

How WebSmart and the iSeries Provide Everything You Need To Deploy Totally Secure and Safe Web Applications

• Introduction

The purpose of this White paper is to explain how WebSmart in conjunction with the iSeries web server (native or Apache versions) provides all the tools and technology for you to create and deploy web applications that are totally secure and safe. These applications can be browser-based, or SOA applications, typically using web services.

• Terminology

References to “client” generally mean web browser, such as Internet Explorer or Firefox (unless otherwise noted).

References to “iSeries” also means AS/400 or i5.

References to “server” also mean iSeries, AS/400 or i5 (unless otherwise noted).

References to “web server” mean iSeries HTTP server (OS/400 or Apache versions) or iSeries java-based servers such as WebSphere or Apache Tomcat.

References to “SOA” mean “Service Oriented Architecture, commonly deployed as ‘web services’.

• Areas of Concern

There are six major areas of concern with regards to security:

1. Client (browser)
2. Network connection (internet, intranet, extranet)
3. Firewall
4. HTTP Server
5. WebSmart applications running on the iSeries (server).
6. SOA applications – requiring communications between an iSeries server and any other server using the HTTP or HTTPS protocols.

We refer to each of these areas of concern in the following sections, and the security issues specific to each one.

• Security Issues

This white paper will address the following security issues:

1. Sending secure information back and forth from client to server using encryption
2. Saving secure information on the server using encryption
3. Prohibiting unauthorized access to the web server.
4. Prohibiting unauthorized access to OS/400 objects (programs, files, data areas, libraries, etc.)
5. Preventing DOS (denial of service) or DDOS (distributed denial of service) attacks such as ping floods or HTTP request floods.
6. In SOA applications, sending secure information back and forth between servers (web services).

We will explain the network, HTTP, iSeries system architecture and WebSmart programming features that make it possible to effectively address all these issues.

• Saving Information on a Client – Using Cookies

In this discussion, the term ‘client’ refers to a browser, running on any platform such as Windows, Macintosh or Linux. Most browsers and PC’s have the capability to store persistent data on the PC, in the form of ‘cookies’. A cookie is a small file containing information related to a specific site. Browser security (for all browsers) is designed so that a cookie created on one site cannot be accessed, modified or recreated by another site (so as to prevent the ability to hijack someone’s cookie from another site.) WebSmart has functions that fully support the use of cookies. While most browsers provide the ability for users to prevent cookies from being created or stored, you should understand that most commercial sites depend upon cookies in order to function correctly. Cookies often contain human-readable data, though, so when you decide to store information in them, that information should be of a non-sensitive nature. This will prevent other users of a client machine from having access to any sensitive information. WebSmart provides functions for creating and managing sessions (called ‘Smurfs’). You can use these functions to store a session ID in a cookie. This is simply a string of characters- 32 unencrypted, 64 encrypted. A smurf can be used to store sensitive information associated with it on the server instead of the client. We discuss this notion in more detail in a later section titled **Saving Secure Information on the Server using Encryption.**

• Sending Secure Information between Client and Server

While data can be secured readily by using encryption and conventional AS/400 object security, there will often be applications where you need to secure data as it is being transmitted across the Internet. For example, any time a credit card number is required to purchase a product, that number should be transmitted securely. This is where SSL (secure sockets layer) technology comes into play. SSL is a protocol and mechanism that ensures that data is encrypted at the source, transmitted in encrypted form, and then decrypted at the target. The server or browser can play the role of source or target, depending on the nature of the transaction. For example, sending a credit card number for a purchase involves the browser as the initiator of a message sent to the server. The encryption technology used involves private/public key encryption. The server must have a digital certificate installed. This is essentially the ‘public’ part of the key. While the mechanics of private / public key encryption are beyond the scope of this white paper, it is sufficient to know that all web servers (iSeries or not) that implement SSL require a digital certificate. The iSeries has, as part of OS/400, facilities for creating your own digital certificates. However, we recommend that you purchase a digital certificate from a trusted CA (certificate authority) such as Verisign or Thawte, so that anyone who accesses the secure part of your site will know that the certificate has been authenticated by a reliable third party, as opposed to one you have simply generated yourself. Bear in mind that implementing SSL on the iSeries is a process completely independent of using WebSmart. Any static pages, CGI programs, or Java servlets can use the SSL infrastructure on your server once it has been configured and activated.

