ENTERPRISE-WIDE SECURITY: AUTHENTICATION AND SINGLE SIGN-ON

A NAC POSITION PAPER

JULY 14, 1996

ABOUT NAC

The Network Applications Consortium (NAC) is a strategic end-user organization dedicated to improving the interoperability of mission-critical applications in a heterogeneous inter-enterprise computing environment. The Consortium's goal is to influence the strategic direction of vendors developing enterprise application and infrastructure technologies. NAC focuses on providing vendors with input and feedback regarding product development and marketing strategies by:

- publishing papers that state NAC's strategic vision of the industry's direction;
- educating vendors on end-user enterprise-wide computing requirements;
- promoting, facilitating, and documenting collaboration among NAC members and vendors;
- advising distributed computing vendors on marketing and product development strategies.

NAC members include:

ABB Power T&D Co.
American Bureau of Shipping
American Medical Securities
Australian Bureau of Statistics
Bell Atlantic Mobile Systems, Inc.
Carolina Power & Light Co.
Compaq Computer Corporation
Continental Grain Company
Federal Deposit Insurance Corporation
International Finance Corporation
MCI Telecommunications
Michigan Department of Commerce
Nike, Inc.
NYNEX
Pacific Bell
Pacific Gas & Electric
Pennsylvania Blue Shield
Public Service Electric & Gas
United States Marine Corps
University of Michigan
World Bank

This paper is the result of NAC's Strategic Interest Group (SIG) process, a collaborative effort of a subset of NAC members whose mission is to provide a cohesive NAC viewpoint on a particular industry sector or technical topic. The following NAC members were instrumental in writing this paper:

Carolina Power & Light	*Harold Albrecht, Steve McGehee*
Compaq Computer Corporation	*Mike Wilhite (SIG Leader)*
Federal Deposit Insurance Co.	*Chuck Taylor*
MCI Telecommunications	*Brian Plackis, Jerry Robinson, Sam Rockwell, Ron Thomas*
Pacific Bell	*James Brentano*
Pennsylvania Blue Shield	*Todd Sebastian*
United States Marine Corps	*Mark Johnson, Janet Palmer*
University of Michigan	*Gordon Leacock*
NetResults	*Doug Obeid*
Authors:	*James Brentano, Kelli Wiseth*

We welcome your feedback about this paper. For more information contact:

Doug Obeid, Executive Director
Network Applications Consortium
c/o NetResults
5214-F Diamond Heights Blvd., Suite 705
San Francisco, CA 94131

Contents

Executive Summary

Three years ago, the directory services market was immature, without clearly identified industry-wide standards. In fact, the concept of an enterprise-wide directory service was itself rarely understood except by leading large organizations such as those comprising NAC. Such organizations became the "voice in the wilderness" about enterprise-wide directory services, touting their benefits and encouraging vendors to deliver products to meet the needs of large organizations.

Today, thanks in part to NAC's participation in the industry dialogue, the directory services story is different: the market is maturing as seen in events such as the industry-wide endorsement of LDAP[1]; the development of Microsoft's client-side solution, ODSI[2]; Novell's delivery of a viable directory service in NetWare 4.1x[3]; and in the "product-izing" of key technologies such as Banyan's StreetTalk through the Universal StreetTalk API.

What does this have to do with security services? NAC believes security services are now in much the same state directory services were three years ago. We see an immature marketplace that doesn't yet fully understand the need for "security services" as a core infrastructure-level service, much less the need for a common standard for security services. Instead, the marketplace is focused on point solutions and stop-gap measures to address problems caused by lack of interoperability.

Although awareness of security issues has heightened considerably in the past year, thanks to wide press coverage of the Internet, digital commerce, and other high-profile trends, the media coverage hasn't helped the cause for IT managers and security analysts who must deal with the false perception that "all the problems are being solved" when in reality, the situation is getting worse.

Nonetheless, NAC continues to champion its vision of a core set of interoperable, infrastructure-level services, including security as well as directory and messaging[4], which interoperate with each other and support all the applications, databases, and services in an organization (see *Appendix A. "NAC Authentication Services Model"* for more information). Some basic business and functional requirements that a security service should meet include:

---

[1] Lightweight Directory Access Protocol.

[2] Open Directory Services Interface.

[3] Novell NDS (NetWare Directory Service)

[4] See NAC's prior papers on enterprise-wide messaging and directory services for more information.

- Distributed processing (client-server) architecture in which the functions of the authentication service itself are exposed through an application programming interface.
- Scaleable to an enterprise-wide scope.
- Capable of secure communications and store.
- Mechanism independent, capable of supporting smart tokens, biometrics, and other two-factor authentication mechanisms as well as passwords.
- Capable of supporting legacy applications.
- Extensible. Other security capabilities and functions that meet the specific needs of a given organization must be able to be added on to the authentication service without affecting interoperability.
- Portable (both capable of being, and likely to be, acceptable across all platforms)
- Easy for end-users to use and for administrators to maintain.

Given the current state of the enterprise, however, the realization of NAC's vision is quite a ways off. In the meantime, NAC in this paper focuses on an immediate concern in the security arena, specifically the identification and authentication process and the problem of multiple logons. Numerous, multiple logons result in:

- Lost productivity.
- Increased potential for compromised security due to lack of convenience.
- Increased support costs.
- Devaluation of passwords as a security mechanism
- Increased administration expense.
- Lack of availability due to expired, forgotten, or out-of-synch passwords.

The problem is a direct result of the lack of interoperability among different operating systems, applications, and services, and it isn't a problem that's going to be solved quickly. Despite the fact that most organizations recognize the need to move away from multiple security implementations, they are at the same time forced to continue implementing native security mechanisms included with the NOS, the database, or the application in order to quickly deploy the applications they need to run their businesses and remain competitive. And the situation will get worse before it gets better. Client-server deployment continues unabated, and at the same time, legacy applications must continue to be supported. End-users are also getting into the development act: Gartner Group predicts that "the volume of applications created directly by users is set to double in the next five years," and that "the average complexity of these applications will increase by at least 50 percent."[5]

In this paper, NAC delves further into these issues, specifically authentication, the problems associated with multiple logons, and the various approaches to achieving single sign-on. The field is marked by no standards, big dollars, and high risk.

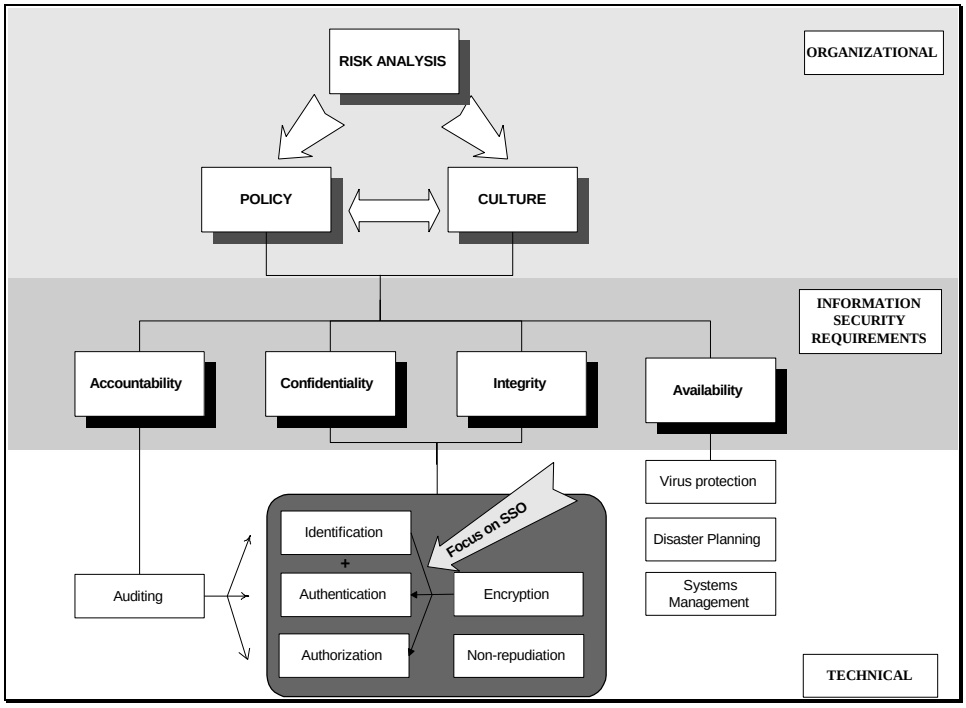
---

[5] Gartner Group Strategic Planning Research Note (SPA-200-109; March 28, 1996; S. Levin)

Introduction

In its first paper on enterprise-wide security,[6] NAC provided a high-level view of the business and technical issues that encompass the distributed computing environment. NAC developed the NAC Security Framework, shown in the figure below, as a means of exploring the security issues relevant to large organizations.

Figure 1. NAC Security Framework



The following is a brief synopsis of the key points NAC presented in its initial paper on enterprise-wide security.

- *Information security* encompasses organizational as well as technical issues, and the organizational issues are no easier to solve than the technical ones. The success of all security measures depends on the symbiotic relationship between *risk analysis*, the organization's *culture*, and the security *policy*.

- Overall *information asset integrity* is the primary motivation behind security implementations: Organizations recognize that information is one of their most important assets, that it must be protected from harm or theft, but that access to it by appropriate individuals must be ensured without impeding productivity or costing too much. "Cost" is defined as TCO (Total Cost of Ownership),

[6] "*Enterprise-wide Security: A NAC Position Paper*" April 1, 1996.

which includes acquisition, implementation, administration, management, and support over the long-term, for the full life-cycle of the technology.

- Information security *requirements* include *confidentiality, integrity*, *availability*, and *accountability*. A variety of mechanisms meet these requirements.

- The most fundamental protection mechanism is the process of *identification and authentication*: Before users can access information resources, a mechanism forces users to 1) identify themselves and, 2) to prove they are who they say they are by providing some evidence.

- The identification and authentication process (or simply the *authentication process*), is typically accomplished by entering a user or logon ID and a password. From an end-user perspective, this activity is generally referred to as the *logon process;* from a service perspective, this activity is generally referred to as *authenticating users.*

- Unfortunately, the process of authenticating users is not an infrastructure-level service (although NAC's long-term vision is that it should be). Each environment, application, database, or service typically has its own mechanism for authenticating user identity. The process is costly, error- and risk-prone, resulting in:

    - Lack of productivity
    - Increased potential for compromised security due to lack of convenience
    - Increased support costs
    - Devaluation of password as a security mechanism
    - Increased administration expense
    - Lack of availability due to expired, forgotten,or out-of-synch passwords

For these reasons, NAC believes that it is crucial to focus on *authentication services* and *single sign-on* implementations at this time. As NAC has done in previous papers, we first present a functional description of the authentication process. We then describe authentication in the context of the heterogeneous enterprise computing environment in order to highlight the problems associated with multiple logons and begin to explore the costs and the risks. Next, NAC examines three key approaches that have emerged to minimize the problem, evaluating each in terms of strengths and weaknesses. Finally, we present our recommendations to infrastructure vendors, application vendors, and NAC member companies and other organizations facing the same issues.

