Web Security Solutions: Central Authentication for Locally Developed Applications

by Noam Arzt and Daryl Chertcoff

Some schools consider their online courseware to be proprietary and do not choose to make it generally available on the Internet.

The development of the Internet over the past few years has led to new requirements for controlling access to files, data, and other material made available on campus networks. Like many institutional networks at colleges and universities, PennNet (the University of Pennsylvania campus network) is primarily an open network. Access to most of its resources and material is freely available to anyone on the Internet. Some providers of information, however, have had reasons to restrict access to network resources either to members of the university community only or to specific individuals. Examples include:

- Some schools consider their online courseware to be proprietary and do not choose to make it generally available on the Internet.
- Certain administrative applications should not be available from off campus, primarily because access to them is not through secure means just yet.
- Local Usenet newsgroups should not be generally available outside of the campus community.
- Certain applications, like requests for new accounts, need to be made by authorized individuals only (for example, for budgetary approval).

And there are many others. Securing the subset of these resources that are available through Web interfaces is the topic of this article.

Initial methods

Initial strategies for controlling access to Web material primarily used two methods of restriction:

Domain-based restriction. The Web allows for access to resources to be restricted by domain (for example, upenn.edu). All attempts to access a directory of information, for instance, that do *not* originate from a computer identified with a "upenn.edu" domain can be prevented from seeing the material. While this is fairly easy to implement, it prevents access from legitimate viewers who may be accessing the material from commercial Internet Service Provider (ISP) accounts that would not have a "upenn.edu" domain associated with the connections. As more contention develops for limited campus modem pools, an increasing number of users will choose to access from off-campus providers.

User-based restriction. Many Web servers also allow for discrete pairs of usernames and passwords to be created and associated with access to directories of information. This allows users to access material from any computer on the Internet as long as they know a valid username and password. The problem with this solution is that it is an administrative burden when scaled to tens, hundreds, or even thousands of users. Furthermore, it requires users to remember many different combinations of usernames and passwords.

These methods can also be used in conjunction to restrict access to information both by the location of the user (domain) and the user's identity (password).

Authentication at Penn

Application development and support at

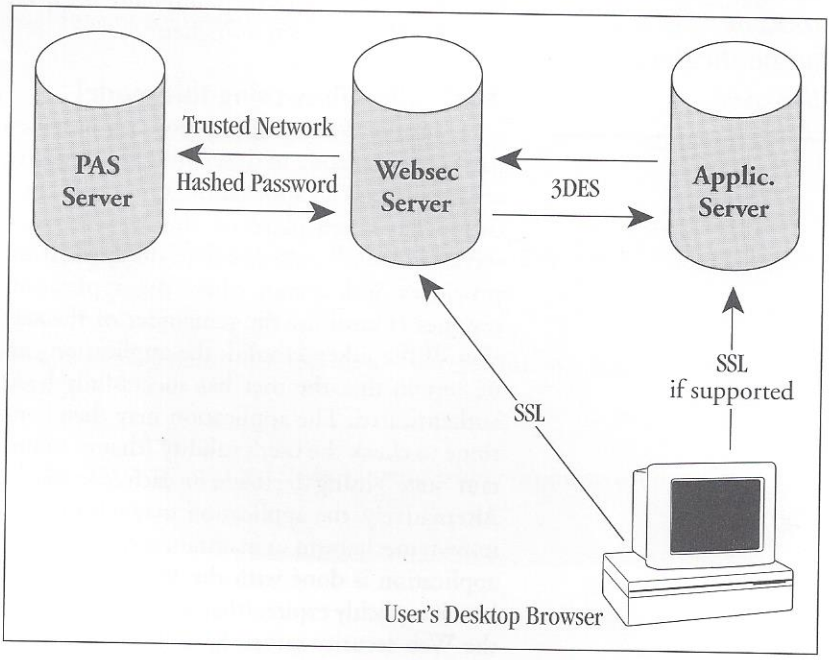
Penn is very distributed—many schools, departments, and even individual faculty members develop applications or need to secure documents on their own servers or in areas of shared Web servers that they control. Penn wished to circumvent the administrative burden associated with maintaining separate usernames and passwords for different Web applications. For several years we have been using a unified username space called the PennNet Authentication System (PAS, see http://www.upenn.edu/computing/netid/) to authenticate access to our modem pools and other network resources. Users create their own PennNet IDs and passwords at special terminals located around campus, typically when they receive their campus identification cards. Users are authorized to create their ID based on their status as students, faculty, or staff as reflected in several central university administrative systems and automatically known to the PAS database. An increasing number of Web developers requested to integrate this campuswide username/password combination into their applications and avoid the need to maintain separate username/password pairs.

However, to secure these usernames and passwords, a system needed to be developed to provide controlled, encrypted access to this namespace as well as to provide an optional facility for maintaining connection "state" for Web applications. Penn developed a generic module based on a three-tier architecture; deployed a dedicated, secure server; and is migrating authenticated applications to this new facility. The primary goal was to enable providers to have flexibility in using the campuswide authentication system in their Web applications while maintaining control of how the authentication system is secured and administered.

Three-tier architecture

The Web security module uses a three-tier architecture to provide application security. This approach has a number of advantages:

- It provides for a separation of services that increases the security of the overall architecture.
- It allows for scalability of the system when required.
- It fits the distributed nature of Penn where staff in schools, centers, and departments work independently and deploy applications on many different servers.
- It can be deployed in conjunction with just about any authentication system. (PAS will likely be replaced over the next several years with Kerberos or another more standards-based authentication system.)



The three pieces of the architecture, as shown above, are the application server, the Web security server, and the PAS server.

Application server. Application developers build applications on their own school, center, or department servers or on the central PennWeb server. This is the primary access point for the user. Developers use a variety of languages and approaches to develop their applications, including simple HTML, Java, JavaScript, and CGI-based programs.