To specify the use of a secure channel, with SSL in effect, you can use https:// in your URL (as opposed to simple ‘http://’). This informs the browser that the link is secured. For example, a shopping cart program called CART that requires a credit card number could be invoked by clicking on a link to a URL such as the following (non-working example link only):

<https://www.bigcompany.com/cart.pgm>

Clicking on this link will result in communications between browser and server utilizing SSL. All information sent back and forth between them will be encrypted at the source, transmitted in encrypted form, and then decrypted at the target, thus preventing any bots from sniffing and reading sensitive information.

For additional resources on implementing SSL, please refer to the WebSmart Reference Guide, Chapter 10, Security. IBM also has reference manuals and Redbooks on the subject. Search for SSL in the iSeries information center, accessible from www.iseries.ibm.com.

• Saving Secure Information on the Server using Encryption

Once information has been transmitted securely, via SSL, you need a way to store that information securely. While conventional iSeries object-level security can be implemented (discussed in a later section), you may need additional security. Credit card numbers and passwords are examples of data that is so sensitive it requires some additional security. You can use the encryption routines provided with WebSmart to store just those pieces of information that are especially sensitive in encrypted form. For example, a customer master file might contain a number of fields that do not need to be private, such as name, address, city etc., while data elements such as password and credit card need to be encrypted. You can use the encrypt PML function to encrypt data and the decrypt PML function to decrypt it. These functions use AES encryption, which utilizes a seed encryption key. As an additional security measure, you will probably want to secure the encryption key from programmers (secure any source code by deleting it or revoking authority to it) to ensure no programmer can simply decrypt data by calling the decrypt function using the correct key.

If you use this approach, even if a user or programmer can view the contents of a file using some kind of iSeries utility (DFU, SQL, DSPPFM etc.) or program, the sensitive data elements will be unintelligible.

In an earlier section we discussed using session ids (smurf ids) to associate a given user's browser session and interaction with the server. Smurf ids provide a convenient programming mechanism for maintaining state in a web application. They also allow you to store sensitive information associated with a session in smurfs, which are essentially server-side cookies. A smurf is simply a piece of data stored in a protected database file in WebSmart. It is associated with its unique smurf (session) id. So, even if you don't choose to store sensitive information such as credit card numbers in your legacy application database, you can use a smurf to store it instead- either for a very short time (such as the life of the browser session), or as persistent data. You can use the encryption functions to encrypt and decrypt data stored in smurfs in the same manner as for conventional database fields.

• Prohibiting Unauthorized Access to the Web Server

Contrary to popular misconceptions, the iSeries Web Server is actually considerably more secure, by default, than the library system of the iSeries. In the library system, objects are implicitly publicly available until you explicitly revoke authority, or assign custom authority (use and management rights). Entire libraries can be left unsecured, with the only line of defense being the inaccessibility of a command-line interface. In contrast to this, the HTTP original server, when running with the default configuration, denies access to everything; so does Apache. For example:

```
<Directory />  
    AllowOverride None  
    Options None  
    Order deny,allow  
    Deny from all  
</Directory>
```

This configuration code tells the Apache web server to deny access to all directories. You can then modify the configuration file so that subsequent directives selectively open up parts of your server. For example:

```
# Tell Apache where to find programs from url /webtest/  
ScriptAliasMatch ^/webtest/(.*)\.pgm$ /qsys.lib/webtest.lib/$1.pgm  
  
# Allow all programs in this directory to run:  
  <Directory /qsys.lib/webtest.lib>  
    AllowOverride None  
    Options None  
    order allow,deny  
    allow from all  
  </Directory>
```

These directives allow CGI programs in library webtest to run. Note that you have to explicitly tell Apache which libraries are available to run programs from.