Authentication Process: Functional Overview

Verifying user identity prior to providing access is typically referred to as *Identification and Authentication*, or *I&A.* In a distributed, client-server environment, identification and authentication is a two-step process that validates the purported identity of a network user (subject) prior to providing that user access to a network resource (object). For example, when users wish to connect to a client-server database application, they typically enter a user name or ID and an additional piece of information that validates identity.

The "additional piece of information that validates identity" — the *authentication token* — can include factors such something the user knows; something the user has; or something the user *is*. Examples are shown in the table below:

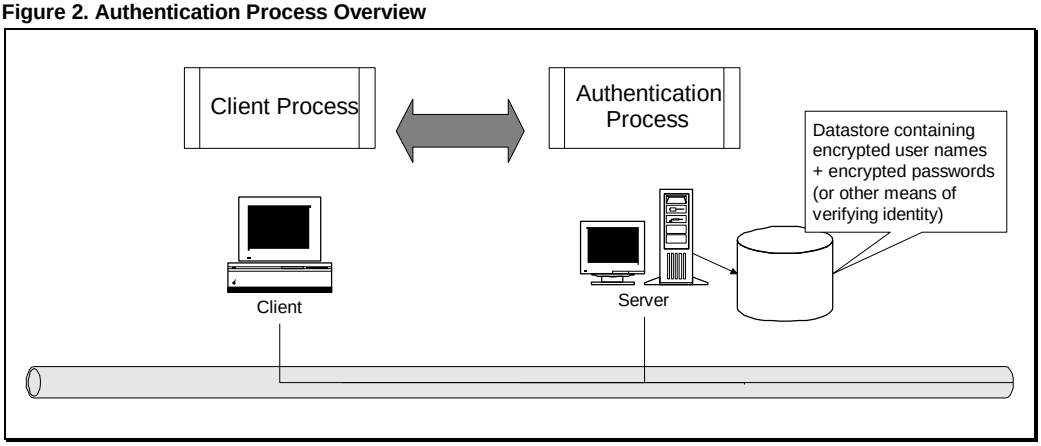
Authentication Token	Security
Biological information, such as a fingerprint, voice sample, or retina image map that can be compared to a biometric template	Most secure ↑
Physical smart-card (also called a "token-card") such as SecurID token-card and Security Dynamics ACE/Server combination that generates and checks a pseudo-random numeric sequence	↑ ↑
Password	Least secure

In general, security is enhanced when two of these authentication tokens are combined for *two-factor authentication;* a two-factor authentication[7] process might require users to provide not only information that they know — say, a password — but also something that they have in their possession, such as a randomly generated number from a token-card which maps to the same randomly generated number on a server. Either factor alone — the password or the token-card — is not adequate.

Regardless of the specific factor used to validate identity, the process works as shown in the figure and described in the table below.

[7] NAC believes that two-factor authentication is becoming a minimum security requirement.

Figure 2. Authentication Process Overview



Client Process	Server Process
1. User enters name and authentication token.	3. Compares encrypted User name and authentication token to encrypted version of the token in authentication database.
2. Client process sends User name and authentication token to server. Entry may be encrypted before sending over network.	4. If entry in database matches entry sent from client, service grants access (based on ACLs (access control lists or other authorization mechanism specific to the service or application.) — If no match, access is denied.

It's the identity of the *user* as an individual that's important; although the figure shows the client and server *processes* validating each other, it's the purported identity of the user that is actually validated — the server confirms that the user sending the request is the same user that it has record of in its database of users.

After the server process validates user identity, the user can access the network object within the parameters specified for that user by means of the network object's *authorization*, or *access control*, mechanism. In the case of a network file share, for example, an *access control list (ACL)* or *access rights list (ARL)* specifies a list of users and, for each user, whether that user can *Read*, *Write*, *Execute*, or *Delete (*or a combination of these access rights) files on the network drive.

Note that as described in the table above, the mechanism that authenticates the user also authorizes access. However, the access control function[8] is frequently separate from the authentication process: the target service, application, database, or server process typically has its own authorization mechanism.

A Word about Passwords

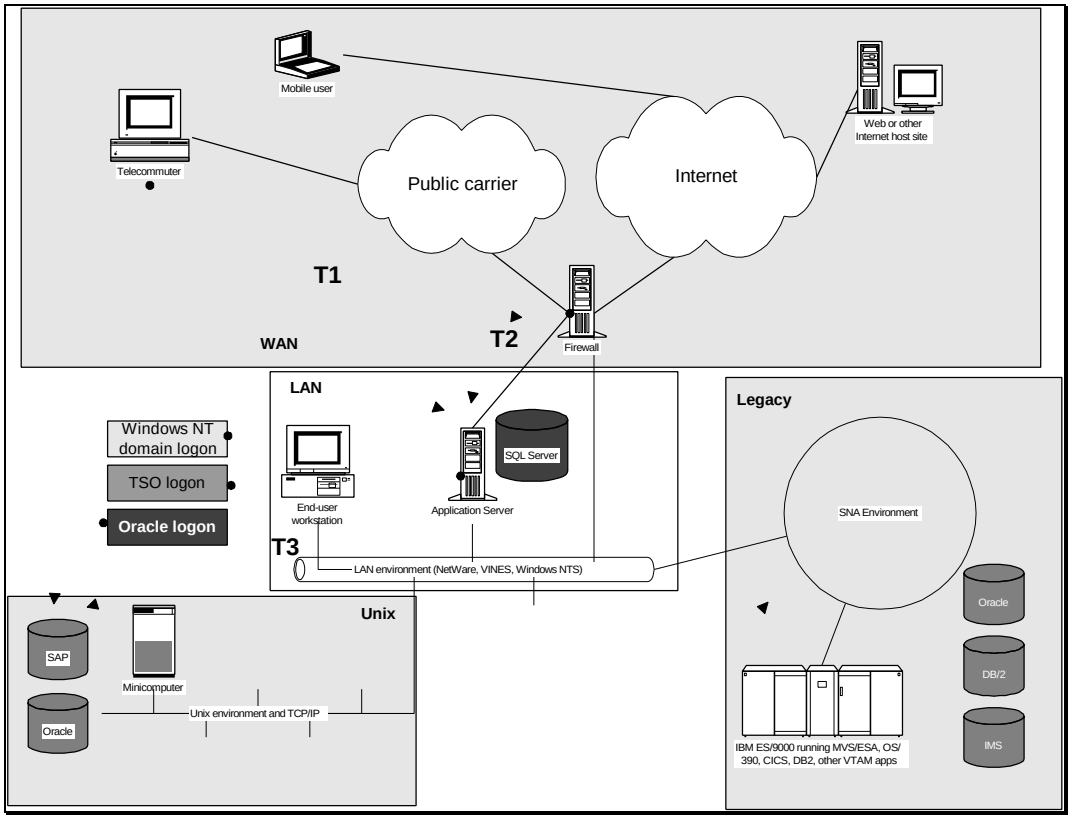
The authentication process is frequently discussed in terms of the user logon and password. However, NAC believes that passwords alone are the least desirable authentication token available, and the industry should move away from relying exclusively on them for authenticating user identity. The chief weakness of passwords is that they are easily compromised, through user abuse, neglect, or mismanagement, and from deliberate attack through spoofing, sniffing or cracking.

[8] Some of the products in the SSO area, such as Axent ESO, H-P Praesidium, and ICL Access Manager, to name just a few, encompass both authentication and authorization features.

Enterprise-wide Authentication

The authentication process described in the context of a ***single*** environment, application, or service looks functionally elegant. However, when one examines the process in the context of today's enterprise-wide, distributed computing environment, amid dozens of other such processes, one sees an altogether different picture. The figure below is a greatly oversimplified view of the environment and only begins to hint at the magnitude of the problem.

Figure 3. The Multi-tiered Heterogeneous Computing Environment



The figure above shows four key areas or environments including the LAN environment; the WAN environment, which today includes the Internet as well as remote dial-in access over the public networks in addition to the traditional private and leased networks; Unix and other mid-range platforms; and the so-called legacy environment typified by the IBM mainframe. This is by no means a complete picture, nor does it begin to show the different client and server platforms and architectures contained in each quadrant. For example, the client operating systems that might be pictured in the LAN environment include AIX, DOS, MacOS, NextStep, OS/2, Solaris, SunOS, Windows 3.x, Windows for Workgroups, Windows NT Workstation, Windows95.

This is the environment that most large organizations are dealing with today. Because no single protocol can be used to communicate between all components in all environments and across every platform, every authentication process runs independently and in parallel with all the other authentication processes in the environment. The result is that users access each of the network operating systems, database systems, and application environments via separate security mechanisms: If one has a dozen different NOS applications or databases to access, one typically has 12 different logons and possibly 12 passwords.

For example, a telecommuter might have a password-protected SecurID to logon to the corporate firewall (T1). Once authenticated by the firewall, he might logon to the local-area network (T2), and then logon separately to Microsoft Mail, Collabra Share, and an Oracle workgroup-level database application. Access to any mainframe-based applications (T3) requires additional and completely separate logon procedures.

The problem of multiple logons doesn't affect only end-users; network or systems administrators must manage all the client and server processes, create every User ID, configure every user profile, give the user his initial password, and perform these administration tasks in all the places necessary for every user.

Conversely, when a user leaves the organization, the administrator must remove all traces of that user from all places in all environments, all applications, all databases, all network operating systems, all legacy systems, and so on. In the example cited above, the administrator — or, more likely, multiple administrators who must coordinate their efforts — would have to configure and manage authentication for this user at the firewall, the local-area network, and in the mainframe environment. The "multiple logon problem" leads to the following issues across an enterprise:

- *Lack of productivity*. The more applications, databases, and services individuals need to do their job, the more time they spend on the logon process. If the authentication mechanism is a password, users must also maintain and manage their passwords, which might include the burdensome task of creating new passwords each month for numerous systems –TSO, CICS, PROFS, Windows NT LAN, LAN-based e-mail system, and so on. (See *Appendix D. "Password Parameters,"* for more information.)

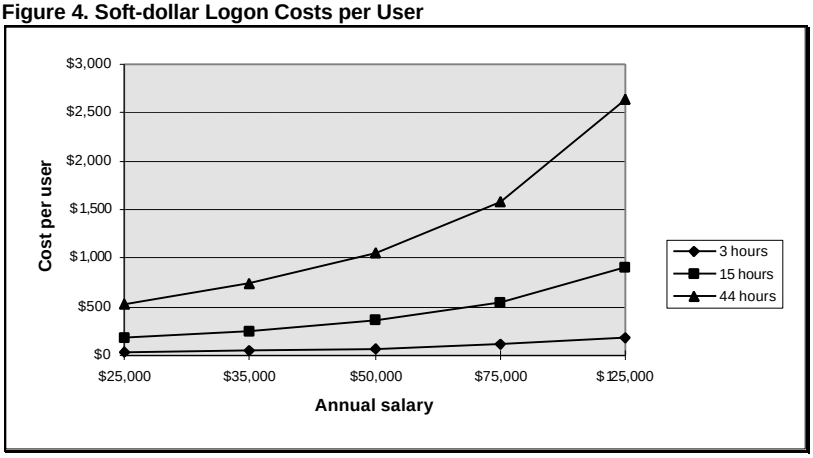
  Statistics for soft-dollar costs — costs in terms of the time an end-user spends on the logon process — range from a low of 3-hours-per year[9] to a high of 44-hours per year[10]. The figure below spreads these estimates across various

[9] Data used by a large U.S. retailer in its business case to support an enterprise-wide single sign-on implementation project.