Web security (Websec) server. A central, dedicated authentication server maintained and operated by the central university networking group provides secure, encrypted authentication against the common username space and maintains the state of the user's connection to the application. This server is operated under the most restrictive access rules of any server on campus and performs this authentication function only. It also runs a Secure Socket Layer (SSL) compliant server. Communications be-

tween this server and application servers are encrypted using 3DES encryption.

PAS server. Actual queries for username and password pairs are made against a central SQL-compliant database. The PAS and Websec servers communicate over a trusted network. Passwords sent over this link have been hashed before transmission.

The user accesses the application on a Secure Socket Layer (SSL) compliant browser, typically Netscape 3.n or higher.

The Web security module authenticates the user.

Application flow using this model

Using the Web security approach, a Web user passes transparently to the central Web security server to access an application. Once authentication has taken place on the Web security server, a token is passed back to the application provider's Web server, where the application resumes control for the remainder of the session. If the token is valid, the application can be certain that the user has successfully been authenticated. The application may then continue to check the user's validity (that is, maintain "state") using the token on each of its pages. Alternatively, the application may wish to use its own mechanism to maintain state. Once the application is done with the Web security token, it explicitly expires (that is, removes) it from the Web security server. Specifically, the steps involved are:

1. The user goes to a Web page where a particular service or application is offered.
2. If the application that the user wishes to use requires authentication, he or she will click on a link provided, which points to an authentication page on the Web security server. This link will contain a string that specifies the application for which the authentication will be taking place.
3. Upon clicking on this link, the user is transferred to the Web security server and presented with a screen to validate him/herself with a PennNet ID and password. He or she has only 45 seconds to do so.
4. The Web security module authenticates the user. If authentication fails, the PennNet ID/password combination is incorrect, or the time limit has been exceeded, a screen is returned indicating the problem and the process stops. The user is given the option to reload the authentication page and start over.
5. If authentication is successful, the next screen, as defined by that particular Web security application, is displayed to the user. Note that all relevant information about the session, including a unique token, has been passed along so that the application is able to validate the session. This unique session token, or Web security token, has been generated by the security module and has been stored, along with other relevant information, in its token database.
6. The user submits the form. The token is passed by the browser to a program representing the first page of the application, located on the application's server. With software modules provided by the central computing group, the application can check the validity of the token by querying the security module's token database, located on the Web security server. If the token is valid, the application knows that the user has authenticated him/herself for that application. If it is not valid, either the user has not authenticated him/herself successfully, the token has explicitly been expired, the token has timed out, or the application for which the token was created is not the same as the one asking for information about the token.
7. For each successive page of the application, if the developer of the application wishes to continue to use the token functionality of the security module to maintain "state," he or she must pass the token back to the Web browser as a hidden field or set a client-side "cookie" containing the token. Upon arrival at each new page, the application must check the validity of the token, following the rules in the above step. If the developers of the application do not wish to continue using the token functionality of the security module, they can develop their own method of maintaining "state," such that for each operation a user performs, they have the assurance that the user has been authenticated appropriately. This may be the case for applications which launch client-side programs, such as Java applets.
8. Once a user is done with the application *or* the application is finished with the security module's token database, it expires (removes) the token from the token database.

Current Applications Using the Security Module

- **Online Directory Update**
  This application allows faculty and staff to update their entry in the campus online e-mail directory (http://www.upenn.edu/computing/directory/dir-update.html).
- **Classlists Management**
  This application allows faculty who are using course-specific listservs to manage their list parameters and settings through a Web interface instead of traditional listserv commands via e-mail (http://www.upenn.edu/computing/classlist/).
- **Online E-mail Account Request Form**
  This application allows for e-mail accounts to be requested for the central time-sharing system with appropriate budgetary authorization (http://www.upenn.edu/computing/email/pobox.html).
- **Career Services Credentials System**
  This application allows students to examine their file of recommendation letters online (http://www.upenn.edu/careerservices/credentials/credentials.html).
- **MedView**
  This is a service for medical students that provides a single point of access on the Web for Internet resources (http://www.med.upenn.edu/educate/studentinfo.html).
- **Anti-virus Software Distribution**
  This site-licensed software is distributed using this service (http://www.upenn.edu/computing/product/desk/avd.html).

A new capability of this product exploits the API in Apache's Web server to allow authentication for static Web pages on Penn's main Web server to be accomplished with a customized set of commands in a standard *htaccess* file. The user experiences essentially the same set of steps as described above. An added feature of this access method is simple authorization: The Web page administrator cannot only secure the page (authentication) but can specify an explicit list of PennNet ID's to restrict access to only a limited number of users (authorization).

We have not had any problems reported with applications that have been deployed using this interface (see sidebar for a list of current applications). Developers, even in fiercely independent units, have appreciated the ability to use a central authentication database. As more and more users access protected Penn resources from commercial ISP accounts, they appreciate the movement away from domain-based authentication that prevents access to information from these types of accounts.

The future

Several enhancements to the Web security service will be deployed over the coming months, including improvements to the authentication system (PAS) and deployment of a hardware-based random number generator for generating tokens. In addition, Penn has an active work group focusing on standards-based approaches to authentication, authorization, and accounting (http://www.upenn.edu/computing/group/aaa/).

We believe that other colleges and universities should be able to take advantage of our architecture. While it is currently built with a variety of standard and proprietary tools, its modular approach would allow the substitution of other components that may be in use on a particular campus and migration to standard tools as they become available.

Noam Arzt (arzt@isc.upenn.edu) is a senior fellow and Daryl Chertcoff (daryl@isc.upenn.edu) is a programmer/analyst in the Information Systems and Computing organization at the University of Pennsylvania. C/E