of style might represent a departure from the norm. Some users may have created their own automated environment, whether through startup shortcuts or scripts, which will no longer work, since these are invoked immediately after platform login and not from the SSO tool. Graphical interfaces are nice, but they should not be mandatory.

The VACMAN Enterprise Solution — Ease of Use

Unlike "single sign-on for the web" or "reduced sign-on" products, the VACMAN Enterprise Integrated Login facility enables a truly single "single" sign-on. Users of Windows 9X, Windows 2000, Windows NT, Solaris, AIX, and HP-UX platforms need to log in only once on their workstation to be authenticated by VACMAN Enterprise and granted access to all enterprise resources. VACMAN Enterprise also can simultaneously generate web credentials, in the form of a cookie, to eliminate the need for a second login when the user accesses web resources protected by VACMAN Enterprise.

Encryption

Surprisingly, not all SSO solutions are secure.

Purchasing a SSO tool doesn't necessarily mean that you have bought a security solution. There are certainly those products on the market today that sell ease of use under the banner of security. Never confuse the two. A false sense of security is the worst kind of security. SSO products need to use encryption, as well as other accepted security practices. Most SSO products will utilize encryption for their own login / management interface. This is fairly straightforward for the vendor, since they control all aspects of the application (the client GUI and the SSO server).

However, this is not the case for the application interface. This is where SSO products differ. Most products that fall into the ease of use category will simply pass the user name and password to the application, and if the application itself does not use encryption, then the login continues to be insecure. Some SSO products address this by replacing components of the application, whether they are dynamic link libraries (DLLs) or entire executable images. This raises another set of issues (such as covering all applications or operating system upgrades) discussed later in this paper. The goal here is to define a new category of SSO products: Secure SSO (SSSO). Automating sign-on is not sufficient to meet today's e-business requirements. SSO products need to address the encryption aspects of the sign-on as well.

The VACMAN Enterprise Solution — Encryption

VACMAN Enterprise provides the highest levels of end-to-end encryption.

By default, VACMAN Enterprise provides the highest levels of end-to-end encryption, authentication, authorization, and auditing. Encryption types (DES, 3DES) and key lengths are selectable. For client / server applications, VACMAN Enterprise provides a thin client-side component (VACMAN Enterprise Desktop Client) that is responsible for managing and encrypting communications between the desktop, the VACMAN Enterprise security infrastructure, and target host. This lightweight component is downloadable and installable by the end user in less than 60 seconds and requires no end user configuration. All configuration information is kept in secure servers on the network.

The VACMAN Enterprise Desktop Client operates transparently, intercepting connection attempts between the user's machine and a target host, and establishing an encrypted tunnel between the client and the VACMAN Enterprise rule server protecting the target.

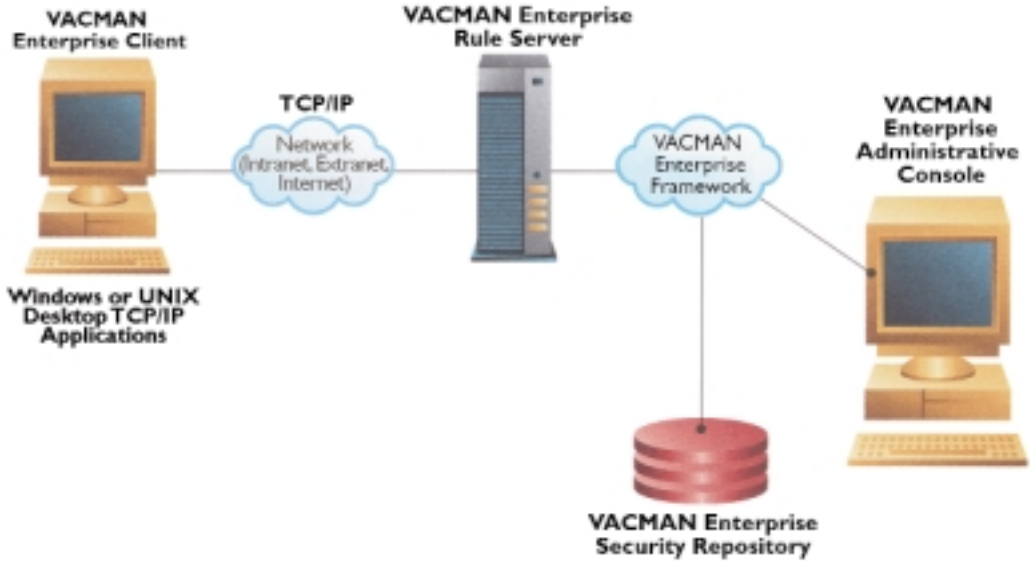

Figure 1. VACMAN Enterprise desktop communication with the VACMAN Enterprise infrastructure. All communications are encrypted. Passwords are always encrypted. Other encryption can be selectively turned off or on depending on application requirements.