[10] Figure cited by Bellcore as being the logon time per year for a user with four applications.

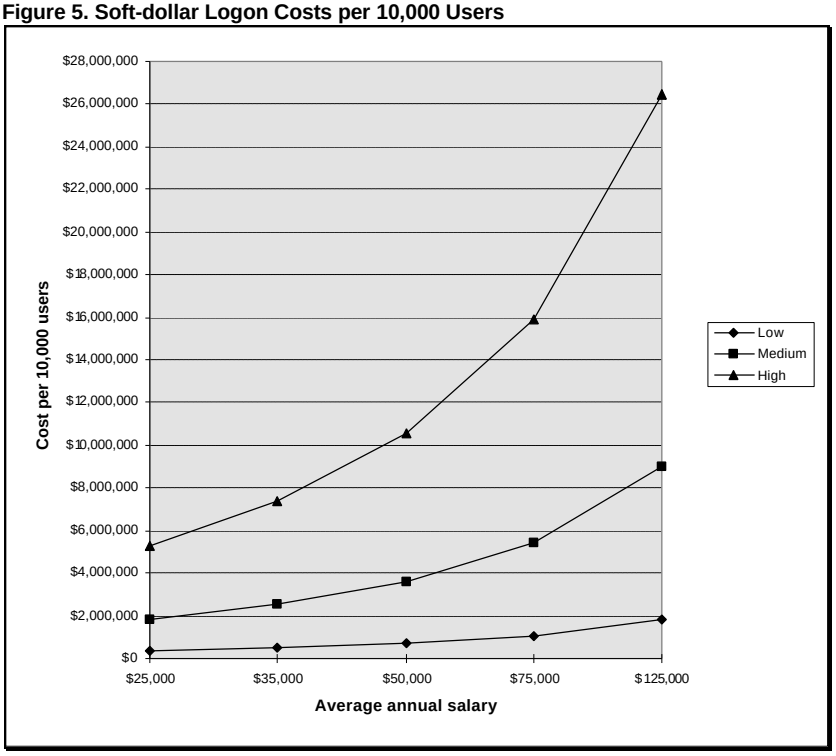
salaries to show the annual cost in terms of lost productivity per user.

Figure 4. Soft-dollar Logon Costs per User



The figure below extends the costs across a mid-sized enterprise and shows that enterprise-wide annual lost productivity amounts to millions of dollars at even the lowest soft-dollar estimates[11].

Figure 5. Soft-dollar Logon Costs per 10,000 Users



[11] Note that automating the logon process merely moves the processing burden to the network or to the back-end process, so it doesn't go away entirely. But by virtue of the fact that machine speed is exponentially faster than human speed, one can still make the argument that there will be performance and productivity gains.

- *Increased potential for compromised security due to lack of convenience*. Because users must keep track of numerous logons and passwords, they frequently write them down — on PostIt® notes, on a list next to their workstation, on the back of business cards. Or they use passwords that are obvious and as easy to guess as they are to remember. In this way end-users become a potential source of compromised security.
- *Increased support costs*. Users who forget their passwords ask the help desk or their system administrator to "reset the password." For example, one help desk[12] reports that 70% of its 90 to 100 calls per day are requests to reset passwords. Time spent resetting passwords impairs the help desk's ability to support other, perhaps more significant, requests.
- *Devaluation of password as a security mechanism[13].* Because of the cavalier manner in which passwords are handled by end-users and the ease with which the help desk can be convinced to reset them, the password has been devalued as an authentication factor, resulting in further risk exposure.
- *Increased administration expense*. Each application, database, and service that has its own authentication mechanism adds to the administration burden in one or more places: at the end-user workstation; at the back-end service, at an intermediary service; or all three places. Synchronizing all UserIDs and passwords across systems, and managing all adds, moves, and changes is an enormous — and expensive — task.
- *Lack of availability due to expired, forgotten, or out-of-synch passwords.* Rather than having a cavalier approach to managing user authentication, some organizations may impose such strict controls over authentication and authorization processes that users don't have timely access to systems. Extra stringent controls and bureaucratic methods may result in lack of availability and productivity.

Thus, the industry's short-sighted approach to I&A — short-sighted in terms of developing I&A as part of core NOS, application, or database functionality, rather than taking a services-oriented approach that could leverage existing mechanisms — is both costly and risky: costly, in the time it takes for end-users to logon, administrators to administer, help desk staff to reset passwords, and so on; risky because multiple logons encourage inappropriate behavior on the part of end-users and because each logon is an additional point to replace or administer.

---

[12] The help desk for a custom line-of-business application at a west coast utility.

[13] In general, NAC believes the industry should move away from the password as an authentication mechanism. See *A Word about Passwords* in the next section for further discussion.

RISK AND COST FRAMEWORK

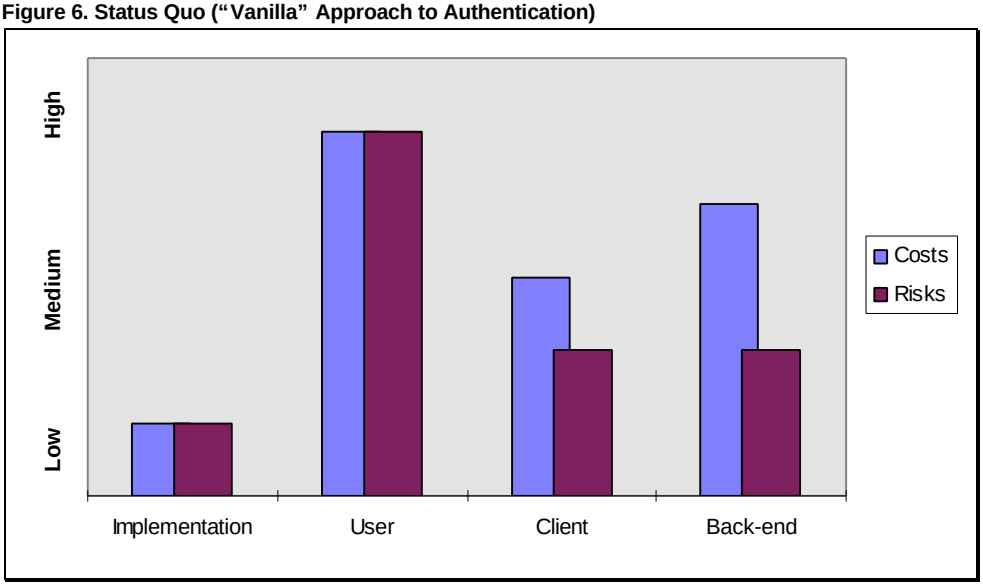
To understand the costs and risks associated with multiple logons (what we are calling the status-quo, or the "vanilla" approach to authentication) and with various alternative approaches to minimizing the number of logons, NAC developed a simple framework for assigning risks and costs to the implementation of a minimal logon strategy and to the user, client, and back-end components of the strategy.

Note that the graphs depict relative, not absolute values. For example, the bars in *Figure 6. Status Quo* represent low, medium, or high costs and risks when compared to the three strategies for reducing logons. For any strategy, including the status quo, or vanilla approach, costs and risks are defined as follows.

Risks refer to security risks associated with implementation, users, clients, and the back-end processes, as defined below:

- ***Implementation risk*** is the amount of additional exposure to security risks caused by implementing a new logon strategy. For example, implementing a new authentication method might require temporarily resetting all users' passwords to a known value.
- ***User risk*** is any increased security exposure that might result in terms of the end-user. For example, in the context of the vanilla approach, where there's an *n*:1 relationship between total applications, systems, databases, and services and the end-user, the user risk is relatively high: the more problematic the logon experience is to the end-user, the greater the tendency for the end-user to try to simplify life by writing down logon IDs and passwords, for example.
- ***Client risk*** represents the amount of exposure in terms of the physical workstation. For example storing passwords on an insecure platform, such as a typical DOS/Windows or Macintosh workstation, increases the risk that the passwords may be compromised.
- ***Back-end risk*** represents the security risk to the existing infrastructure services, systems, and applications that results when a method alters, affects, or otherwise manipulates another back-end mechanism. For example, a system with strictly enforceable password controls (such as length, content, uniqueness) would be weakened when front-ended by a system with less stringent controls.

Figure 6. Status Quo ("Vanilla" Approach to Authentication)



Costs refer to both hard- and soft-dollar costs associated with each component. It is particularly important to focus on the total cost of ownership (TCO), which includes the lifetime costs associated with a strategy or technology. As with most computer systems, the on-going costs of ownership will eventually dwarf the initial acquisition costs.

- ***Implementation costs*** include all initial costs of implementing a strategy, including roll-out, training, modifying any applications as needed — costs for everything the organization must do to adopt the method in question.
- ***User costs*** include on-going costs in terms of user productivity, not only in terms of time spent interacting with authentication processes, but also in terms of users' ability to easily and reliably accomplish their work.
- ***Client costs*** include all administrative and management costs associated with the client workstation; for example, the costs of maintaining files or applications on a large number of workstations.
- ***Back-end costs*** include all costs associated with back-end servers, services, and applications. Again, these are typically administrative and management costs, but they may also include additional costs of ownership of hardware and software.

In the next section we examine three general approaches to minimizing the overall number of logons. Each approach has its own set of costs and risks to consider. There are three additional factors that aren't captured in the charts but which must be taken into account and which depend on the organization rather than the approach:

- Total users. The number of users in an organization will skew the relative values of user risk and cost and client risk and cost. For example, an

organization with only 500 users accept an increase in client costs when balanced against the back-end cost of purchasing and deploying a new server while an organization with 10,000 users would make the opposite choice.

- Availability. A specific approach may not be immediately available in the context of a particular organization's mix of platforms, network operating systems, and applications, so the relative costs and risks of the approach may be immaterial.

- Practicality. A specific approach may not be practical or realistic. For example, an approach which includes abandoning legacy applications isn't a viable choice for most organizations.

