Derek Lawless

June 20, 2004

GCWN version 4.0 option A

Deploying and Securing SOAP Enabled COM+ Applications

Abstract

This paper will offer guidance and insight into securing SOAP enabled web applications using COM+, IIS, and Microsoft Windows® 2000 Server. In order to do this properly, the paper will use a mock web application. The paper will first present background on web applications, where technology has taken them, and the Microsoft solution for e-business. Then it will explore some of the business needs associated with the mock application and the environment in which it will operate. The paper will then delve into the details of securing IIS, COM+, and the SOAP files. Operating system hardening and ADO security will be largely outside the scope of this paper. Finally, the risks will be explored, the possible security solutions explained, and the final choice will be scripted to allow large scale deployment in an enterprise environment.

Background and Security Challenge Identification

The Internet as we know it today has changed considerably since the days of the first ARPANET back in 1969 (Internet Society, 2003). Today, with nearly 250 million nodes, the Internet has grown to become a primary means of doing business around the world. This new business model has been enabled by the advent of web-enabled applications. Such web-enabled applications allow common Internet users to communicate and create transactions with business servers.

As e-business grew through its infancy in the 1990s, the race for companies to become web-enabled was eclipsing the security of the companies' systems. As a result, hackers moved from system to system with little consequence, wreaking havoc as they went. As identity theft began to soar, legislation and customer demand forced companies to begin looking at Internet security to protect their customers' data.

In addition to Internet applications, companies also began moving many of their internal business applications online. Intranet applications could be migrated from stand alone services on each employee's workstation to centralized servers where they could more easily be administered, supported, backed up, and secured. During this time, a major issue began to arise within security. Many applications had multiple functions and companies did not want all employees accessing all functionality. Human resource employees should be the only ones with access to employee records. Managers should have access to only their direct reports' records. Regular employees should have access only to their own records. This granularity is next to impossible with current operating system authorization schemes. With operating system authorization, access can be either permitted or denied to a file resource but business functionality authorization cannot be controlled.

Enter MTS (and later COM+)

During about the same period of time, Microsoft was busy implementing their web-enablement solution for their Windows® NT 4.0 operating system. Microsoft

Transaction Service (MTS) became an integral part of Microsoft's web strategy with version 2.0. MTS 2.0 was integrated into Internet Information Services (IIS) 4.0 and allowed COM programmers to create business objects that could be accessed via ASP code from IIS. In the MTS paradigm, packages contained components. Each component contained interfaces and each interface contained methods. Microsoft realized the importance of being able to control access to pieces of a business application and integrated role-based security into MTS. Role-based security could be applied at the package level, component level, interface level, and/or method level. This allowed security administrators to implement the granularity of security necessary to meet business requirements. Combined with appropriate application design, certain roles could be allowed access to perform tasks while others were denied that access. MTS later evolved into COM+ with Windows® 2000 but the paradigm stayed essentially the same, although packages were now known as applications.

Now granularity was attained at the business logic level, but what about the data level? Consider an application with 1000 users. The business logic and role-based security prevent all but 10 of those 1000 from performing an action that involves data stored on a database server. The 10 users can change at any given time but their role will not change. It seems foolish to give each individual's ID access to the data level. This situation not only creates an administrative nightmare, but also introduces a potential security vulnerability. A user could theoretically write his own malicious component or other code snippet to access the data and would gain access since he is directly authorized to the data. A solution would need to be developed where users could gain access to the data through applications but not be directly authorized to the data with user credentials. To address this issue, Microsoft introduced the idea of impersonation into MTS. With MTS a COM object can impersonate a caller and perform actions for the client with the client's security context (access files, call other COM objects, etc.) on the local server the object is running on, but then make a network call using the COM object's security context. The COM object can now be authorized to the data resource. Only by going through the MTS application can a user access the data.

The three-tier approach has quickly become a de-facto standard for medium to large enterprises. In this model the presentation tier, business logic tier, and data tier are implemented as separate entities to eliminate single points of failure and to utilize increases in performance and security. Using this paradigm requires logically (and often physically) separate servers that are networked together for communication. The trouble here was that often the backend servers would run a certain operating system or web server while the business-tier and/or presentation-tier would run a different operating system or web server. A way to communicate between physically separate servers that may be running multiple flavors of operating systems and web servers was needed; enter the Simple Object Access Protocol.

SOAP

The Simple Object Access Protocol, or SOAP, is a standard that uses XML messages to communicate between objects. The objects can be written in any language and run on any operating system or web server. Web services need only be "SOAP enabled" to allow them to communicate with any other web service connected to the Internet. SOAP utilizes HTTP as its underlying protocol which any web server can transmit and receive. This opened up the Internet business world like no other. Two separate companies could now have web services that pulled information from each other by simply using SOAP to transmit messages and make application calls.