Multiple Authentication Mechanisms

Make sure your SSO solution can interface with other authentication systems.

The authentication technologies available to us today are far more advanced than the password based systems that represent the bulk of our legacy infrastructure. IT is faced with the challenge of how to integrate stronger authentication techniques into applications without creating another security infrastructure to maintain.

One of the ways to introduce these newer authentication techniques into an organization is through SSO. If the SSO system can provide support for the new infrastructure, while at the same time making it easy to integrate the legacy password system, your organization will receive a greater return on investment. The overall security of the environment can be strengthened while at the same time making it easier to use.

The technical challenge for SSO products boils down to an exercise in "mapping." This is especially true when dealing with certificate-based or token-based systems. The SSO system must be capable of performing certificate login, requiring it to interface with a certificate directory (most likely through the use of LDAP — Lightweight Directory Access Protocol), in order to retrieve both the user's certificate and the certificate revocation list (CRL) associated with it. Once the authentication phase is complete, the SSO system will need to map the user (a simple way would be to use the CN — certificate holder's name — attribute from the certificate) to an internal profile, possibly containing names and passwords of other target systems. The same is true of token-based systems. In general, SSO systems need to be good at interfacing with other authentication systems. In a sense, they need to act as "authentication gateways."

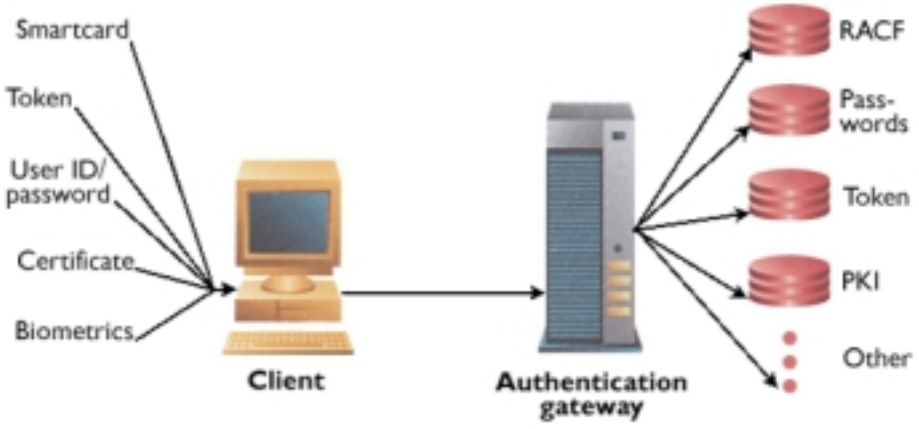

Figure 2. A secure single sign-on solution acting as an authentication gateway. All communications channels are encrypted.

The VACMAN Enterprise Solution — Multiple Authentication Mechanisms

VACMAN Enterprise integrates easily into computing environments where multiple authentication mechanisms are in use. VACMAN Enterprise unites disparate authentication mechanisms used within an organization, now or in the future, by mapping successful authentication using a given method to the user's global VACMAN Enterprise account and identity. This single network identity, or credential, which is digitally signed and cannot be forged, forms the basis for VACMAN Enterprise secure single sign-on to any VACMAN Enterprise-protected application or file.

In cases where the entity being authenticated (by any allowed method) does not have a VACMAN Enterprise account, administrators can enable Dynamic User Registration (DUR). In this case, after successful authentication, VACMAN Enterprise prompts the user for the information required to create an account, creates the account and network credential, and grants access rights to that user based on the rights of the groups or roles to which the user is initially assigned.

Account Management and Password Synchronization

Increase security through the use of random passwords generated and stored by the SSO system.

Every SSO system supports a management interface to its server. Administrators use this interface to add and delete user accounts, set or reset passwords, and enforce password and account policies on its users. That is fine for the SSO system itself — but what about the target systems that it is signing on to? How does the SSO system update its targets? Is that its responsibility? Is this to be done in a batch system or does it have to be updated in real time? Different products take different approaches to solving these issues.

Some SSO products do not attempt to deal with this issue and defer to the larger systems management products to handle account and password synchronization. Their sole mission in life is one of automating the sign-on to the target systems. In this case, these products will have an interface that will allow them to be updated by the larger system when an account is created or a password is changed. The SSO product is treated as another target in the management system.

Other SSO products focus on password synchronization with minimal account management. The premise behind these products is that SSO can only be accomplished with a synchronized password on all targets. This way the SSO product can be deployed entirely as a server-side solution with minimal interference on the client-side. Since all the passwords are the same, there is nothing to store and nothing to remember other than the single user name and password.