Minimal Logon Strategies

There are three basic approaches to reducing the number of logons for end-users:

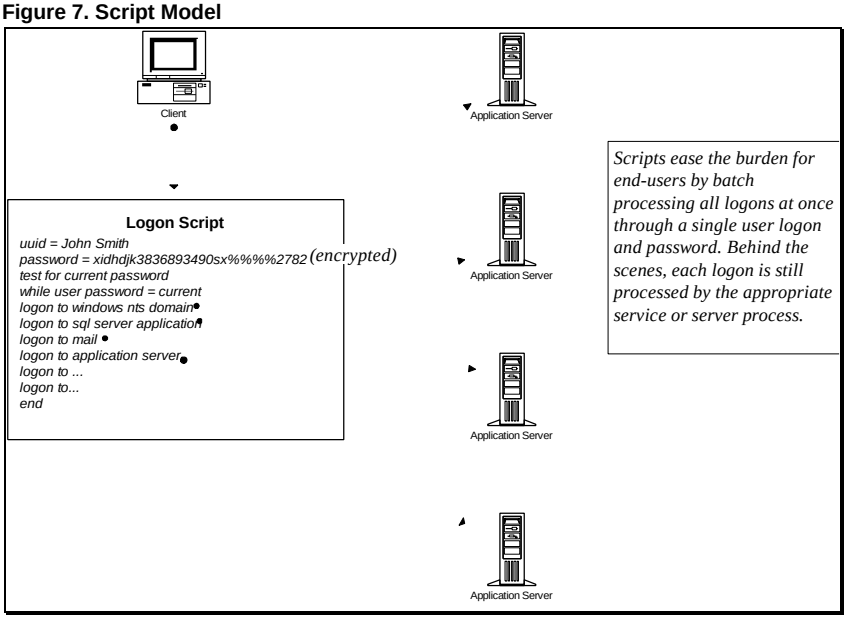
- Scripting
- Middleware
- Broker

As you'll see from the discussions below, only the broker model represents a true "authentication service" in which there is only one instance of the user ID and password (or other token) in the enterprise. The other two approaches, scripting and middleware, provide the end-user with a single logon process but multiple instances of the user ID and password remain distributed throughout the enterprise LAN/WAN — for example, in the NOS security service, in a mainframe authentication table, in the user tables of various applications and database management systems.

SCRIPTING MODEL

One of the easiest ways to give end-users a single logon (or, more strictly speaking, the illusion of a single logon) is through the use of a script. A script programmatically processes, either on-demand or at startup, all the user logons, just as a DOS batch file allows a single command to cause the execution of multiple DOS commands. Scripting is common in the mainframe environment, where VTAM session managers consolidate session logons for TSO, CICS, and TMON under a single logon. Another common example is the password caching feature in Microsoft Windows for Workgroups.

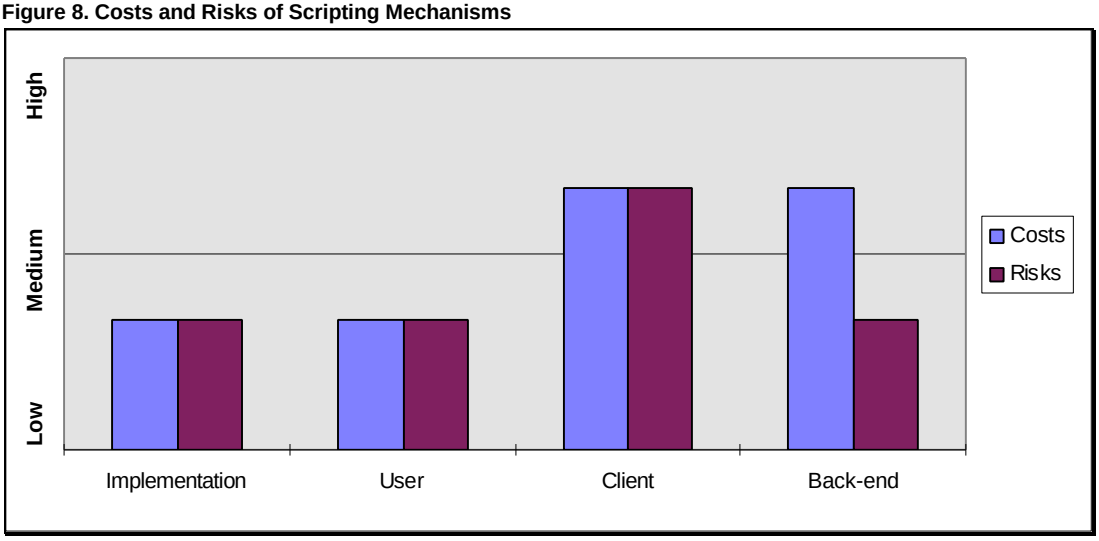
Figure 7. Script Model



Scripts ease the burden for end-users by processing all logons through a single user logon and password. By reducing the number of logons for end-users, scripts can decrease the security risk at the end-user because end-users that have only a single logon password don't have as strong a need to write it down. Another advantage of scripting is that implementation costs are low, and it's relatively quick to deploy — typically, no changes need be made to existing applications to implement scripts.

Because the script simply funnels all existing logons into one and doesn't tamper with the underlying logon mechanisms, scripts can be thought of as "fault-tolerant" in the sense that if the script fails for some reason, users can still logon using the native mechanisms of each application — the scripts supplement but don't replace the logon processes.

Figure 8. Costs and Risks of Scripting Mechanisms



That said, scripting has many disadvantages. Although end-users may have only a single logon to deal with, behind the scenes, each logon is still processed by the appropriate service or server process. What this means is that all user logons and passwords (or other authentication tokens) still must be created and administered with the native client and server mechanisms, and, additionally, that the script files that tie to the two together must be created and maintained. As the number of client and server processes that will be consolidated into the script grows, maintaining scripts throughout an enterprise becomes a costly and burdensome administrative task which is also error-prone as administrators attempt to keep up with version changes, different releases, different vendor products, and nested scripts in a fully distributed logon environment.

Thus, the cost of script maintenance shows up in the cost-risk table as a relatively high bar for client and back-end. In addition, because the script is often tied to the end-user's workstation, the risk at the client is also high — additional security

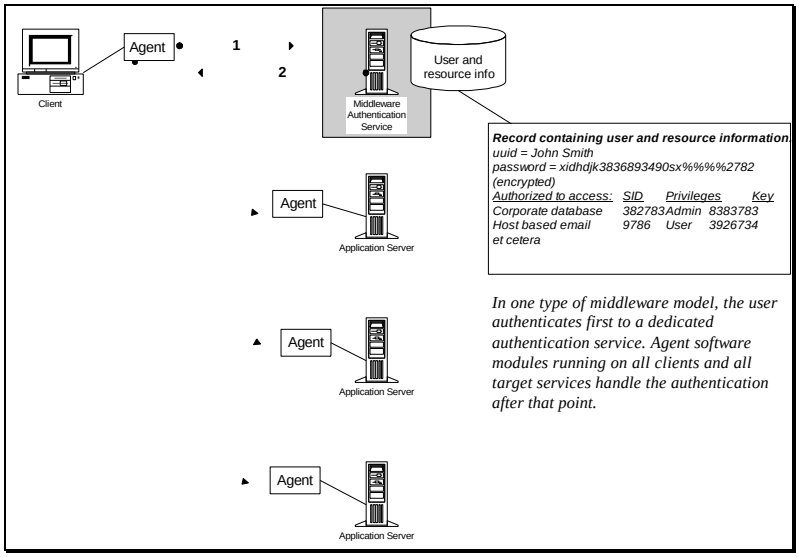
measures may need to be added to the client workstations to ensure that only the right person executes the logon script. Furthermore, the scripting method does nothing to improve or enhance the existing security, for example, if the base application uses an unencrypted telnet or 3270 logon, that's what the logon script will provide.

MIDDLEWARE MODEL

Similar in function to a gateway, a middleware[14] authentication service manages authentication for all clients by communicating with all services and applications in the environment. In order to do this, the middleware service must be able to communicate with each requested service using the target service's API and protocol — just as any gateway must support all targets to which it is connected.

In the example figure below, client software modules and target server software modules (called Agents, in the Axent product model) interact with each other to authenticate users. Before this can happen, though, the client first authenticates to the Authentication Service and receives a list of resources that it can use, with the appropriate access rights. End-user agents communicate with the authentication server, which contains definitions of users, secured resources, and policy information that provides connection between the users and secured resources.

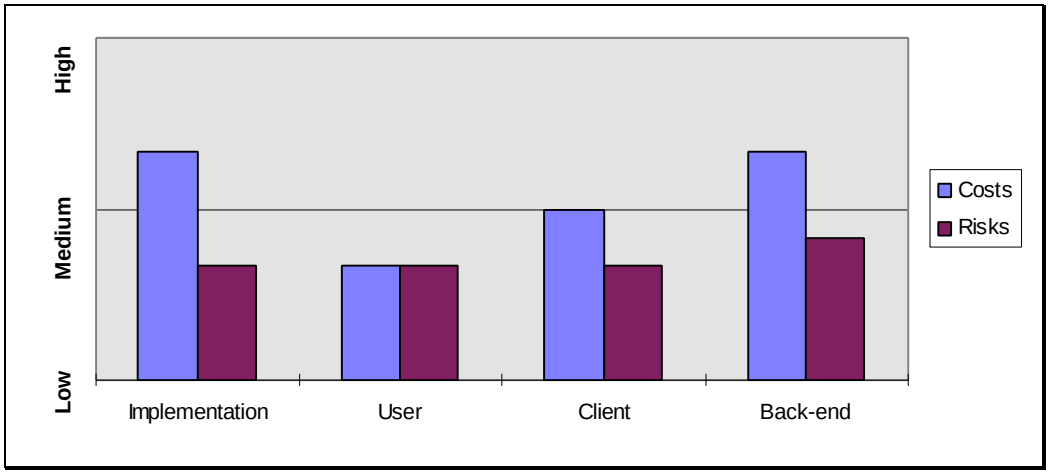
Figure 9. Middleware Model



[14] NAC's description of a middleware authentication service is similar to what some analysts (such as Meta Group) describe as the broker model. However, because the scheme typically relies on distributed software modules (agents), and because the broker model as NAC describes it depends on a common protocol and API (and the middleware model does not), it seems more appropriate to refer to the model as middleware. Nonetheless, be aware that these models and their definitions are subject to different interpretations.

The middleware model as implemented in products such as Axent's OmniGuard provides a centralized approach that can be easy to manage. This approach has the potential to improve security because the user has only one password, thus the risk in terms of the end-user is lower. As with the scripting method, the middleware approach doesn't require changing server code; nonetheless, implementation costs are high because of the need to install and configure the initial agent software throughout the enterprise.

Figure 10. Costs and Risks of Middleware



However, it does require an additional server. In addition, the success of the scheme depends upon the vendor's ability to deliver agents in all the flavors needed for a particular enterprise.

In variations on this model, scaling may be an issue if the middleware server is the primary authentication provider for all other servers, services, and applications on the network. Synchronization of all user information among middleware servers and application platforms may also be a problem.

Finally, the middleware server may present a single point of failure which may not allow workaround.