Unfortunately, HTTP itself is a plain text protocol. This means that SOAP is a plain text protocol. Businesses soon realized that important business data such as credit card numbers and authentication schemes were being passed across the Internet in plain text and could easily be intercepted by a hacker with a network sniffer. In addition, SOAP simplified the process of making an application call to another web service. Without a proper authentication scheme, critical web services offered by banks, insurance companies, and any other online business entity would be wide open to tampering, destruction, and theft.

Thankfully, a protocol was already in place on the Internet that could be utilized by HTTP to provide security transmission of data. Secure Sockets Layer (SSL) provides a protocol over which HTTP can ride. HTTPS (HTTP over SSL) allows for 128-bit end-to-end encryption of data. The encryption occurs just above the Transport layer in the OSI Network Stack so network sniffing devices will only see encrypted packets traversing the network. SOAP data could now safely be transmitted without worry of loss of integrity or confidentiality.

Business Background and Mock Application Requirements

In the course of explaining the controls necessary to deploy and secure a SOAP-enabled COM+ application, this paper uses the fictitious company ACME Banking. ACME is a small to medium sized financial institution that has a need for an online bank for customers and employees alike to access. The application needs to be a web-application to allow customers working on any variety of operating system or web browser to easily make use of the application. Customers should not be able to perform deposits but they can perform withdrawals and check their own balances. ACME employees need to be able to access the application to perform withdrawals, deposits, and obtain a customer's balance. Employees are either tellers or managers. Tellers cannot authorize loans, but can perform all other functions. The bank managers need the ability to authorize loans and retrieve balances.

The solution can be easily broken up into roles based on functionality. The Customer role will have the ability to retrieve balances and perform withdrawals, but will not be able to deposit or authorize loans. The Teller role will be able to withdraw, deposit, and retrieve balances but not authorize loans. The Manager role will be able to authorize loans and retrieve balances but does not need functionality to deposit or withdraw.

Role-based security allows security to be associated directly with functionality. By wrapping the security into the functionality, management and business partners more easily see the benefits of security and security administrators get the authorization controls they desire. The solution for this application primarily uses Microsoft security controls which are already built into the OS product. This reduces the cost associated with additional software. By scripting and automating the security controls along with the server setup, administrative resource usage is kept to a minimum as well. This should put a smile on any manager's face.

Buyoff

It is worth noting something about the acceptance of security by those who pay for it. Business management has often been slow to see a benefit to increased security. Security administrators ramming security controls down management's throats without providing threat and vulnerability statements with risk assessments led to the consensus that security wasn't really necessary. In the absence of incidents, businesses were lulled into a false sense of security. Vulnerability and risk assessments are the most important tools security analysts have in obtaining buy off from management to implement security controls. Business requirements should determine security controls, not paranoid security analysts and not frugal managers.

Environment

ACME will be utilizing Microsoft Windows® 2000, IIS 5.0, and COM+. Microsoft also has the .NET platform available for web application development. The reason this paper does not explore that newer technology is three-fold. First, many companies are accustomed to Visual Basic 6 programming and do not have the budget needed to retrain programmers or hire consultants in order to implement .NET web applications. Second, there is an increased cost to maintaining the .NET infrastructure and training administrators and support personnel with the knowledge necessary for this task. Lastly, even with .NET, COM+ can be utilized to implement much of the role-based security. Many companies are reluctant to move to Microsoft Windows® Server 2003, IIS 6.0, and .NET until industry standards and best practices have been solidified.

Currently ACME has a three-tiered environment; a DMZ with two presentation servers, a trusted zone with two business servers and two Intranet presentation servers, and a clustered database environment. The DMZ presentation servers will be accessed by customers and the Intranet presentation servers will be accessed by employees. Figure 1 shows the network diagram for ACME Corp. The DMZ servers are stand-alone Windows® 2000 servers in workgroup mode. This was chosen because with only two servers, a domain in the DMZ is unnecessary. In addition if a domain were implemented, a single compromise of an ID would compromise the entire DMZ as opposed to just a single server. The trusted zone servers are members of a Windows® 2000 Active Directory domain. Currently there are 20 employee users of the application and an anticipated 5,000 customers.