Figure 3. Synchronizing passwords on all target systems.

A third alternative is an SSO system that can operate both with synchronized passwords as well as non-synchronized passwords. The non-synchronized environment is far more difficult to manage. The SSO system needs to maintain an SSO map (call them “e-Post Its”) where the user’s profile defines the name and password for a specific application on a specific host. There is also an implied default hierarchy in case there is no specific entry for application or host.

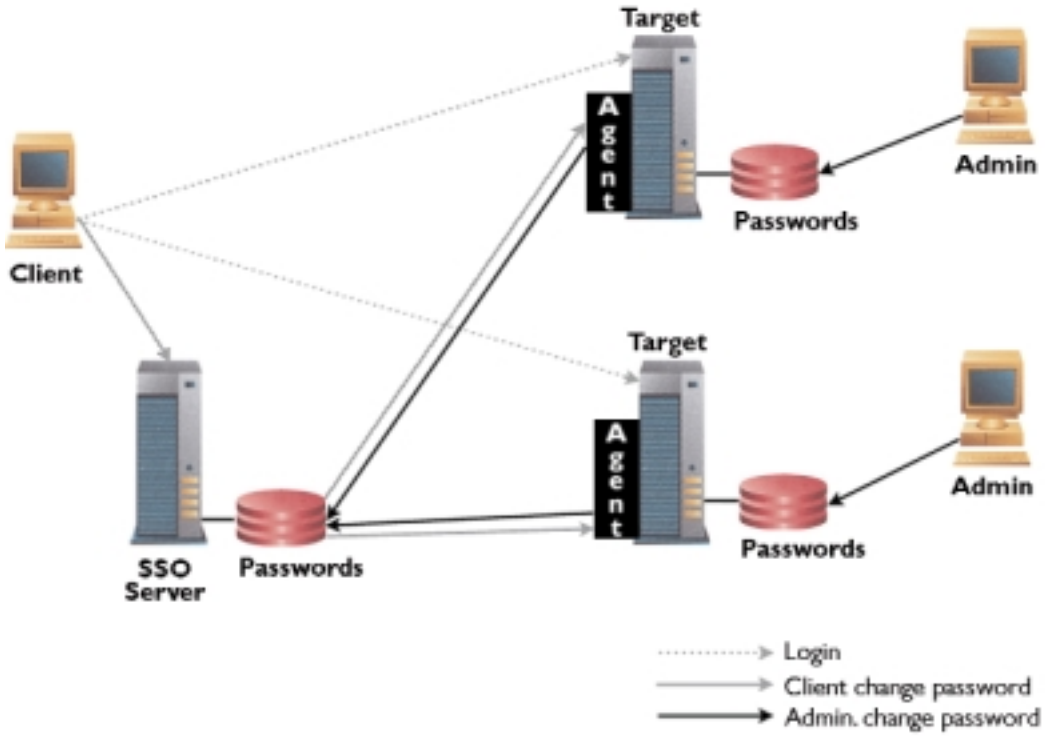

Figure 4. An SSO system operating with and without synchronized passwords.

Another advantage of non-synchronized SSO systems is that they can employ random passwords for any of the targets that they support. A random password is inherently stronger than a typed one. What is more important though than its strength is that a random password is never known by the end user. Therefore sign-on to the target system must be done through the SSO system and cannot be circumvented. As a safeguard, in the event that the SSO system is not operating properly, the administrator can always assign a temporary static password to allow the user to log in.

In all of the above cases one important aspect has not been mentioned. What happens when an administrator on target A changes your password or disables your account? How is the SSO system notified? The answer is different for the three implementations described above. In the case of a systems management tool this should never happen, because the premise is that the management tool is the center of the universe and the

targets are not to be administered independently. In the case of synchronized passwords, changing one will necessitate a change in all targets, which requires synchronizing agents on all targets. Depending on the number of target systems, this could be an expensive operation, especially if performed in real time. Finally, in the non-synchronized model, only the SSO Server needs to be updated with the change of the target, nothing else.

The VACMAN Enterprise Solution — Account Management & Password Synchronization

Centralized control of accounts and passwords across platforms eases management burdens and reduces costs.

VACMAN Enterprise provides a comprehensive graphical account manager for control of all security features pertaining to users, groups, policies, account attributes, and certificates. When an account is created, modified, or deleted in the VACMAN Enterprise infrastructure, VACMAN Enterprise can propagate that change or changes automatically to other, relevant, non-VACMAN Enterprise systems, easing the account management burdens placed on administrators.

VACMAN Enterprise belongs to the class of highly flexible single sign-on solutions that work with non-synchronized as well as synchronized passwords. For example, VACMAN Enterprise can interface with other authentication systems such as Windows NT, LDAP, RADIUS, OS/390 Security Server, UNIX and others, at run-time, giving it the ability to automatically synchronize any account or password changes. It also can use other authentication services, such as a password strength server, to perform password validation, history checking, and random password generation.