• How to Challenge Requests for Valid User IDs and Passwords

In addition to opening access to very specific, limited areas of the server, you can also cause the browser to issue challenges for user authentication. This technique allows you to ensure that only authorized users access those areas of the server that have been opened up. You can base user authentication on iSeries user ids, in which case all iSeries authority features are invoked, including the '3 mistakes and you are disabled' feature for repeat wrong password attempts. Or, you can use validation lists. A validation list is an OS/400 object of type *VLDL. It contains entries with both secure and unsecured information (eg user ids and passwords).

When the browser challenges a user for their user id and password, the HTTP server can validate that information against a validation list entry. Another option for user authentication is to use a database file, and authenticate with a WebSmart program that validates the user and password against records in that file. In this case, you probably want to secure passwords by storing them in encrypted form in the file (using the AES encryption functionality of WebSmart). Using this approach requires that you write a WebSmart front-end login program, and that all subsequent programs in the application check to ensure that the user has successfully logged in. You can do this with the session id functions of WebSmart. For example, the login program can do the following:

1. Prompt for user id and password
2. Validate against a database file (match the user id by key, and match on decrypted password).
3. When validated, create a session id (smurf id) and set a smurf (server-side cookie) to store the user name, and any custom authority settings.

Subsequent programs would do the following:

1. Check for the presence of a valid session id.
2. Check for the presence of a smurf for that session id, containing the appropriate value to indicate the user is authorized to proceed.
3. If no session id or smurf is found, redirect to a 'not authorized' page.

Chapter 10, Security of the WebSmart Reference Guide discusses authorization and authentication in more detail.

• Prohibiting Unauthorized Access to OS/400 Objects

By using the HTTP server authentication method for protecting access to various areas of your site, you can also impose standard OS/400 object-level security. Normally, a special user profile of QTMHHTTP1 is used for jobs running in the web server. Object rights will be determined by the authority to each object available for QTMHHTTP1. As an example of how to impose object-level security, let's assume you have a common library of files, used by both green-screen users and by some web users. If you want to ensure that only green-screen users with the correct authority can access certain files in the application, you can either explicitly grant data rights to files for those users, or grant rights to *PUBLIC and explicitly revoke rights for user QTMHHTTP1. The web server protection directives also let you specify any other user profiles to run under, in addition to the default of QTMHHTTP1. In the original HTTP server, these are protect directives.

• Adopting User Profiles Other Than QTMHHTTP1

A protect directive in the HTTP server configuration can specify to use the default user profile (QTMHHTTP1), or the user profile of the user who has signed on, just like in a traditional 5250 interactive job. In the following example, whenever a user tries to access a page with /cgidev/ in the path portion of the url, they will first see a challenge box, asking for a valid user id and password. That request, and all subsequent requests to that path from that client, will use the user profile that is entered.

```
Protection ProtMySite {
  ServerID MySite
    PasswdFile %%SYSTEM%%
    ACLOverride Off
    mask All
    UserID %%CLIENT%%
  }
protect /cgidev/* ProtMySite
exec /cgidev/*.pgm /qsys.lib/cgidevpgm.lib/*.pgm
pass /cgidev/* /compweb/dev/*
```

So, for example, if the user attempts to go to this url:

www.bigcompany/cgidev/login.pgm

the browser will challenge them for a valid user id and password. Because %%CLIENT%% (instead of %%SERVER%%) is specified for the UserID keyword in the protection directive, the web server will handle the request using the user profile specified at the client. So, if the user signs on as FRED, then all object rights, etc. will be checked against FRED's user profile, not QTMHHTTP1.

You can also configure the Apache web server to accomplish the same thing. Here's an equivalent example:

```
< Location /cgidev/ >
  Require valid-user
  AuthType Basic
  AuthName ProtMySite
  PasswdFile %%SYSTEM%%
  UserID %%CLIENT%%
</Location>
```

You can also adopt a user profile directly in a WebSmart program. WebSmart comes with a set of PML functions for using iSeries user profiles. These functions provide all the capabilities you have in interactive jobs – validating the user profile, changing passwords (if you have authority to do so) and causing a job to run under a given user profile. They are implemented by using native OS/400 APIs that work with user profile security at the operating system level. These PML functions provide the additional flexibility to control security at a program interface level, instead of at the HTTP server level. For example, you could write a WebSmart program that presents an initial login page, asking for iSeries user profile and password. Once the user has successfully entered a valid user profile and password, any subsequent page requests will use this for security purposes. For example, if user FRED logs on, and attempts to access the payroll file, but does not have read data rights to the file, then iSeries object-level authority will kick in and prevent FRED from accessing the data. Object-level authority applies to any OS/400 object type, such as data areas, user spaces, spool files and programs, not just database files.