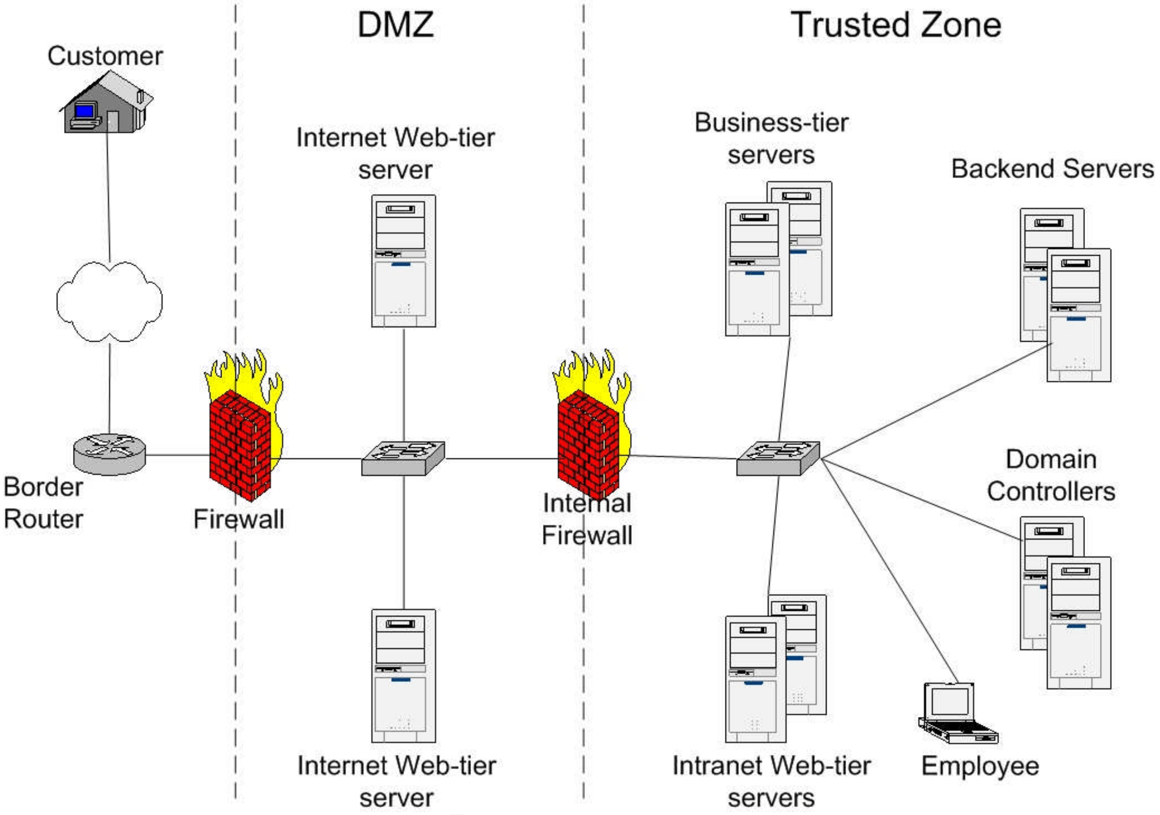


Figure 1

ACME has the desire to deploy the application quickly and consistently to meet future growth. Currently two administrators maintain the environment, so an automated solution of some type is essential.

Deriving a Solution

To reduce the chances of configuration errors due to manual installation and setup, as much of the final solution will be scripted as possible. This will include a small number of scripts that can be loaded when a new server is installed before it is brought onto the network. Some tasks will be performed manually out of necessity, but the goal is to keep these tasks to a minimum and only when scripting is not possible.

As mentioned prior, operating system hardening is largely out of scope for this paper. It will be assumed from here forward that the OS has been installed, all service packs have been downloaded and installed, all security hot fixes are installed, and the file system has been necessarily hardened. Unnecessary services should be turned off and security policies, whether local or through GPO, should be in place. It will also be assumed that IIS, COM+, and Microsoft's SOAP toolkit are already installed on the server.

Network level security will also be out of scope but proper border control including firewalls, anti-virus software, router ACLs, intrusion detection (IDS), and other miscellaneous network security controls should be in place to accent the OS security.

Database security is out of scope as well. ADO calls will originate from the mid-tier servers and end at the clustered database servers. Securing the ADO call will not be discussed nor will securing the database servers. Obviously, security administrators would want to implement encryption of the ADO call and proper database and SQL security on the servers.

Before moving into the solution, a note on the format the paper will follow should be included. The paper will move step by step through the web server (IIS), application server (COM+), and application file security. Each section will initially explain the security controls and how they are implemented before using the example mock application. Background information will be provided where appropriate but the ultimate goal will be to explain how to implement the security controls. The intended audience of the paper is security administrators who have a reasonable background both in Windows 2000 administration and also in general security principals. The paper will not explain every tiny detail or give a "click by click" guide to the implementation but will instead describe the settings that need to be changed and where they are located.

`Courier` type will be used for files, folders, and code examples. *Italics* will be used for specific names such as COM+ application names, virtual directory names, etc.