Authorization

Authorization is an advanced feature of SSO. It is not required per se to implement a successful SSO environment, but once a user is authenticated to the SSO system it would be beneficial if we could control their access to applications from one place. By placing access control features into the SSO server and managing them centrally, an administrator can control the use of applications throughout an enterprise.Common authorization requirements include enabling or disabling access by user, group, location, and time of day.

Authorization is not necessarily limited to the login itself. If the SSO system is well integrated into the target application environment, either through server-side agents or through higher-level management products, then it is possible to control access to application resources from the SSO system itself.

Sometimes there might be overlap between the authorization implemented in the SSO Server and the target application, while other times they actually might conflict. It is possible that the SSO system will allow a user to login to a database, but the DBA will have disabled the account temporarily or even permanently. This is one of the reasons why some SSO products shy away from authorization and focus simply on automating the login.

Once a user's identity is established by the SSO system, authorization capabilities give you added power and control.

The VACMAN Enterprise Solution — Authorization

Because VACMAN Enterprise operates at the protocol level, it can easily offer access control (authorization) as part of its secure single sign-on solution. VACMAN Enterprise uses software plug-ins, Protocol Support Modules (PSMs), to adapt itself to the resources it protects. Each PSM is a small piece of code that describes the resource (FTP or Oracle, for example), its operations, its data, and its state, to the VACMAN Enterprise Servers. The PSM tells VACMAN Enterprise how to perform login, how to provide fine-grain security features by inspecting the application protocol data stream, what requests and responses need to be generated on behalf of the user, and what information needs to be captured to provide authorization and auditing. VACMAN Enterprise is delivered with many PSMs, and users can write their own PSMs to enable security for new (or legacy) applications that are as yet unsupported.

VACMAN Enterprise performs authorization at the connection level (is this user allowed to connect to this host?), operation level (does this user have permission to use Telnet, or to perform an HTTP POST), and the object level (does this user have permission to access this file?). It also can manipulate the data contained in requests and returned as a result of the request based on rules set by the administrator. This last feature is especially important in tailoring application responses based on an individual's profile information stored with that user's VACMAN Enterprise account.

Session Management

Advanced SSO systems enable you to automatically log users off the network.

The SSO system also can track the number of users logged in, the times that they have logged on and off, an so on. This information is classified under the broad banner of session management. A good example of this is the RADIUS protocol. What is of more interest is the ability to "bounce" a user off the network. In a centralized environment, where all the terminals are connected to a single source this is not that difficult. In a distributed environment this becomes much more interesting. How does one get logged off automatically if there is no current network activity? A user could have achieved sign-on through the system but is currently not signed on to any particular application. Supporting this type of session management requires a more intricate design, one where the SSO system actually contacts the client desktop to "destroy" the SSO credentials.

The VACMAN Enterprise Solution — Session Management

VACMAN Enterprise provides session information as part of its auditing functions. Audited information includes connection attempts, type of operation requested, target objects requested, user identity information, date and time of requests, and so on. In addition, VACMAN Enterprise provides the mechanisms, such as inactivity timers, required to bounce users off the network, or off a web server, when necessary. Timers can be session wide, applying to all activities of a logged in user, or tied to the access of a specific resource. The use of such mechanisms is an important security practice, often used to reduce the chances of an unauthorized user gaining control of a machine that a registered, logged-in user has left unattended.

A Secure Single Sign-On Infrastructure

Look for replication of key services and automatic fail-over in any SSO solution.

It should be obvious by now that an SSO system is itself a large, multi-platform, distributed application. Therefore it requires all the common design principles that any system of this type should exhibit. The SSO Server needs to be replicated, both for fail-over as well as load balancing reasons. The SSO server can never be a single point of failure. In turn the SSO clients must be capable of locating the server with minimal effort. Users tend to get impatient when it takes too long to log in.

The SSO server must also serve as a security / policy engine. A good SSO system will know how to enforce strong passwords by performing actions like dictionary checking, password construction policies, account and password expiration policies, as well as support a hierarchy of these to make administration easier.

Finally, SSO systems today are being asked to scale to tens and hundreds of thousands of users. The storage aspects of such systems demand that SSO products utilize advanced database technology or an enterprise directory, preferably one with an LDAP interface to it.

The VACMAN Enterprise Solution — Infrastructure

VACMAN Enterprise is an enterprise class solution, providing secure single sign-on, and other security services, to your organization in a cohesive, manageable manner. VACMAN Enterprise backs up its services with a robust and scalable infrastructure, including replicated databases and services, which eliminates single points of failure and enables you to balance system loads for optimal performance.