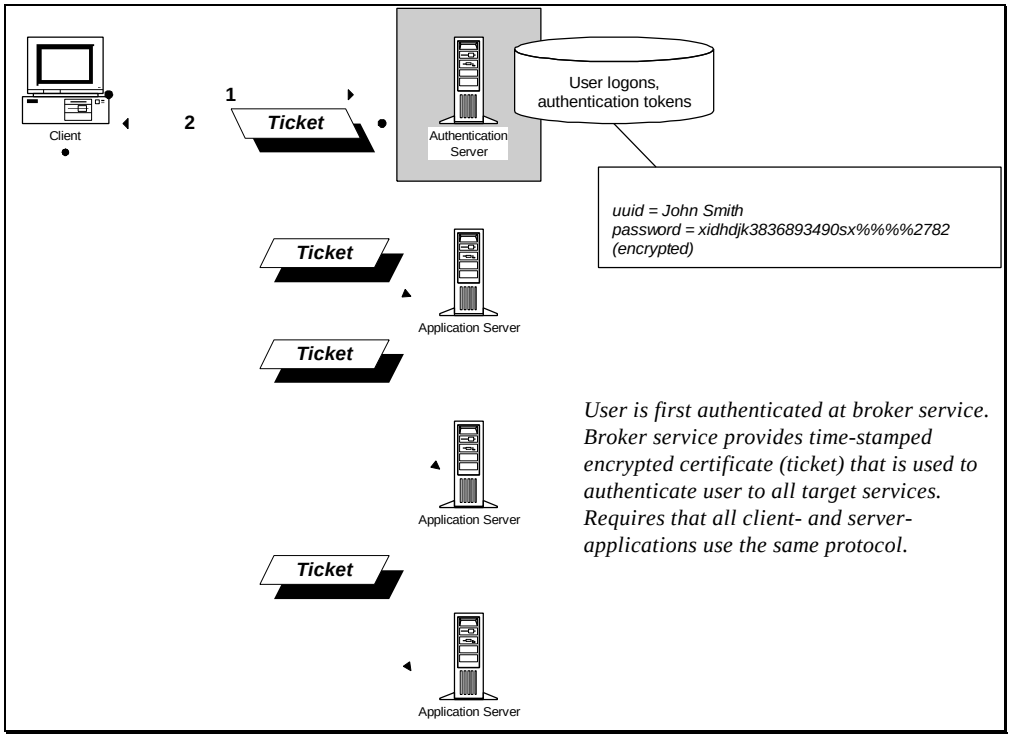
BROKER MODEL

The broker model comes closest to NAC's vision of an interoperable authentication service, but again the broker model isn't necessarily interoperable across all platforms. In the broker model, all client and server processes authenticate themselves to a physically secure authentication service. The service, in turn, issues a time-stamped credential, or "ticket." The ticket is used later when the user attempts to access another service. Although the process of authentication

is invisible to users after the first one, behind the scenes the credential issued to the workstation authenticates the user to each target service.

The broker model can be implemented as a “push” or “pull’ mechanism. In the push model, the client first communicates with the Authentication Server to get a ticket for the desired (target) server; in the pull model, the client first contacts the target server (or service, application, whatever) which in turn contacts the Authentication Server for the necessary ticket.

Figure 11. Broker Model



Kerberos, an authentication protocol that was devised as part of MIT’s project Athena[15], is a widely used authentication service. The protocol has been implemented by several vendors, including CyberSafe, and has also been adopted by the OSF for its DCE, although the DCE version of Kerberos is at this point not compatible with MIT Kerberos[16]; current implementations of Kerberos include both versions 4 and 5.

Other authentication services that can be loosely said to fit this model include products such as Bull Access Master Service, ICL/Access Manager, and

---

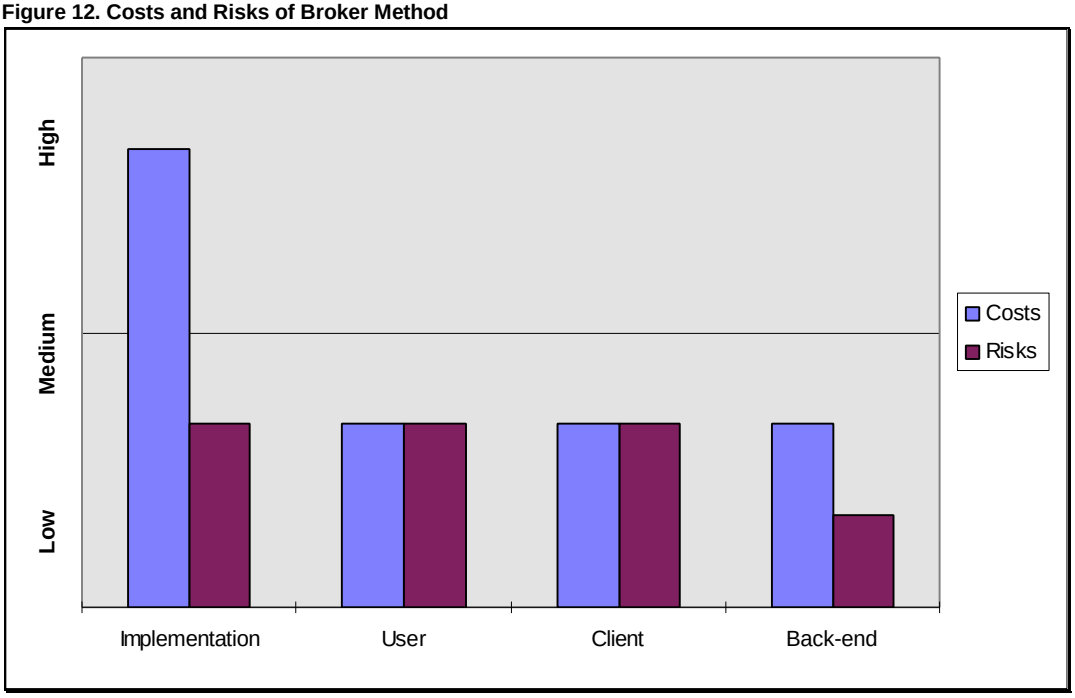
[15] An experimental distributed computing environment begun in the early 1980s at MIT in conjunction with Digital Equipment and IBM.

[16] See *Kerberos as a Functional Model* in Appendix A. for more information.

SESAME. An authentication framework, X.509, is specified as part of the X.500 directory service recommendation. The NetWare 4.x directory service includes a ticket-based authentication process that is conceptually similar to Kerberos in that it issues tickets which may be used for SSO to applications supported in the NetWare environment.

Of the three approaches, the broker model is the most elegant architecturally. By providing a single authentication service the broker model, in theory, can provide a single source for authentication across all applications, network operating systems, platforms, and services - presuming that all these network resources are compatible with the authentication service. It's also potentially the most secure in that it can provide mutual, strong authentication and time-stamped tickets that resist replay attacks.

Figure 12. Costs and Risks of Broker Method



However, as the least mature alternative (in an admittedly immature marketplace), the broker model is also the most difficult (and therefore costly) to implement since all applications, network operating systems, platforms, and services must use the same protocol. All new applications must be written, and all existing applications must be rewritten, to use the service. This means they must all use the same protocol, which in itself presents a possible long-term financial risk — if the selected protocol fails to achieve market share and becomes obsolete, the organization will be forced to accept a huge loss in terms of migrating applications away from the obsolete approach.

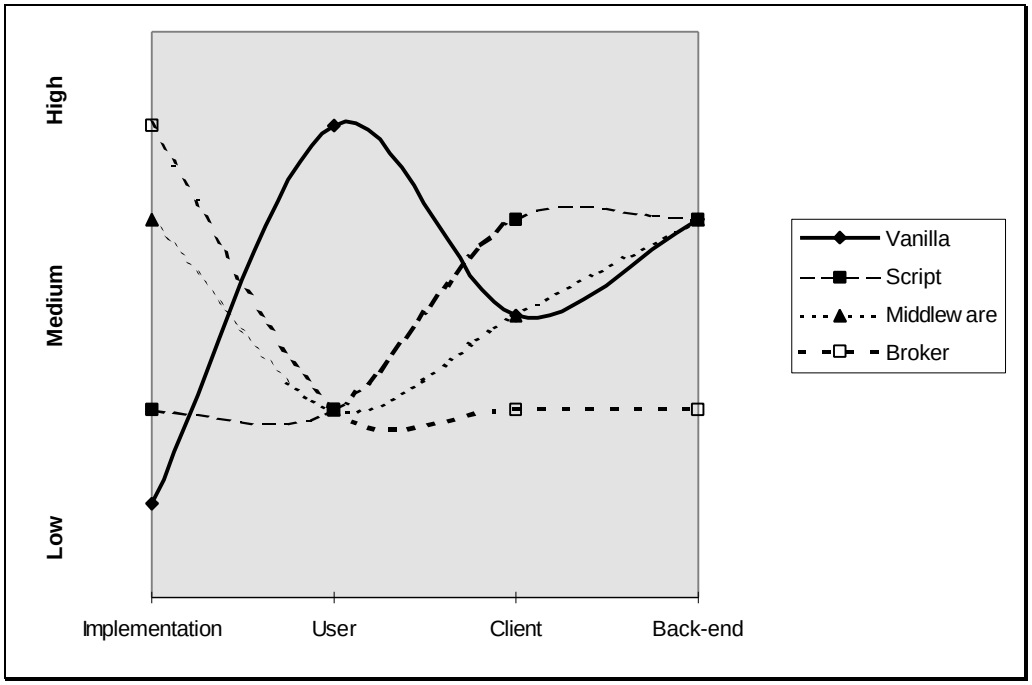
There is some potential security risk in that broker service is itself a single point of failure. If the service is down, or if performance is poor, all applications and services can become unavailable. On the other hand, the fact of a single instance of the user id and password (or other authentication token) in the enterprise greatly reduces the human and policy errors which plague systems with multiple security systems.

Conclusion

Each minimal logon strategy has both advantages and disadvantages, the relative importance of each of which are affected by factors such as the number of users in an organization, the mix of new and legacy applications, and the sense of urgency around reducing the costs and risks associated with multiple logons. For example, even if an organization is willing to modify all its existing applications to move to the broker model, the time it would take to do so may be unacceptable.

The two figures below present side-by-side comparisons of the costs and risks respectively of the status quo and each of the three major approaches.

Figure 12. Costs of Methods Compared



In most environments the cost of the status quo is associated with the user and with administering multiple back-end services.