The Big Picture

The application security flow for the solution is shown in Figure 2. An ID makes a call to an ASP on the Presentation-Tier. That ASP makes a call to the *VBank* COM+ object that then makes a call to a helper *SoapUtils* COM+ object on the presentation server. The COM+ object (*SoapUtils*) makes the SOAP call to the Mid-Tier requesting the WSDL file of the component to which it wishes to make the SOAP call. IIS requests basic authentication and the SOAP client on the Presentation-Tier returns the ID and Password in the HTTP Header. IIS checks the file permissions on the WSDL that is requested using the ID passed via Basic Authentication. If the ID is authorized, the SOAP ISAPI handles the request by passing it to COM+. COM+ checks the role security set on the application, component, interface, and method being invoked by the SOAP call. If all pass, then the call is processed by the specific *VBankBackend* COM+ component and a SOAP response is sent back to the SOAP client. If the call fails, the SOAP response will provide errors to the client.

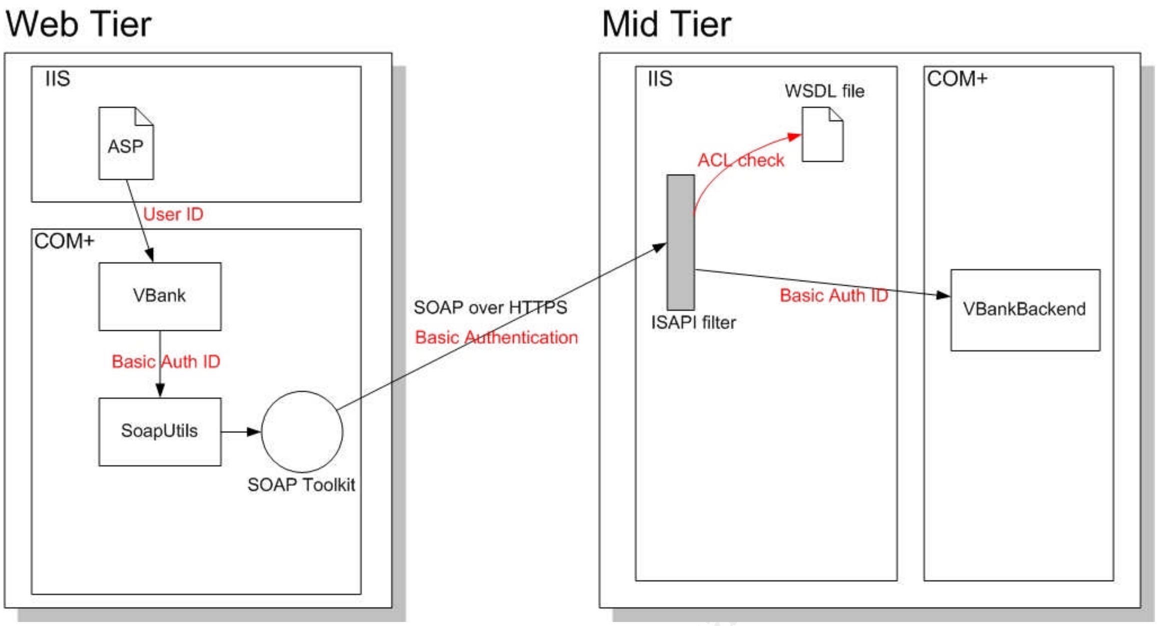


Figure 2

Installing the Application Files

The first task after the OS and environment have been setup will be to install all the files that are needed for the application and to create the necessary file structure to support the application. The files necessary for the application are:

Business Tier (Mid-Tier server) Files

- `VBankBackend.dll` → This file contains components needed to create the COM+ application *VBankBackend.*
- `authorizeLoan.wsdl` and `authorizeLoan.wsml` → SOAP infrastructure files needed on the server side to enable SOAP calls to the *authorizeLoan* component.
- `Deposit.wsdl` and `Deposit.wsml` → SOAP infrastructure files needed on the server side to enable SOAP calls to the *Deposit* component.
- `Withdrawal.wsdl` and `Withdrawal.wsml` → SOAP infrastructure files needed on the server side to enable SOAP calls to the *Withdrawal* component.
- `retrieveBalance.wsdl` and `retrieveBalance.wsml` → SOAP infrastructure files needed on the server side to enable SOAP calls to the *retrieveBalance* component.

Presentation Tier (Web-Tier server) Files

- `index.asp` → This file serves as the front end for all users. It is installed on the Presentation server and invokes a dispatcher COM+ application running on the Presentation-Tier server.
- `SoapUtils.dll` → This file contains components needed to create the COM+ application *SoapUtils*.