Replication — To provide maximum scalability and failure isolation, while maintaining the core security infrastructure, VACMAN Enterprise partitions security functionality into distinct components:

- Security Server / Repository — all user and application authentication requests are mapped to unique network credentials. This is also the focal point for all user, group, and password database administration.
- Certificate Server — for environments that require access to certificate management services using browsers, and applications that already use certificate based authentication.
- Master Rule Server — this is where all connection and object rules are maintained.

Each of these servers is replicated, with automatic fail-over should one server become unavailable. The Master rule server is replicated as Backup and Slave servers, with backup servers ready at any time to become the master should the current master become unavailable.

Management — The VACMAN Enterprise infrastructure enables VACMAN Enterprise server resources, and the security rules they contain, to be managed securely from anywhere in the network. After authentication, administrators can use the VACMAN Enterprise administrative console from any node.

Implementing a Single Sign-On System

Evaluating and procuring a single sign-on system involves many factors. Features and benefits of various SSO products must be analyzed, the hardware necessary for the deployment must be procured, management support must be lined up, the proper personnel to implement the system must be brought on, and overall management of the project must be planned. But in this process, two key issues are often overlooked. The first is software distribution and the other is database population.

Software Distribution

Effective software distribution and SSO database population is essential for any SSO implementation.

Software distribution is really not an issue for web applications. Every desktop comes equipped with a browser. The web protocols of today can perform basic authentication and certificate-based authentication. Other web-based applications utilize their own private login pages, which SSO products need to know how to deal with. There is nothing special that needs to be installed on the desktop to secure a browser. No wonder most of the web security vendors advertise support for single sign-on.

The same is not true of client-server and legacy applications. Here we ultimately need to put something on the desktop. The issue now becomes how easy is it to deploy the client-side component of SSO? You cannot afford to make a mistake here. What it would cost to rectify a mistake across 10,000 desktops? Ask the following questions, and make sure the vendor of the solution you choose has an acceptable answer.

- How big is the desktop component?
- How easy is it to install and un-install?
- Can deployment be simplified by having end users do the installation?
- Once installed, is the solution easy to configure?
- How much information about the SSO system is stored locally on the desktop vs. on the SSO server?

The answers to these questions are vital in evaluating and deploying a SSO solution.

The second issue that most organizations never consider until it is too late is populating the SSO database. Imagine having the following conversation with your SSO vendor:

Vendor: "Ok, we're about to bring up the new SSO system."

Customer: "Great, can't wait."

Vendor: "Oh, by the way, we are going to need all your accounts and passwords to initialize the new system."

Customer: "Great, they're all on the mainframe in RACF."

Vendor: "Oh, we can't use those (sigh)."

Customer: "Why not?"

Vendor: "Well, what is actually stored for the password is a hash, so it is useless to us. Furthermore, the RACF IDs are not accepted as proper account names in the new system."

Customer: "So what do you propose?"

Vendor: "How about we reset everything and start from scratch?"

Scene: Mental image of vendor getting knocked on the head with a frying pan…

This is symptomatic of what we call the "Genesis Event." It is unacceptable for a vendor to ask customers to completely tear down the existing security infrastructures, rename accounts, reset passwords and policies in the name of a new SSO system. No company can afford to be "reborn again" just to implement SSO. That is why a mature SSO system must be capable of a smooth, almost seamless integration of the existing security environments into its own.

SSO Database Population Through User Self Registration

Dynamic User Registration is key to a successful and less costly SSO deployment.

The first technique suitable to solve this problem is user self-registration. Once the SSO system is in place, users are put through a quick and easy registration process. The classic problem of assigning a user a new account on a system is distributing the password to them. With user self-registration, the first registration event the user uses an account and password that can be validated against an existing security database. This serves a number of purposes. First it validates the user before loading their information into the SSO

system. Second it allows the SSO system to “activate” the user, possibly asking them for some additional pieces of information. Finally, it allows the SSO system to synchronize itself with the other target systems. Part of this final phase of registration could be to create accounts on other target systems or to contact the company's public key infrastructure (PKI) and register for a certificate.

Password Learning

Another requirement for a quick implementation of a SSO system is password learning. Since most password databases are not useable in an import operation, there is no way for an administrator to insert your password into the SSO system for that particular target. Instead, the SSO system simply monitors your login to that application the first time, records your password and stores it, encrypted, of course, in the SSO system. The next time the user attempts to login the SSO system will perform the action.

The VACMAN Enterprise Solution — Ease of Implementation

VACMAN Enterprise eases the transition to a secure single sign-on system by addressing several deployment issues:

- Self-installing, lightweight client software for the end user’s desktop or portable computer. The client performs automatic encryption and tunneling for communications between the desktop and the VACMAN Enterprise infrastructure and its protected resources, including legacy applications. The VACMAN Enterprise client software is small enough to download to the desktop machine and self-install in less than 60 seconds. No configuration is required on the part of the end user. Configuration information is securely stored on VACMAN Enterprise servers.