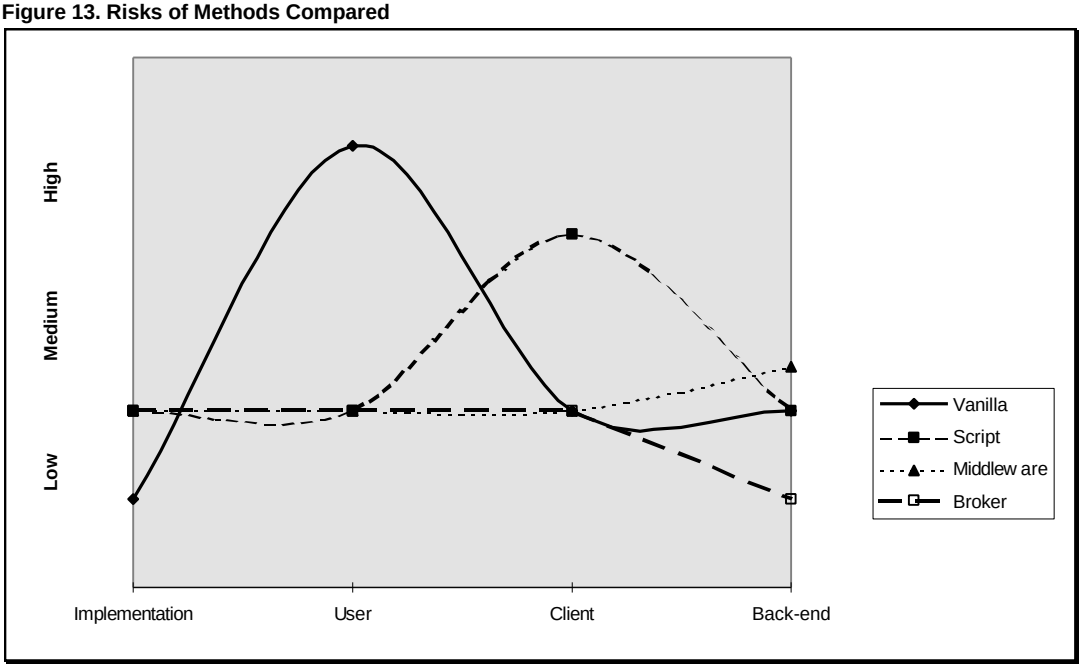
The scripting model shifts the cost from the end-user to the administration of the client. The implementation cost is driven by the product of the number of clients and the number of applications. This approach will be favored by organizations where productivity is critical and there is not a sufficiently long financial view to appreciate the long-term costs of maintaining the client software.

With its low implementation costs and its potential to lower risk at the end-user, scripting is also a viable short-term solution for sites with large investments in legacy applications, or with large investments in disparate applications across numerous platforms; such diversity might preclude a middleware or brokered implementation due to lack of support for the range of platforms or applications.

The middleware model reduces the costs at the user and the client but with greater implementation and back-end costs. This shifting of the costs away from the user/client makes this approach more financially attractive to organizations with very large numbers of users. Of course, the viability of the middleware approach in any given organization will depend on the completeness of the selected vendor's product line in terms of number of platforms, both client and servers, supported by the middleware technology.

The broker model requires a long-term commitment to recover the very high implementation costs associated with modifying all applications. Once implemented, however it has the lowest on-going cost of ownership of any of the approaches. Sites with a minimal investment in legacy applications are the best ones to consider this approach.

Figure 13. Risks of Methods Compared



The vanilla (or status quo) associates most of the risk with the end-users . This is generally not acceptable since the user is the least controllable element of the authentication process.

The scripting approach essentially moves the risk from the end-user to the client. Scripting does nothing to improve the quality of the authentication token (e.g., plaintext transmission of passwords) and it requires the token to be stored in the user's environment. In many cases this is on an unsecured workstation. There is some implementation risk associated with scripting, for example, because of the potential need to bulk transfer large numbers of passwords to the client.

The middleware approach introduces some additional risk at the back-end by adding a point where security can be compromised. In general, this more than made up for by the reduction in risk at both the user and the client. The middleware solution also allows the enforcement of a single authentication policy and provides the opportunity to improve the quality and secrecy of the authentication token (password or otherwise). As with the scripting method some risk is associated with the implementation itself. There is also the risk that the authentication in the middleware component may not be as rigorous as some of the systems it is front-ending.

The broker model is almost certainly the most secure approach. It is the only one which allows a single instance of the user/token pair. This virtually eliminates the problems of breach of security based on errors in administration, such as not removing visitor or contractor accounts.

For most, if not all, organizations the optimal path will involve a combination of two or more of these approaches. For example a combination of a broker authentication service directly supporting new applications and providing credentials to a middleware component which front-ends legacy systems. Organizations may also be forced into the unfortunate position of supporting multiple implementations of a single approach. One NAC company, for example, is in the process of deploying both DCE and Kerberos security mechanisms to support two different sets of applications from different vendors.

One area where some relief may be forthcoming is in the client APIs. The GSS-API, first published in 1993, is beginning to gain increasing mindshare. Although a common API doesn't address the problems of developing a single security service or of dealing with legacy applications, it does offer increased opportunity for interoperability between future clients and servers. This would allow new applications to leverage a variety of security services and would make an organization's choice of a particular authentication protocol or service less critical.

NAC Recommendations

General Recommendations

In the short and medium term, IT organizations should focus on reducing the number of different authentication methods rather than waiting for a "silver bullet" to provide an enterprise-wide single logon. Purchasing decisions should favor vendors whose products will interface with existing network operating systems or other dominant services or APIs. For example, if you're in a Kerberos environment and you need to install a new payroll application, limit your choices to those that can use the Kerberos service for authentication and don't rely on proprietary protocols that they alone use. This approach will begin to reduce the current and future problems associated with multiple logons, and at the same time will decrease the costs associated with migrating to a single authentication service if and when a viable standard emerges.

Internally developed applications should both use existing authentication mechanisms to reduce the proliferation of logons, and should encapsulate authentication processes (as well as any interaction with other infrastructure services) to reduce the cost of migration once a dominant standard(s) emerges.

In the long run, end-user companies should be participating in both the "official" standards organizations and in the standard-setting process in the market place. This serves consumers both by influencing vendors to develop products that actually meet the consumers' needs and by keeping their own IT strategies aligned to the realities of the market.

Recommendations to Infrastructure Vendors

- Provide support for the GSS-API and other platform-independent mechanisms. This will allow your services to be used by an increasing number of applications, which in turn makes your services a better investment for your customers.

- Publish and market all APIs (not just security, but directory, messaging, and so on) so that developers can and will write to infrastructure services instead of coding their own implementations in piece-meal fashion.

- Use your own APIs (meaning the APIs you've published), not backdoor functionality. This will improve the quality of the APIs and make a convincing case that they provide all the functionality required to build production applications. This is particularly important for vendors who have strong market presence in both the application and infrastructure areas, e.g., Microsoft, Lotus/IBM, Novell/WordPerfect.

Recommendations to Application Vendors

- Don't use proprietary security protocols to differentiate your products. Build and package applications to use a variety of authentication protocols. As customers increasingly understand the costs of non-interoperable solutions they will become less willing to accept products which lock them in to supporting multiple competing infrastructure components

Recommendations to NAC Members and Other Consumers

- Apply economic pressure on vendors to implement the most promising emerging security standards. RFPs and purchase orders should include the requirement to use infrastructure authentication services.

- Include full life-cycle costs of infrastructure (including security) in RFPs and purchase decisions – that is, don't overlook the long-term costs of selecting proprietary authentication implementations.

- Address compelling issues now with tactical solutions that don't lead to lock in. For example, in choosing among three different databases, select the one that offers support for the widest range of authentication methods.

- Quantify the authentication problem in terms of your own organization. For example, if your help desk doesn't currently collect and publish statistics on the number of password resets per month and questions about logon problems, have them begin doing so. In order to make your own case, you'll need to analyze your own costs.

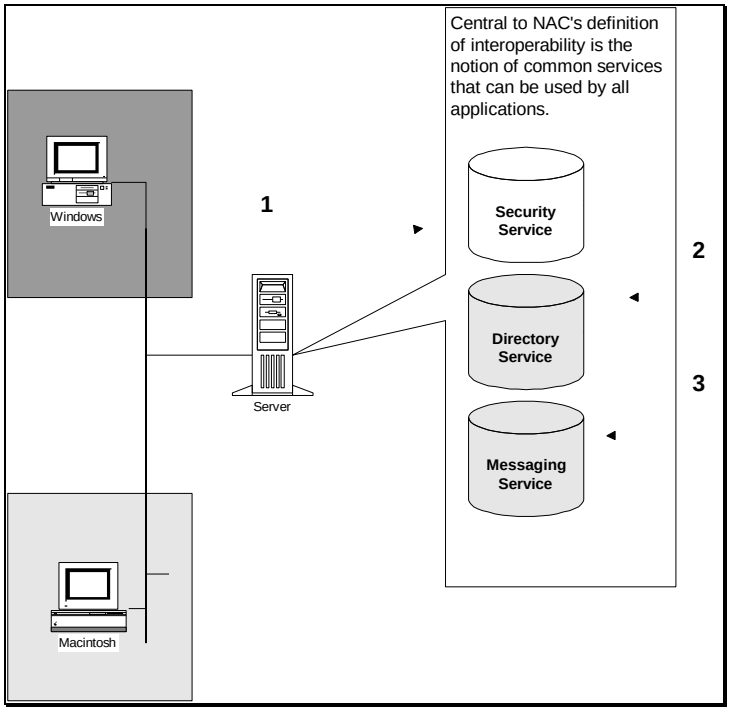
Appendix A. NAC Generic Authentication Services Model

Rather than authenticate users on an application-by-application basis, an *authentication service* can provide a common mechanism to validate user identity, ideally across all network operating systems, applications, and services. An authentication service has a database of user accounts, passwords, and information about services. When a user logs on, the request is sent to the authentication service, which issues a ticket or some other such "credential" after authenticating the user. The credential then authenticates the user to all services and applications for a specific period of time.

Basic functions available through the API and supported by the protocol should include:

- initializing a security context
- authenticating client to server
- authenticating server to client
- authenticating per-message data origin (thus providing non-repudiation)
- authenticating per-message data integrity

Figure A. NAC Generic Authentication Service Model



The common services interoperate with each other as well as with the network applications that need them. In the figure above:

the user logs onto the enterprise authentication service;

❷ the security service and directory service interact to validate the user by comparing

credentials held in the directory service; and

the message and directory services work together to resolve addresses for messages transmitted by users (and processes).

The NAC Generic Authentication Service model is NAC's vision of a single, enterprise-wide authentication service. The presumption is that all applications, databases, network operating systems, and services would use this service (and this service alone) to authenticate user identity. An authentication service designed as an infrastructure-level service capable of authenticating all users of all applications throughout the enterprise must meet the following business and functional requirements.

BUSINESS REQUIREMENTS

The authentication service must be:

- Cost-effective (and easy) to manage and administer.
- Scaleable to an enterprise-wide scope.
- Capable of secure communications and store.
- Mechanism independent.
- Capable of supporting legacy applications.
- Extensible. Other security capabilities and functions that meet the specific needs of a given organization must be able to be added on to the authentication service without affecting interoperability.
- Portable (both capable of and likely to be acceptable across all platforms)
- Easy to use from an end-user perspective.
- Fraud-proof.
- Highly available.