- `VBank.dll` → This file contains components needed to create the COM+ application *VBank*.

The directory structure is not of great concern but creating a structure that is consistent and flexible is desirable. Create a structure that will allow future administrators easy navigation as well as provide the flexibility and consistency to allow additional web applications to be deployed.

Business Tier (Mid-Tier server) Directory Structure

- `%APPDIR%\VBANK` will be the root directory for the *VirtualBank* web application
- `%APPDIR%\VBANK\DLL` will contain `VBankBackend.dll`
- `%APPDIR%\VBANK\website` will be the website root directory (the Virtual Directory)
- `%APPDIR%\VBANK\website\WSDL` will contain the WSDL and WSML files for the *VirtualBank* web application

Presentation Tier (Web-Tier server) Directory Structure

- `%APPDIR%\VBANK` will be the root directory for the *VirtualBank* web application on the presentation server
- `%APPDIR%\VBANK\website` will be the website root directory and virtual directory for users
- `%APPDIR%\VBANK\DLL` will contain the `SoapUtils.dll` and `VBank.dll` files
- `%APPDIR%\WSDL` will contain the client WSDL files for making the SOAP call to the Business-Tier server

It is a good idea to create some environmental variables that will allow scripts to utilize some common information. The mock system will contain the following environmental variables:

- %APPDIR% → *C:\APPS*
- %LOGDIR% → *C:\LOGS*
- %SERVER% → *MIDTIER*, *INTER*, or *INTRA* (This corresponds to MIDTIER for the Mid-Tier server, INTER for the DMZ presentation server, and INTRA for the Intranet presentation server)

The %APPDIR% variable will be used to create the directory structure and %SERVER% will be used to determine which environment a script is running on, if needed. %LOGDIR% will be the location used for logs on the systems.

Creating the Necessary Windows Accounts (IDs)

Before any application security is applied, it is important to create the IDs that will be needed for access to the application. These IDs will be used to secure the application files as well as the COM+ role-based security. There are several ways to implement these IDs.

One way is to use Active Directory to create domain IDs for the users, domain global groups to organize them and assign the roles, and domain local groups to assign file permissions. This allows for the least administrative overhead since it utilizes Active Directory exclusively; however if one of the SOAP IDs is compromised, that ID can be exploited everywhere in the domain.

A second way to implement the ID management is by using only local computer IDs. This offers the most security since the IDs are unique to only one (or a few machines at most) and a compromise would be limited to only those machines which have the ID and identical password. This choice however is undesirable since it becomes an administrative nightmare to create local IDs for all users and constantly update these on all servers.

The desirable option is a combination of the above two. For customers coming from the Internet, domain IDs will not be created. They will utilize information from a B2C LDAP directory and share a single SOAP ID passed from the Presentation-Tier to the Mid-Tier. Internally, the employees will each have domain accounts in Active Directory which they currently use to logon to their workstations. They will be members of global groups which will be added to domain local groups and associated with the roles that the application uses. The *VBank* application will perform programmatic checks to see whether a user is an employee or a customer. The *SoapUtils* application will then make the SOAP call using either the customer SOAP ID or the appropriate employee SOAP ID that will represent the employee. The SOAP ID is determined using `IsCallerInRole()` programmatic security in *VBank* and the ID and password is passed from *VBank* to *SoapUtils*.

The Mid-Tier will contain the following IDs and groups:

Local Accounts and Groups

- SOAP_VBANK_CUST → Local ID for the Customer SOAP call
- SOAP_VBANK_CUST_L → Local group used for Customer role membership and file permissions
- SOAP_VBANK_MGR_L → Local group used for Manager role membership and file permissions
- SOAP_VBANK_TLR_L → Local group used for Teller role membership and file permissions
- SOAP_VBANK_MGR → Local ID for the Manager SOAP call
- SOAP_VBANK_TLR → Local ID for the Teller SOAP call
- SOAP_VBANK_L → Local group containing all SOAP_VBANK IDs
- COM_VBANK → Local account that the *VBank* COM+ application will run under on the presentation servers
- COM_VBANK_BACK → The ID that the *VBankBackend* COM+ application will run under

Domain Accounts and Groups

- Employee's individual account → Domain ID used to authenticate employee
- VBANK_MGR_G → Global group containing Manager domain IDs
- VBANK_TLR_G → Global group containing Teller domain IDs
- VBANK_MGR_DLG → Domain Local group containing VBANK_MGR_G
- VBANK_TLR_DLG → Domain Local group containing VBANK_TLR_G

The local accounts will require logon local rights to the Mid-tier server in order for Basic Authentication to work. This can be attained most easily by allowing the SOAP_VBANK_L group logon local rights. This setting will be defined using local group policy for the Mid-tier servers. The setting is located under User Rights Assignment → Log on locally in the Local Policies snap-in.