For web applications there is nothing to install on the desktop. VACMAN Enterprise takes care of performing single sign-on with a normal browser.

- Dynamic User Registration (DUR). With VACMAN Enterprise DUR, new users (which may be all users in a new SSO installation) without a VACMAN Enterprise account can register, automatically have an account created, and be granted a VACMAN Enterprise identity. Upon first login to VACMAN Enterprise,

users must authenticate against an alternate, non-VACMAN Enterprise source of information such as a human resources file, or a third-party authentication mechanism. After this authentication, DUR proceeds to create the VACMAN Enterprise account and accounts on any other target systems.

- Password recording. VACMAN Enterprise can be configured to learn target-/application- specific passwords as the user enters them the first time. These learned passwords are stored encrypted in the VACMAN Enterprise security repository. VACMAN Enterprise then automatically takes care of the login during subsequent attempts by that user.

Sample SSO Architectures

Now that we have explored a variety of requirements for SSO systems, it is time that to take a look at some of the more popular architectures of SSO systems. These architectures fall into five categories:

- Scripting
- API toolkit
- DLL replacement
- Application replacement
- Protocol interception, inspection and manipulation

A closer examination of each of these architectures will show their strengths and weaknesses.

Scripting

Scripting can automate logins but doesn't necessarily secure existing applications.

Scripting is by far the simplest approach. It is fairly easy to implement, as it is non-invasive to either the client or server applications. Its primary goal is to automate the login procedure. At the same time it is not concerned with securing any of the existing applications. Also, scripts are not necessarily intuitive to write and certainly present a scalability concern if they need to be managed on each and every workstation.

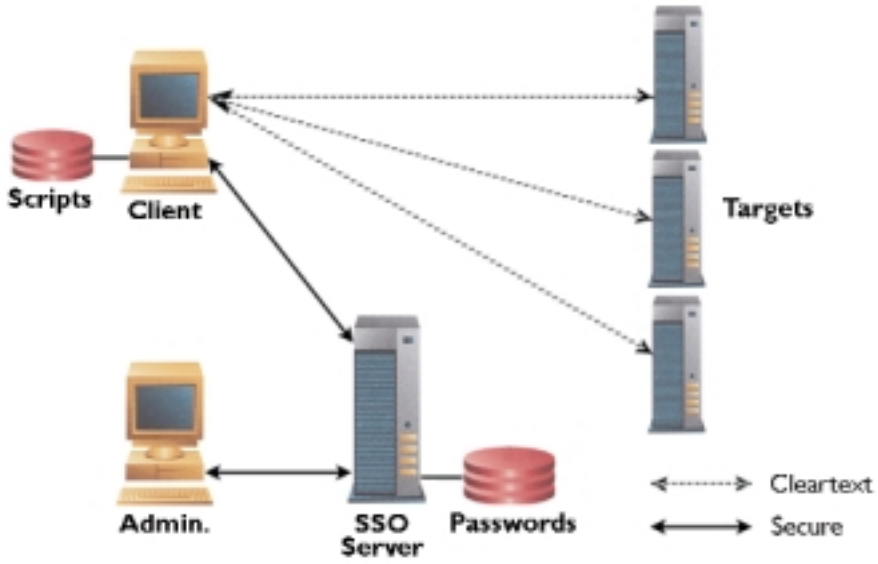

Figure 5. A scripting approach to single sign-on. The script retrieves the user ID and password from the SSO server, but the client may transmit the user ID/password unencrypted to the application.

API Toolkits

API toolkits require diversion of programmer resources from business functionality to security functionality.

API toolkits are probably the most common method for enabling security, not just SSO, in applications today. It is also by far the most invasive. Customers are required to gather all the applications that they want to secure and re-implement them using a brand new set of APIs. This approach certainly has the advantage that security is embedded inside the application and therefore can achieve a high degree of optimization and efficiency. However, this method does not work at all for COTS (commercial off the shelf) products, where the customer does not own the source code to the application.

DLL Replacement

DLL replacements are similar to the API approach, except that here the SSO system is making an assumption that most of the applications use one or more of the same system DLLs, and therefore they replace them with ones that implement authentication and credentials management. These SSO systems most often also augment the network connection with encryption facilities. The issue here is that it is impossible to cover all applications by replacing DLLs. A good example is the WINSOCK DLL. Some SSO implementations replace this DLL because it is a popular one that is used for socket-based applications. However, Microsoft Office applications do not use sockets, they use a different network paradigm entirely (it is called SMB). Replacing system DLLs can also be problematic when a new platform distribution is loaded on the workstation, as it will often replace the SSO DLL.