In summary, you can allow or deny access to OS/400 objects using these mechanisms:

1. HTTP Server configuration directives
2. WebSmart user profile functions and coding techniques that interface with native OS/400 APIs
3. OS/400 object-level security

• Imposing Additional Security by Extending the Web Servers

Both the Original HTTP Server and the IBM HTTP Server Powered By Apache can be extended with user-written code to allow you to impose additional security constraints on your server. Normally, all URL requests are handled by the web servers configuration directives, as discussed in a section above titled **Prohibiting Unauthorized Access to the Web Server**. As a request comes in to the server, it is picked up by the HTTP server, and matched against the configuration rules to see if it qualifies for consideration for a specific action. If it does, that action is taken. This might be to explicitly allow or deny access to that file, or to explicitly running a program. For example, in the Original HTTP server, an exec statement is used to control which CGI programs can run, while a pass statement determines what static files can be served. There are other types of statement, such as service statements, that allow you to specify a module of code to run for a URL. For example:

```
service /nexus/intra/*.pgm
/QSYS.LIB/XL_SMSLIB.LIB/sc_httpsrv.srvpgm:xl_service
```

Here, any requests such as: <http://myserver/nexus/intra/nxmenu.pgm> will be intercepted by this statement, and passed to the service program call sc_httpsrv in XL_SMSLIB. This service program needs to be coded to conform to the requirements for receiving URL requests within the HTTP server. For example, in Nexus, our iSeries-centric Web Portal product, we use this service program to further qualify program execution requests. Using this approach, we can control which specific Nexus users are authorized to view pages or run Nexus components. This allows us to have a database-driven security system that provides an extension to the existing security constraints, much like a 5250 menu system can provide additional security to standard OS/400 object authority.

In this case, the Nexus web server extensions check against a links protection database that determines if users or groups of Nexus users are permitted or denied access to programs or static pages. Because this security measure takes place within the web server, it is extremely secure- nothing has to be done at the application programming level, or the operating system object security level, because unauthorized requests will be rejected by the HTTP server itself, prior to being conveyed to operating system.

• Ping Floods, DOS or DDOS Floods

This subject is really concerned with general network security issues, rather than specific to WebSmart and the iSeries, but we discuss it here because, regardless of what platform you choose for your web server, you need to give consideration to how you will prevent attempted attacks on your network from adversely affecting your operations.

Ping, DOS or DDOS floods are attacks on your server where your server is bombarded with incoming requests for responses. DDOS attacks are more lethal than DOS attacks, because many machines are involved in bombarding your server. These kinds of attacks are executed by malicious internet users. Some are sophisticated programmers, some are 'script-kiddies' – people with some technical knowledge who know how to run a pre-programmed script. The premise behind all these types of attacks is the same: flood your server with so much traffic that it eventually locks up.

The best approach to use to mitigate the effects of these attacks is to use a sacrificial lamb – a combination of hardware/software that absorbs the hits as they occur, but which does not allow any of the flooding traffic to go through to the web server or other servers (mail, FTP, file servers, etc.) inside your network. This is one of the roles of firewalls and routers. So, in general, it is a good idea to have a firewall in place at the conjunction of the internet to your internal network. The firewall may end up going down, but it will prevent any other part of your network from being affected. Although it is beyond the scope of this document to cover them in detail, there are many commercial solutions available to cope with these sorts of attacks. Some companies address this by having an iSeries in their DMZ exclusively dedicated to web serving, which accesses data on their actual production iSeries. In every case it is recommended that a firewall be used as well.

Smaller scale DOS attacks can be mitigated by controlling the number of threads (jobs) that the web server will spawn under heavy loads, and to limit the priority of the web serving jobs. There are also a variety of configuration options within the Apache configuration which relate to detecting and handling DOS attacks.