This is a good point to include a note on the COM+ application identities, COM_VBANK and COM_VBANK_BACK. Password policies are out of scope for this paper; however it is good to note something about passwords for these accounts. Since these accounts will not be used by any users, they should have long, complex pass-phrases. Windows 2000 supports up to 127 character pass-phrases so it is easiest to set the account passwords as 30 or 40 character phrases and then set them to never expire. This reduces administrative headache and still provides exceptional password security. Be sure to document these passwords and keep the documentation secure in case the need to recreate the IDs arises.

The Teller and Manager domain accounts only need *Access this computer from the network* privilege on the Intranet presentation server. This can be set using a Domain GPO but most likely will be set using an OU GPO. The GPO will be out of scope for the paper but this setting is needed to allow Integrated Windows authentication to work.

Securing the Application Files

Now that the files for the application have been installed they need to be secured using NTFS file permissions. The Least Privilege model suggests that only those with need for access be granted access; all other accounts should not have access to any of the files (Saltzer and Schroeder, 1975). In the case of SOAP infrastructure, the WSDL file security is the most important. This is because for this solution, Basic Authentication will be used at the Virtual Directory on IIS. When IIS receives a request for the WSDL file, it will first check the file permissions on the WSDL. If the account requesting the file has permission, it will allow the call to be processed by the SOAP ISAPI filter and forwarded on to COM+. If not, then the SOAP ISAPI filter will return an "Access Denied" error in the SOAP response. Microsoft's implementation of the SOAP ISAPI allowed it to handle the access denied instead of returning a 401 by IIS. In the event of a WSDL file permission error, IIS will return a 200 but the SOAP response will show “Access Denied.” The IDs accessing the WSDL files need only Read access on them in order to be authorized after the Basic Authentication call.

Although the file security on the .dll files is of lesser importance than the server WSDL files in a SOAP enabled application, incorrect permissions on the .dll files can cause access denied errors that are hard to debug. All IDs that will be accessing the COM+ objects associated with the .dll file will need read and execute privilege on the .dll. Securing the .dll file will also prevent unauthorized users or components on the server from making COM/DCOM calls to the COM+ objects. COM/DCOM calls are not audited so preventing this is beneficial from a forensics point of view.

The WSML files associated with each WSDL file should carry the exact same permissions that the corresponding WSDL contains. Any ASP pages should be secured appropriately as well. If the ASP is located on the DMZ Presentation-Tier, Authenticated Users should have read and execute privileges on them to enable the anonymous user account, IUSR_<computer> to view it. The ASP on the Intranet Presentation-Tier should allow the VBANK_TLR_DLG and VBANK_MGR_DLG groups read and execute privilege.

The mock application files will have the following effective permissions where F=Full Control, R=Read, and E=Execute:

- `VBank.dll` (on DMZ servers) → Administrators:F, SYSTEM:F, Authenticated Users:RE
- `VBank.dll` (on Intranet servers) → Administrators:F, SYSTEM:F, VBANK_MGR_DLG:RE, VBANK_TLR_DLG:RE
- `VBankBackend.dll` → Administrators:F, SYSTEM:F, SOAP_VBANK_L:R
- `authorizeLoan.wsdl` and `authorizeLoan.wsml` → Administrators:F, SYSTEM:F, SOAP_VBANK_MGR_L:R
- `Deposit.wsdl` and `Deposit.wsml` → Administrators:F, SYSTEM:F, SOAP_VBANK_TLR_L:R
- `Withdrawal.wsdl` and `Withdrawal.wsml` → Administrators:F, SYSTEM:F, SOAP_VBANK_TLR_L:R, SOAP_VBANK_CUST_L:R
- `retrieveBalance.wsdl` and `retrieveBalance.wsml` → Administrators:F, SYSTEM:F, SOAP_VBANK_L:R