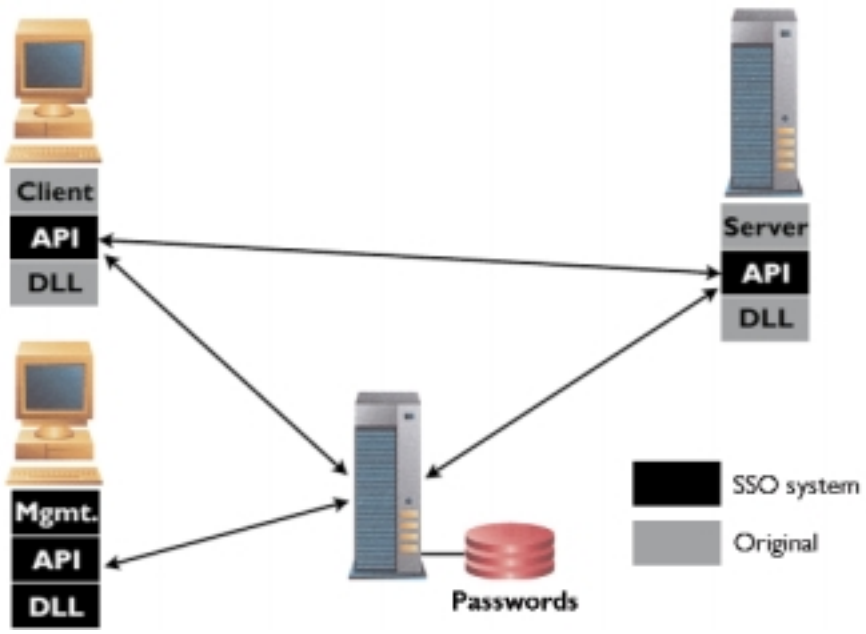

Figure 6. Using API toolkits to effect secure single sign-on requires re-coding of each existing application, and programming to the SSO API for new applications.

Application Replacement

Yet another approach is to replace commonly used applications with ones that are SSO-enabled. Although this approach probably yields the tightest integration between the application and the security infrastructure, it is completely impractical to ask one SSO vendor to build hundreds of applications, especially when that vendor doesn't own them. Furthermore, this puts a significant burden on the customer to deploy this type of solution. Customers quickly realize that instead of being in the business they were in yesterday, they now have been sucked into the software distribution business.

Protocol Interception, Inspection, and Manipulation

Enable SSO and strong security without application coding or DLL replacement.

The last approach is yet another twist on the DLL and API toolkit approach. Instead of replacing system DLLs or introducing modified applications, the protocol interception, inspection and manipulation approach tackles security at two different layers: network and application. Network interception is often accomplished through the dynamic loading of a device driver. The driver is responsible for low-level connection management and encryption. The application layer is responsible for adding security content to the application dynamically, at runtime, so there is no modification to existing applications.

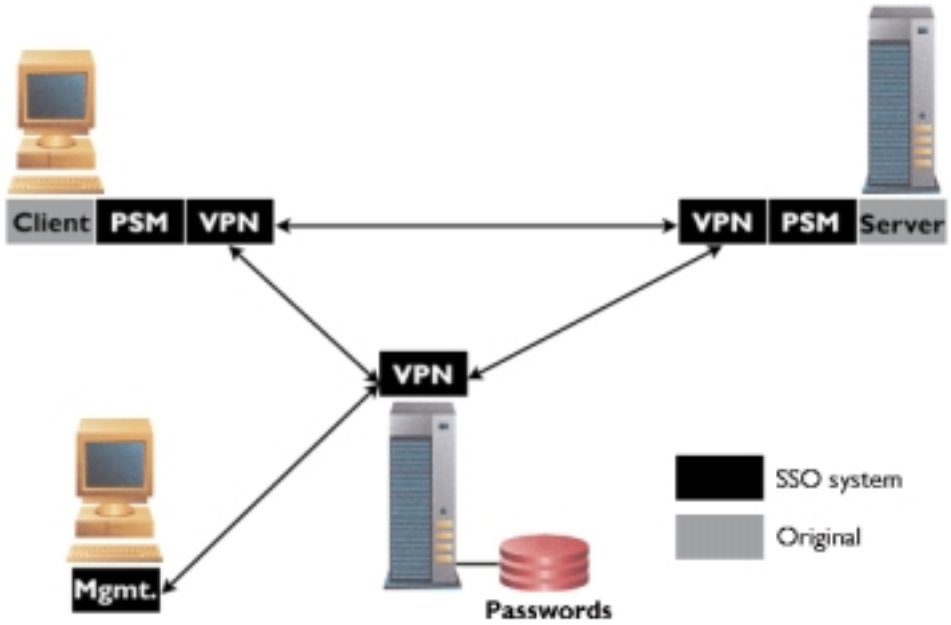

Figure 7. Secure single sign-on using protocol interception. Relevant network traffic is intercepted, examined, and processed below the application layer. The PSM (protocol support module) tells the system how to handle the traffic of a particular application (Telnet, Oracle, FTP, etc.).