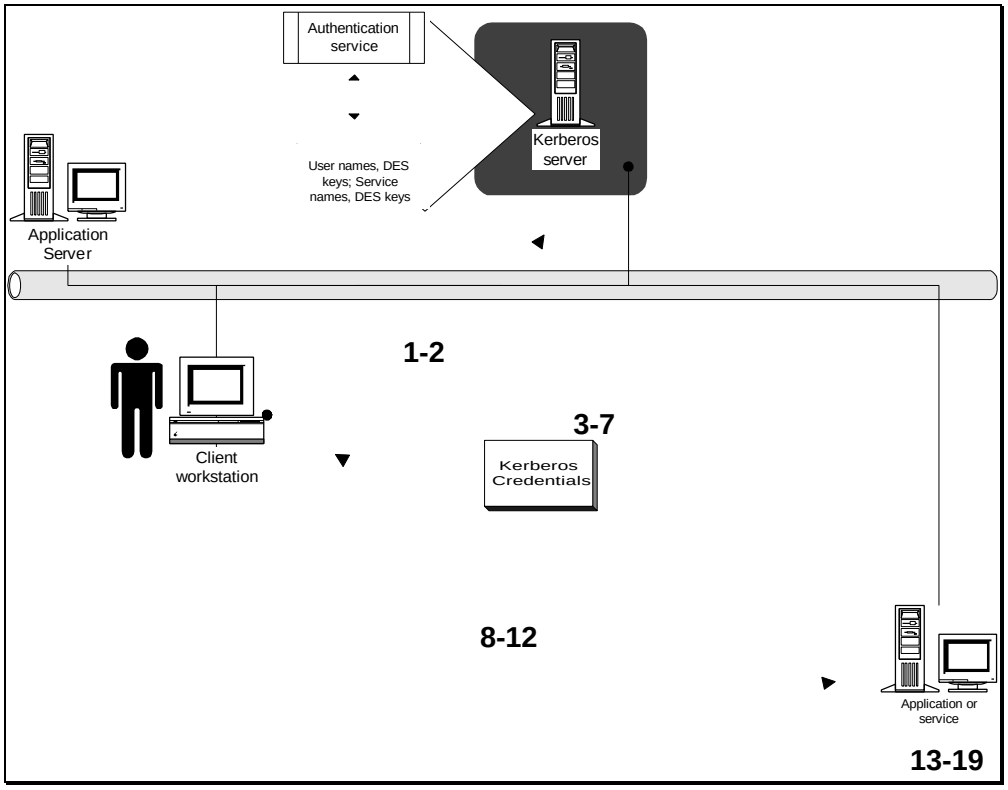
FUNCTIONAL REQUIREMENTS

The authentication service must be:

- Designed as the server portion of a client-server distributed process in which the functions of the authentication service itself are exposed through an application programming interface.
- Capable of being used by other services as well as other clients.
- Capable of re-verifying users to a subsequent process (in the sense of a child process spawned by a parent process).
- Capable of implementing and enforcing security policies regarding password parameters (password aging, alpha-numeric characters, character length, limitations, non-dictionary passwords, and so on).
- Mechanism independent.
- Capable of interfacing with user activity and administration activity logs.
- Capable of assigning and managing encryption keys, or interfacing with a service that assigns and manages encryption keys.
- Capable of authorizing user activities after they have been authenticated, or interfacing with a service that manages user authorizations.

KERBEROS AS A FUNCTIONAL MODEL

Kerberos is one model that meets some (although not all) of NAC's requirements. NAC is not endorsing Kerberos — it has several shortcomings, including the fact that it doesn't provide non-repudiation services — but as a process model, Kerberos has some merit and is worth understanding. Kerberos is a secret-key based cryptographic authentication protocol; the protocol specification describes the action between an end-user (subject), a Kerberos authentication server, and the target service (object).



End-user and Client Application	Kerberos Authentication Server	Target Service
1. User enters name and name of requested service into the client application.	3. Checks database to verify User's access rights.	13. Receives Ticket + Authenticator.
2. Kerberos client software sends request to Kerberos server.	4. Generates random Session Key.	14. Decrypts both.
8. Receives ticket back from Kerberos server and prompts user for password.	5. Creates Ticket.	15. Verifies that it is the Service identified in the Ticket.
9. Translates password to User's secret key.	6. Encypts Ticket + Session Key with User's key.	16. Checks Timestamp to make sure the ticket is still valid.
10. Uses Secret key to decrypt Ticket and Session key.	7. Encrypts both with requested service key to make credentials and sends back to requesting workstation.	17. Removes Session Key and uses session Key to decrypt Authenticator.
11. Builds Authenticator from properly decrypted Ticket + Session key.		18. Validates client name in Ticket and Authenticator.
12. Sends copy of Ticket + Authenticator to requested service.		19. Authenticates user.

Appendix B. Product Directory

Given the dynamic nature of Internet web publishing, some of the information may not be current by the time you read this.

Single Sign-on Mechanisms

Vendor and Product	Web Address
Axent OmniGuard/ESO Enterprise SignOn)	http://www.axent.com/
Bellcore One-Pass System	http://www.bellcore.com/
Groupe Bull Integrated Systems/Access Master Service	http://www.bull.com/
CA-Unicenter/Single Sign-on	http://www.cai.com/products/dsm/sca.htm
CKS MyNet	http://www.cksweb.com/
CyberSafe Challenger	http://www.cybersafe.com
Cylink	http://www.cylink.com/
DynaSoft BoKS	http://www.dynas.se/
IBM Distributed Security Manager (DSM/MVS)	http://www.ibm.com
IBM NetSP Secured Logon Coordinator (SLC)	http://www.ibm.com
IBM RACF	http://www.ibm.com
ICL Enterprises/Access Manager	http://www.icl.com
Mergent SSO/DACS	http://www.mergent.com
Millennium	http://www.millenniumcc.com/
New Dimension Software Enterprise Security	http://www.ndsoft.com
Nortel Entrust	http://www.nortel.com/entrust/
Open Horizon Connections	http://www.openhorizon.com/
Schumann AG SAM	http://www.schumann-ag.de/
Symantec Secure Access	http://www.symantec.com/
Uti-maco Logon Guard	http://www.utimaco.com

Authorization Mechanisms

In the SSO literature, many products are grouped under the catch-all category "Authentication and Access." The following products can interact with many of the Authentication mechanisms in the first table above.

Vendor and Product	Web Address
CA-TopSecret	http://www.cai.com/
HP Praesidium	http://www.hp.com
ICL Enterprises/Access Manager	http://www.icl.com
Memco SeOS	http://www.memco.com
Uti-maco Safe Guard Easy for Windows95	http://www.utimaco.com

Operating Systems, Network Operating Systems

Vendor and Product	Web Address
Banyan VINES	http://www.banyan.com/
Digital	http://www.digital.com/
Hewlett-Packard (HP/UX)	http://www.hp.com/
IBM (SNA Environment)	http://www.ibm.com/
IBM/Lotus	http://www.lotus.com
Microsoft Windows NT	http://www.microsoft.com
Novell NetWare	http://www.novell.com
Sun Microsystems	http://www.sun.com

RDBMS (Relational Database Management Systems)

Vendor and Product	Web Address
Informix	http://www.informix.com
Microsoft SQL Server	http://www.microsoft.com
Oracle Oracle 7.x	http://www.oracle.com/
SAP R/3 System	http://www.sap.com
Sybase	http://www.sybase.com

Appendix C. Summary Table of Pros and Cons for Minimal Logon Strategies

	Pros	Cons
Scripting	• Fast deployment. • Minimal change to applications (client or server). • No single point of failure. Failure of the mechanism doesn't preclude logon, as some of the other methods do — if the logon script fails, the user can still logon individually.	• Difficult to maintain over time due to version-control problems (different brands, different versions, different releases). • Expensive to manage. Doesn't eliminate multiple logons, simply consolidates them for the end-user. Administrators must still manage fully distributed logon environment. • Expensive to maintain due to high administrative overhead. Scripts must be written, distributed, and maintained in multiple places. Nested scripts pose a major maintenance problem. • Increases risk at the client workstation because the script information is tied to the workstation. Therefore, forces the need for security at th PC workstation
Middleware	• A centralized approach that can be easy to manage. • Potential to improve security because user has only one password. Non-encrypted passwords can be kept off most of the network. • Doesn't require changing server code.	• Requires additional server. • Scaling may be an issue if the middleware server is the primary authentication provider for all other servers, services, and application on the network. Synchronization of all user information among middleware servers and application platforms may also be a problem. • Potentially aA single point of failure (depends on design of specific implementation) which may not allow workaround.
Broker	• Most elegant architecture of the three approaches: the broker model provides a single authentication service which, in theory, can provide a single source for authentication across all applications, network operating systems, platforms, and services. • Most secure. Provides mutual, strong authentication and time-stamped tickets that circumvent replay.	• Most difficult to implement since all applications, network operating systems, platforms, and services must use the same protocol. All applications must be written to use the service; existing and legacy applications must be rewritten to use. • Most potential financial risk in the long-term if selected mechanism fails to achieve market share; potential obsolescence. • Most potential security risk in that broker service is itself a potential single point of failure. If the service is down, there is no workaround. • Least mature.

Appendix D. Password Parameters

The table below is a sample list of some operating systems, network operating systems, databases, and database tools. Tl might have a table listing dozens if not hundreds of implementations. In addition, not all the variables are shown. Settings expiration, restrictions for ID and similar passwords, and many other password parameters are different among products

Lege
●
❍
∞

Environment	Minimum Length	Minimum Age	Maximum Age	Allowed characters	Repeat Characters	Required characters?	Case-sensitive	User Can Change?	History	Grace Period	Force Change Initial Value	Maxim Failure
Access Builder	None	∞	None	Alphanumeric	∞	None	●	❍ [1]	None	❍	na	6 teln 1 Wi
Banyan VINES	0-15 chars	None [3]	1-52 weeks [4]	Alphanumeric, spaces, punctuation	∞	None	●	●/❍	10	None/ ∞	●/❍	3
Braintree (Oracle)	35072	1-999 hours	1-99 days [4]	Alphanumeric, spaces (not first), punctuation	Subject to restricted word list	First character cannot be a space; can require both letters and non-letters to be valid	❍	●	0-9	None	❍ [5]	10
Digital VMS	0-32	∞	0-99	Alphanumeric, $, underscore	∞	None	❍	●/❍	0-2000	None/ ∞	●/❍	1-9 o
IVR/PIN	1-12 chars	na	∞	Numeric only	∞	All numbers	na	●	None	na	❍ [13]	∞