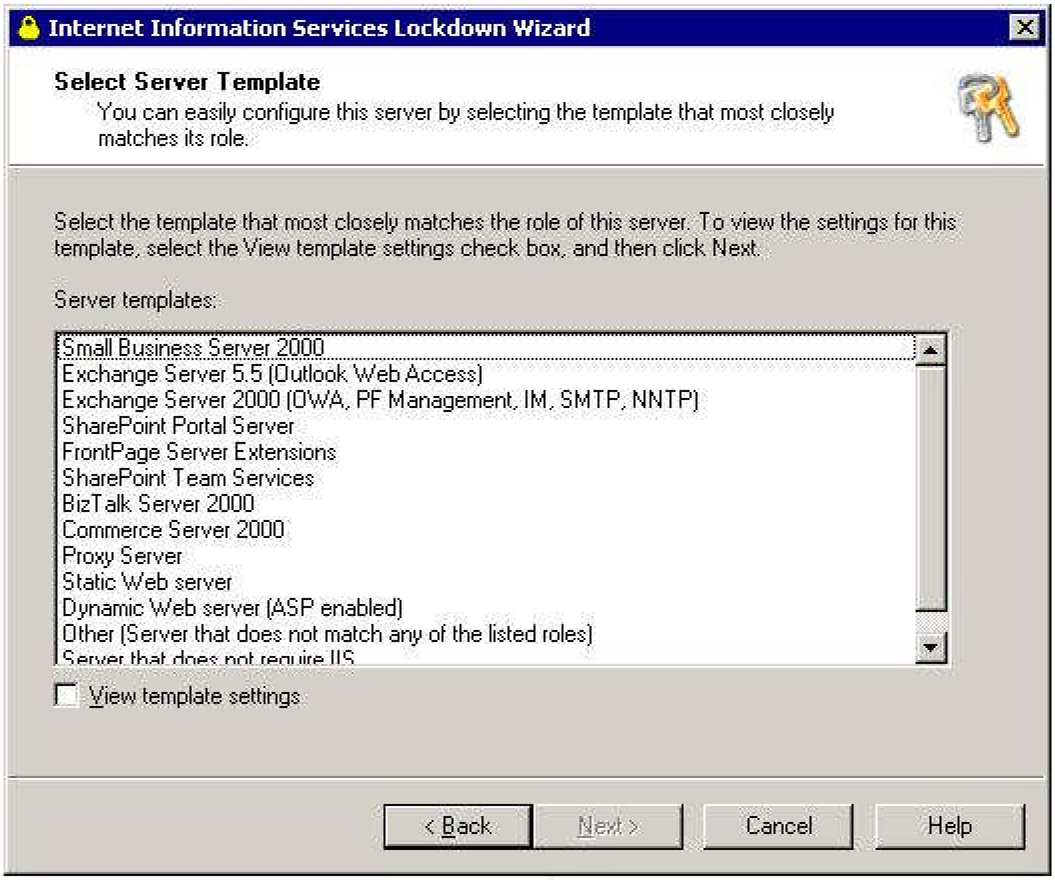
It is a good idea to implement policies within the organization preventing designers and programmers from placing sensitive information in WSDL and WSML files. Although the files are locked down on the server side, the client side often has copies of the files that are not secured beyond authenticated users. IP addresses, IDs, and passwords should definitely not be placed in WSDL and WSML files.

Setting Up and Securing Internet Information Services

Once the file system has been setup and all the application files installed and secured, it is time to address IIS. IIS 5.0 is the web server installed by default with Windows® 2000 Server. There are several pieces involved with the IIS setup. The first would be hardening the out of the box install of IIS and getting it ready to serve web applications securely. The second piece would be setting up the Web Site for the server. This involves making configuration changes to either the Default Web Site that is installed with IIS or disabling the Default Web Site, creating a new Web Site, and configuring that. Many of the settings defined in the Web Site will trickle down to any virtual directories and some can only be changed at the Web Site layer. Third, the server must be setup with an X.509 certificate in order for the server to be able to serve content using SSL. Lastly the virtual directory for the web application must be created, configured, and brought online.

Hardening IIS

Out of the box, IIS 5.0 arrives with many potential vulnerabilities. The very first thing to do is to install all hot fixes from Microsoft to patch all known vulnerabilities. This paper assumes this step is already complete so let us move on. A good start to eliminating the potential vulnerabilities is to download the *IIS Lockdown Tool* from Microsoft. This is a must have tool for anyone running IIS. The tool has a good user interface and does a fairly thorough job of locking down much of IIS. The first screen (Figure 3) that will come up with the tool will ask what Server Template you would like to apply. The choices range from an Exchange server to a server that does not require IIS (thus allowing IIS to be uninstalled). The mock application uses ASP pages so from this point on the paper will use Dynamic Web Server template. The next screen asks whether to install *URLScan*. *URLScan* is an ISAPI filter which screens incoming requests to the web server and blocks those requests that look malicious. It comes with an .ini file called `urlscan.ini` which can be configured to further lockdown IIS. *URLScan* is also an excellent tool and should be installed. The next screen will show what changes the Lockdown tool is applying to IIS.



Internet Information Services Lockdown Wizard

Select Server Template

You can easily configure this server by selecting the template that most closely matches its role.