This approach has a number of advantages. First it is totally transparent to existing applications. There is no dependency on the application architecture as it relies only on the network connection. Second, it is quick and easy to deploy. No applications are replaced. No DLLs are modified. No toolkits are required. By adding authorization parameters to the system, total security enforcement is possible. The downside to this approach is that as new operating systems are introduced, the SSO system has to keep up with these changes.

Table 2 summarizes the pros and cons of these architectures.

Architecture	Pros	Cons
Scripting	-Simple. -Easy to use, easy to adapt. -Least intrusive to application.	-No encryption. -Distribution/management issues if kept on the desktop. -Security risk if password is stored in the script. -Unable to provide multi-tier authentication.
API Toolkits	-Native security is always the most efficient. -Easier to map security points into application.	-Requires retooling every application. -Which triggers another distribution cycle. -Very difficult to accomplish with legacy applications. -Impossible with COTS products. -Time to market.
DLL Replacement	-No retooling of applications. -Transparent to application.	-Application conflicts -How many DLLs do you replace? -Replacing WINSOCK is not sufficient. -OS upgrades might destroy DLL.
Application Replacement	-Highest degree of transparency.	-Not scalable (need to replace EVERYTHING). -No interoperability. -Vendor dependency. -Customer now in the software distribution business.
Protocol Interception, Inspection and Manipulation	-Application and DLL transparency. -Enforces security policy at protocol level. -Easy and quick to deploy. -Combines the best of packet filtering and application proxy technologies. -Provides multi-tier authentication.	-OS dependency.

Table 2. Pros and cons of SSO architectures.

The VACMAN Enterprise Solution — Architecture

The last thing an organization needs is to be asked to re-write applications, or to modify the application execution environment, or to require the scrapping of an existing password-based system to enable strong security. Because the VACMAN Enterprise architecture utilizes the protocol interception, inspection, and manipulation model, you can realize the benefits of strong security for TCP/IP networks without the pain of re-inventing your current computing environment. You can:

- Strengthen the use of passwords in your organization while bringing in newer, stronger authentication methods such as digital certificates or token-based systems.
- Achieve significant cost savings by normalizing today's various security management tasks under a single infrastructure; by reducing the demand on technical support for recovery of lost passwords; by reducing the amount of time users spend logging in to various systems each day; and by reducing losses incurred by security breaches.
- Realize revenue enhancement from the positive effect on your current and prospective networked business partners when they know that shared data is secure, and available only to designated individuals properly authenticated and authorized.

Conclusions

Single sign-on is not easy. Solutions range from use as a simple automation tool to one that encompasses a variety of management and security disciplines. Before implementation, take it slow. Do your research and find the SSO product that most matches your requirements and your user population. There always will be tradeoffs between ease of use, complexity of administration, and security. No one product does it all. If you don't have the expertise in-house to implement SSO, seek the assistance from your vendor or a third party. At the end of the day you will be glad you did.

About VASCO

VASCO secures the enterprise from the mainframe to the Internet with infrastructure solutions that enable secure e-business and e-commerce, protect sensitive information, and safeguard the identity of users. The company's Digipass® and VACMAN® product families offer end-to-end security through strong authentication and digital signature, true and secure single sign-on, access control, and web portal security, while sharply reducing the time and effort required to deploy and manage security. VASCO's customers include hundreds of financial institutions, blue-chip corporations, and government agencies in more than 50 countries. More information is available at www.vasco.com.

For regional offices or to learn more about us, visit our web site at www.vasco.com

SECURITY BEYOND e-MAGINATION

AMERICAS HQ
VASCO Data Security, Inc.
1901 Meyers Road, Suite 210
Oakbrook Terrace, Illinois 60181, USA
phone: +1.630.932.8844
fax: +1.630.932.8852
e-mail: info_usa@vasco.com

EMEA HQ
VASCO Data Security nv/sa
Koningin Astridlaan 164
B-1780 Wemmel, Belgium
phone: +32.2.456.98.10
fax: +32.2.456.98.20
e-mail: info_europe@vasco.com

APAC HQ
VASCO Data Security Asia-Pacific Pte Ltd.
#15–03 Prudential Tower, 30 Cecil Street
049712 Singapore
phone: +65.232.2727
fax: +65.232.2888
email: info_asia@vasco.com