• SOA Applications such as Web Services

SOA (Service Oriented Architecture) applications usually involve web services – the transmission of information directly between servers over the HTTP transport protocol. An example of an SOA application is credit card validation and approval. In this case, a server sends credit card information along with a potential sales transaction to a web server, in the form of an HTTP request. The initiator of the request is referred to as a web service 'consumer' while the web server that responds and provides the validation data is a web service 'provider'. In order for such a transaction to be secure, the data must be encrypted at either end of the connection and transported back and forth in encrypted form. This is done by using the HTTPS (HTTP secure) protocol. WebSmart fully supports acting as both a web services consumer provider and consumer using HTTPS. So, in the above example, you can write a WebSmart application that makes a request to a credit card approval web server – your application acts as the consumer – and send that request totally encrypted. In addition, the response can be received across an encrypted channel, ensuring no-one can 'sniff' the data and read it as it transits the public network of the internet. Furthermore, as with browser-based applications, once any response is received by the WebSmart application, you can encrypt it using AES encryption, as described earlier, to store it on the server in an undecipherable form.

Note that SOA applications are platform-agnostic, meaning that as long as you have two web servers that fully support HTTP and HTTPS, it doesn't matter what the hardware, operating system or web server software (eg Apache, IIS) is.

• Conclusion

Security issues on the iSeries for web applications are comprehensively addressed by the following combination of technologies:

1. SSL HTTP transport mechanism between client and server (where the 'client' is a browser)
2. SSL HTTP transport mechanism for SOA applications implemented as web services.
3. Firewalls and routers
4. iSeries-centric web server software: Original HTTP server or IBM HTTP server powered by Apache
5. WebSmart
6. 128-bit AES encryption algorithms (included in WebSmart)
7. OS/400 object-level security

By using each of these technologies in the appropriate manner, as outlined in this document, you can develop and deploy iSeries-centric database web applications that are extremely secure, safe and reliable. These can be either browser-based applications or SOA applications.

• Common Security Questions

If you prefer summary answers to your security questions, rather than having to read the details of this white paper, please browse this list of common security questions and their answers.

Question: How do I prevent unauthorized access to libraries, files, programs etc. on my iSeries by Internet or Intranet users?

Answer: Via a combination of the following:

1. HTTP configuration directives. These are your first line of defense on the iSeries. By default, the HTTP server is configured to deny access to everything on your server. In addition, users can only access iSeries programs that implement the HTTP transport protocol. In other words, it is impossible to run programs that normally run in green-screen interactive jobs.
2. HTTP Protection directives (a subset of the configuration directives). A protection directive determines who is authorized to access what areas of your server. You can configure protection directives to do the following:
 - a. Protect specific areas of your server – either in the QSYS file system or the IFS.
 - b. Adopt a user profile for the execution of a CGI program
 - c. Challenge remote users, requiring them to enter a valid user name and password in order to access certain areas of your web site.
3. Using validation lists in WebSmart programs. Validation lists are used in conjunction with HTTP server configuration directives. They provide a flexible alternative to OS/400 user profiles for managing access to your web site. WebSmart provides full support for adding, changing, deleting users from validation lists, including password management. Passwords are stored in an encrypted format.
4. Using native OS/400 object security. You can use commands such as `edtobjaut` to manage authority to objects on your system. Each web job runs under a specific user profile. The default profile is `QTMHHTTP1`. So if you want to be absolutely certain that web users cannot access a payroll master file, for example, you can revoke object authority for user profile `QTMHHTTP1` from that object.

5. Using your own database files in conjunction with a WebSmart program. For example, if you have a customer master file, you can use the customer number as a valid login id. To do this, write a WebSmart program that prompts the user to enter their unique login id and password. The WebSmart program then accepts this information, and checks to see if there is a match against the appropriate file. If so, it continues on. If not, it sends back a page to the browser informing the user that there is an error. Optionally, you could have the WebSmart program count the consecutive number of failed attempts, and turn on a 'disable web access flag' for this account (in the appropriate file). This is further insurance against unwanted access.

Question: How do I make sure information is sent securely back and forth from web server to client?