Environment	Minimum Length	Minimum Age	Maximum Age	Allowed characters	Repeat Characters	Required characters?	Case-sensitive	User Can Change?	History	Grace Period	Force Change Initial Value	Maxim Failure
Lotus Notes	0 to 15	na	∞	Alphanumeric, spaces, special charrs	∞	None	●	● [8]	None	na	❍	∞
Mergent PC/DACS	1-15 chars	na	0-365	Alphanumeric, special characters	∞	None	❍	●	0-12	na	❍	Jan-9
Microsoft Access	0 except by recommendation	na	∞	Alphanumeric	∞			●	None			
Oracle	1 char	na	∞	Alphanumeric, spaces (not 1st), punctuation	∞	First character cannot be a space	❍	● [6]	None	na	❍	∞
PC Poweron	0; 7 chars max	na	∞	Alphanumeric, spaces, special characters	∞	None	❍	●	None	na	na	3
Remote Control	Unknown	na	∞	Alphanumeric	∞	None		●	None	na	na	3
Screensaver	0 char	na	∞	Alphanumeric	∞	None		●	None	na	na	∞
Top Secret (MVS/CICS /TSS)	3-8 chars	1-99 days or none	1-255 days [4]	Alphanumeric, special characters @,#,$,|,{, and }	See RPW; 0-7 pairs or disabled	None; See MASK to set requirements	❍	●/❍	2 to 65 (including current)	1-255 or 0 to deactivate	●/❍	1-254 deacti

Environment	Minimum Length	Minimum Age	Maximum Age	Allowed characters	Repeat Characters	Required characters?	Case-sensitive	User Can Change?	History	Grace Period	Force Change Initial Value	Maxim Failure
Unix 10.0	6-8 or none (max length 8)	5 days By user (see also password history)	0-99 [16]	7 bit ASCII	∞	Two letters and one non-letter	● [9]	●	150-180 days [17]	By user in days	●/❍ [14]	defau 9
VM/PROFS	1-8 chars	0	1-365 days	Alphanumeric	First four not allowed/ ∞	None	❍	●/❍	0-9	15 or ∞	● [15]	5 betv IPL's succe login
Voicemail	4-16 chars	None	90 days [4]	Numeric only	∞	All numbers	na	●	0-5	None	●	9-Jan
Windows NT/MOZART	0 to 14 chars	1-999 days	1-999 days	Alphanumeric	∞	None	●	●/❍	1-24 or none	None	●/❍	3
Windows95	0 char	na	∞	Alphanumeric	∞	None		●	None	na	●	∞

[1] Technically possible but not able to synch multiple servers.
[2] Subject to administrator review
[3] Effectively there is a 1 day age enforced for purposes of computing the password cycle.
[4] Or unlimited
[5] May be supported through password checker routine and scripts
[6] Assumes pass-through SQL or application
[7] As either constant words, prefix, suffix or search string. To be determined. Also qwerty strings.
[8] Does not disable old passwords so not recommended.
[9] Except for purposes of comparing passwords.
[10] Either forward, backward, or circular shift
[11] See list.
[12] Forward or backward.
[13] Random
[14] No if can't use aging
[15] By procedure, random
[16]By user in days, cannot set autoexpire default if aging isused
[17] And one must differ by 3 characters.

Appendix E. Glossary

access control list	An authorization mechanism, specifically a list attached to a particular network resource, which specifies the users (human, agent, process) that can access the resource as well as the level of access that they have. For example, an ACL attached to a directory on a file share will list the user accounts. Access levels typically provide either read, write, execute, modify, delete, or create permissions, or combinations of these. Also referred to by some systems as an "access rights list."
access controls	Mechanisms that specify and oversee the specific ability of one entity to access another. Such mechanisms may include hardware features, software features, operating procedures, management procedures.
access	The ability of one entity in a distributed computing system to view, change, or communicate with another entity.
ACL	Access control list. Same concept as an ARL.
ARL	Access rights list. Same concept as an ACL.
authentication	A process in which one entity verifies the identity of another entity.
authorization	The process of determining whether a client may use a network resource and if so, the type of access allowed for each.
client	An application that initiates a connection to a server. More generally, a computer that requests and receives information from another system through a network.
cryptography	The process of communicating in or deciphering secret writings or ciphers.
DES	Data Encryption Standard. A data encryption standard developed by the US government based on classified research.
digital signature	Encrypted data appended to the end of a message (or accompanying a binary file) that functions as a signature to attest to the authenticity of the file. If any change is made to the signed file, the digital signature does not verify.
encryption	Any process that converts plaintext into ciphertext.

firewall	A network protection mechanism that selectively prevents or permits traffic between internetworks, monitors access to network services, and provides an audit trail.
GSS-API	Generic Security Service Application Programming Interface. Generic Security Service Application Programming Interface. An application programming interface that secures communications between a client (or a calling application) and a security service provider. GSS-API can support any underlying cryptographic mechanisms – it's only focused on the communications channel.
Kerberos	An authentication protocol that defines a series of messages that enable a client to acquire a security ticket. Secret-key based.
non-repudiation	A characteristic provided by means of a digital signature which proves the identity of the originator of a message.
public key encryption	A security technique that uses two keys: a public key and a private key. The public key is published and is used to encrypt data, while the private key must be known only to its owner. Messages encrypted with the public key can only be decrypted with the associated private key. Conversely, messages encrypted with the private key can only be decrypted with the public key.
RACF	Resource Access Control Facility
RSA	RSA is a public-key encryption algorithm. (RSA are the initials of the algorithm's developers, Rivest, Shamir, and Adleman.) .
server	The server is the application entity that responds to requests for connections from clients.
SESAME	Secure European System for Applications in a Multivendor Environment. SESAME is a European Community security project whose intent and function is similar to Kerberos, that is, user authentication with distributed access control.
symmetric cipher	A symmetric cipher has the property that the same key can be used for decryption and encryption. An asymmetric cipher does not have this behavior, Some examples of symmetric ciphers: IDEA, RC2, RC4, and DES.
X.509	An optional part of the X.500 recommendation, X.509 is the so-called "authentication framework" — basically, X.509 spells out how one could maintain a public-key as an additional attributed connected to an entry in the X.500 directory. The X.509 specifies the format for the certificate, and recommends RSA public-key authentication mechanism in wide-use worldwide. (RSA is abandoning its own certificate scheme and incorporating X.509).

Appendix F. References

Applied Cryptography: Protocols, Algorithms, and Source Code in C. Second Edition. Schneier. John Wiley & Sons, Inc. New York, NY 10158. 1995.

Building Internet Firewalls Chapman and Zwicky. 1995. O'Reilly & Associates, Inc. Sebastopol, CA 95472

Computer Security Basics. Russell and Gangemi Sr. 1991. O'Reilly and Associates, Inc., Sebastopol, CA 95472

Department of Defense Trusted Computer System Evaluation Criteria. DoD 5200.28-STD, December 26, l985

Distributed Computing: Implementation and Management Strategies. Raman Khanna, editor. 1994. Prentice Hall. Englewood Cliffs, NJ 07632

FIPS PUB 46-2: Data Encryption Standard.

FIPS PUB 48: Guidelines on Evaluation of Techniques for Automated Personal Identification.

FIPS PUB 74: Guidelines for Implementing and Using the NBS Data Encryption Standard.

FIPS PUB 81: DES Modes of Operation.

FIPS PUB 83: Guideline of User Authentication Techniques for Computer Network Access Control.

FIPS PUB 112: Password Usage.

FIPS PUB 113: Computer Data Authentication.

FIPS PUB 171: Key Management Using ANSI X9.17.

FIPS PUB 180: Secure Hash Standard.

FIPS PUB 185: Escrowed Encryption Standard. 2/19/94

FIPS PUB 186: Digital Signature Standard. 5/19/94

FIPS PUB 190: Guideline for the Use of Advanced Authentication Technology Alternatives. 9/28/94.

Firewalls and Internet Security: Repelling the Wiley Hacker. Cheswick and Bellovin. Addison-Wesley Publishing Company. 1994.

Guide to Writing DCE Applications. Shirley, Hu, Magid. O'Reilly and Associates, Inc., Sebastopol, California, 1994.

Network Security. Data and Voice Communications. Simonds. McGraw-Hill. 1996.

Network Security. Private Communication in a Public World. Kaufman, Perlman, and Speciner. Prentice Hall. Englewood Cliffs, NJ 07632. 1995.

NIST Special Publication 500-166 "*Computer Viruses and Related Threats*"

NIST Special Publication 500-169 "*Executive Guide to the Protection of Information Resources.*"

NIST Special Publication 500-170 "*Management Guide to the Protection of Information Resources.*"

NIST Special Publication 500-171 "*Computer Users Guide to the Protection of Information Resources.*"

PGP: Pretty Good Privacy. Garfinkel. 1995. O'Reilly and Associates, Inc., Sebastopol, CA 95472

Power Programming with RPC. Bloomer. O'Reilly and Associates, Inc., Sebastopol, California, 1992.

Recommendations X.500 - X.521. ITU-T (International Telecommunication Union, formerly CCITT) Blue Book. 1988 Recommendation. Phillips Business Information, Inc., Omnicom PPI. 7811 Montrose Road; Potomac, Maryland 20854. 1-800-OMNICOM.

RFC1244 (Informational) "*Site Security Handbook.*" Holbrook, Reynolds. 07/23/1991.

RFC1319 (Informational) "*The MD2 Message-Digest Algorithm.*" Kaliski. 04/16/1992.

RFC1320 (Informational) "*The MD4 Message-Digest Algorithm.*" Rivest, 04/16/1992.

RFC1321 (Informational) "*The MD5 Message-Digest Algorithm.*" Rivest, 04/16/1992.

RFC1411. "*Telnet Authentication: Kerberos Version 4."* D. Borman, Editor. January 1993.

RFC1507 (Informational) "*DASS (Distributed Authentication Security Service)."* C. Kaufman. September 1993.

RFC1508 (Standards Track) "*Generic Security Service Application Program Interface.*" Linn. 09/10/1993.

RFC1510 (Standards Track) "*The Kerberos Network Authentication Service (V5).*" Kohl, Neuman. September 1993.

RFC1511 (Informational) "*Common Authentication Technology Overview.*" Linn, 09/10/1993.

RFC1636. (Informational) "*Report of IAB Workshop on Security in the Internet Architecture.*"R. Braden, D. Clark, S. Crocker, C. Huitema. June 1994.

RFC1704. (Informational) "*On Internet Authentication.*" R. Atkinson. 10/94.

RFC1825 (Standards Track) "*Security Architecture for the Internet Protocol.*" Atkinson. 08/09/1995.

RFC1862 (Informational) "*Report of the IAB Workshop on Internet Information Infrastructure, October 12-14, 1994.*" McCahill, Schwartz, Sollins, Verschuren, Weider. 11/03/1995.

RFC1865 (Informational) "*EDI Meets the Internet: Frequently Asked Questions about Electronic Data Interchange (EDI) on the Internet.*" Houser, Griffin, Hage. 01/04/1996.

RFC1898 (Informational) "*CyberCash Credit Card Protocol Version 0.8.*" Eastlake, Boesch, Crocker, Yesil. 02/19/1996.

Security Architecture for Open Distributed Systems. Muftic, Patel, Sanders, Colon, Heijnsdijk, Pulkkinen. John Wiley and Sons, England. 1993.