Answer: Implement SSL on your server. This requires installing a security certificate, which you can purchase for a small annual fee from companies such as Verisign (www.verisign.com). The WebSmart Users Guide includes a chapter that has a comprehensive cookbook for implementing SSL on the iSeries (customers tell us it's much easier to follow than IBM's Redbook on the subject).

SSL encrypts any data to be transmitted over the network (internet or intranet) at the source (either server or browser). Once it reaches its destination, it is decrypted. Browsers have built-in support for SSL. Most B2C sites, such as Amazon and Expedia, use SSL when transmitting sensitive data such as credit card numbers.

Question: How can I encrypt data on my server, so that programmers or other 'power users' cannot read it?

Answer: WebSmart has encryption functions that you can use to encrypt or decrypt sensitive data. These functions use AES, a 128-bit encryption algorithm. See glossary of terms for more details. For example, you can encrypt credit card numbers in a WebSmart program prior to saving them to a database file. If any user looks at the file using a green-screen session, they will not be able to decipher the contents of the credit card field.

Question: How can I encrypt data stored on a client (in cookies), so other people's account information cannot be hijacked.

Answer: Cookies are useful for retaining information about a user so that on subsequent visits to a site the server programs can identify that user and 'personalize the web experience' for them. Amazon is a good example of the use of this technology. After logging in the first time, Amazon will remember who you are by creating a cookie on your client system. Whenever you return to Amazon, they can provide you with recommendations based on your surfing habits at their site, because they can uniquely identify you by reading the contents of the cookie on your system. Generally speaking you would limit cookie information to non-critical information because it is stored on the client. Consider using smurfs instead (smurfs generally still use cookies, but the only information stored in the cookie is a session id).

Question: I want authority to be OS/400 object-based, so that only certain OS/400 users can access specific objects such as files and data areas. How can I adopt a user profile for my web jobs, instead of the standard web user profile IBM uses?

Answer: You can configure either the original IBM HTTP server or the Apache web server to challenge a visitor and ask for a valid user profile and password. Once successfully entered, that user profile can be designated as the one that is used for running CGI programs on the web server, effectively adopting that user profile's authority. In addition, WebSmart comes with a set of user profile functions, so you can write application code that provides this same functionality.

• Glossary of Terms

AES

AES is a 128 bit private-key encryption algorithm, considered virtually unbreakable, and endorsed by the U.S. government as the best algorithm to use. It beats out both DES and triple-DES. WebSmart includes functions that let you encrypt any data using AES. This means you can store sensitive data in iSeries database files encrypted with AES (eg. Passwords or credit card numbers). This prevents rogue programmers or power users who may somehow have unauthorized access to database information from hijacking sensitive data from your files.

Apache Web Server

See HTTP server.

Client

A browser running on a PC or Macintosh. The most popular browser is Microsoft's IE. Other browsers include Netscape, Opera and Mozilla, most of which are available for Windows, Macintosh or Linux operating systems.

Cookie

Persistent data stored on a client via the browser. The data can persist for just the life of an open browser session, or until a specified expiry time. Most browsers provide the ability for the user to set various levels of cookie acceptance (including none at all). Use of cookies by commercial web sites is commonplace. Each cookie is associated with the web site that created it. Browsers contain security code that makes it impossible for any web site to access, modify or recreate a cookie from another site.

Firewall

Software or hardware that controls incoming and outgoing traffic to internal network services such as HTTP (web serving), FTP (file transfer) and email. A firewall serves as the first line of defense to prevent unauthorized access to your network, and to prevent DOS (Denial of Service) attacks. Typically, a network administrator configures the firewall to permit only certain packets of data to enter inside the network. A firewall can be an internet appliance (similar to a router), a PC running firewall software, or software running on an iSeries. Many organizations use a separate piece of hardware so that if the firewall is inundated with outside attacks and subsequently goes down, none of the other servers behind the firewall are affected.

HTTP server

For purposes of this document, the software that provides HTTP transport capabilities for serving web pages. On the iSeries there are two different HTTP Servers that can be installed. The Original HTTP server is the one that was first provided with OS/400, and is based on the CERN HTTP server. The Apache web server is the latest one. It is configurable and runs like the Unix version of the Apache Web Server, which is the most popular web server on the web. The Unix Apache Web Server is Open Source, and covered by the GPL (General Public License). See The Apache Foundation at www.apache.org for more details. The OS/400 version of the Apache Server is *not* Open Source. Other popular web servers include Microsoft's IIS.

IFS

Integrated File System. The area on the iSeries which is used to store static stream files such as images, HTML documents, text documents, etc. The structure of the IFS is similar to a Unix or Windows directory structure. You can work with the IFS directly in a green-screen session using commands such as WRKLNK, or through Operations Navigator, or through Client Access. The IFS can be directly accessed as another network drive from your PC by mapping it to a drive letter, just like any other network drive. Files on the IFS are protected via a combination of OS/400 object security and HTTP server directives.

iSeries

AKA AS/400 or i5

I5

New name for AS/400 or iSeries

Nexus Web Portal

A web portal application built with WebSmart that utilizes the WebSmart WAS. Nexus includes special security features that are integrated into the Original HTTP or Apache Web Servers.

Original HTTP Server

See HTTP server.

OS/400

Operating system used on iSeries and AS/400

PML

The programming language for developing web applications that is integrated into the WebSmart IDE (development tool).

QSYS File System

The traditional file system of OS/400 where objects such as programs, physical and logical files are kept. This uses a library system for organization, where libraries are objects stored in a special system library called QSYS, and other objects are stored within those libraries. (In effect, a library is nothing more than a directory of objects). The HTTP server directives are used to prevent unauthorized access to objects in the QSYS File System.

Server

Generally refers to iSeries, AS/400 or i5 – although in the case of SOA and web services, it refers to any web server platform.

Security Certificate

Provided by an independent company such as Verisign (www.verisign.com). It certifies that your web server is a trusted source for serving information, and allows you to implement SSL on your web server.

Smurf

Server-side piece of information, associated with a corresponding Smurfid. Also, similar in concept to cookies, except stored on the server instead of the client. Smurfs provide a mechanism for securely storing unique pieces of data associated with a given user of a web application.

Smurfid

Unique session id that can be generated by WebSmart programming. The session id can be transmitted back and forth between client and server and provides a mechanism for maintaining stateful awareness between them. A smurf id can be 32 digits, or 64 encrypted characters. Values are sufficiently unique and random to be almost impossible to hijack.

SOA

Service Oriented Architecture – the latest buzzword to describe server-to-server communications, often referred to as ‘web services’. SOA applications allow the direct exchange of information between servers without human intervention across common protocols (HTTP and HTTPs), generally using industry-standard data formats such as XML documents.

SSL

Secure Socket Layers- encrypted data transport mechanism for sending data back and forth from client (browser) to server (web server). Data is encrypted at the source, transported in encrypted form across the network, then decrypted at the target. Requires installation of a Security Certificate on the server (most modern browsers have SSL support built in). Security Certificates can be purchased from companies such as Verisign www.verisign.com. Most companies that have B2C (Business to Consumer) sites that accept credit card transactions use SSL to ensure financial transactions are secure. See also Security Certificate for more information.

Validation List

A type of OS/400 object (type is *VLDL) that contains validation list entries. Each entry has a user id and password, in addition to other programmer-customizable information. A validation list can be referenced by an HTTP configuration protection directive. This causes the browser to present a ‘challenge’ dialog box, asking for user id and password. The user id and password are then processed in the authentication phase of the HTTP server, and checked against the validation list. If a successful match is found, the HTTP server indicates that the user has been successfully authenticated, and it continues on to serve any pages that user is authorized to. If the user id or password or both do not match, the server tells the browser to present the challenge dialog box again, up to a maximum of three (3) times.

WAS

WebSmart Web Application Server – a set of functions utilized in WebSmart-developed applications that help deploy web applications on the iSeries.

Web Server

Synonymous with HTTP server.

Web Services

See SOA.

For additional information on WebSmart or other BCD products, please contact:



Business Computer Design Int'l, Inc.
950 York Road
Hinsdale, Illinois 60521
Phone: 630-986-0800
email: sales@bcdsoftware.com
Website: www.bcdsoftware.com