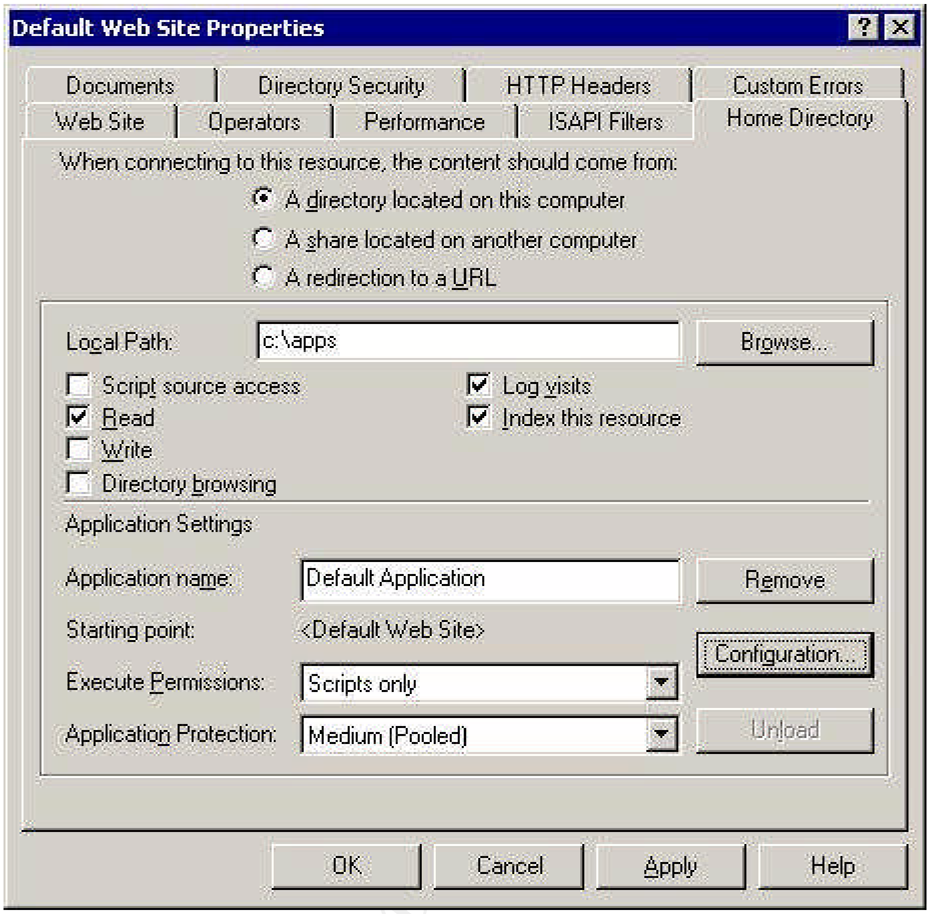
Select the template that most closely matches the role of this server. To view the settings for this template, select the View template settings check box, and then click Next.

Server templates:

Small Business Server 2000
Exchange Server 5.5 (Outlook Web Access)
Exchange Server 2000 (OWA, PF Management, IM, SMTP, NNTP)
SharePoint Portal Server
FrontPage Server Extensions
SharePoint Team Services
BizTalk Server 2000
Commerce Server 2000
Proxy Server
Static Web server
Dynamic Web server (ASP enabled)
Other (Server that does not match any of the listed roles)
Server that does not require IIS
□ View template settings
< Back
Next >
Cancel
Help

Figure 3

The first change that the Lockdown tool will make is to remove many unnecessary app mappings. By default IIS comes with a plethora of mappings for various file extensions such as .asp, .cer, .htr, .printer, etc. The App Mappings allow IIS to map each file type to a corresponding ISAPI filter which will process the requests for that file type. Unnecessary app mappings increase the exposure potential for your web server. In the past, vulnerabilities have been found within the ISAPI filters themselves and malformed requests for certain file types have allowed attackers to gain elevated SYSTEM privileges. For the Dynamic Web Server template, it will remove the .idq, .htw, .ida, .idc, .shtml, .shtm, .stm, .htr, and .printer mappings. It is a good idea to check the effective setting for app mappings under the IIS Snap-in under Web Site → Properties → Home Directory → Configuration to ensure that the Lockdown tool removed the app mappings and to remove any additional unneeded mappings. Figure 4 shows the Home Directory tab and Configuration button.



When connecting to this resource, the content should come from:

● A directory located on this computer
○ A share located on another computer
○ A redirection to a URL

Local Path: c:\apps

□ Script source access
☑ Read
□ Write
□ Directory browsing
☑ Log visits
☑ Index this resource

Application Settings

Application name: Default Application

Starting point: <Default Web Site>

Execute Permissions: Scripts only

Application Protection: Medium (Pooled)

Figure 4

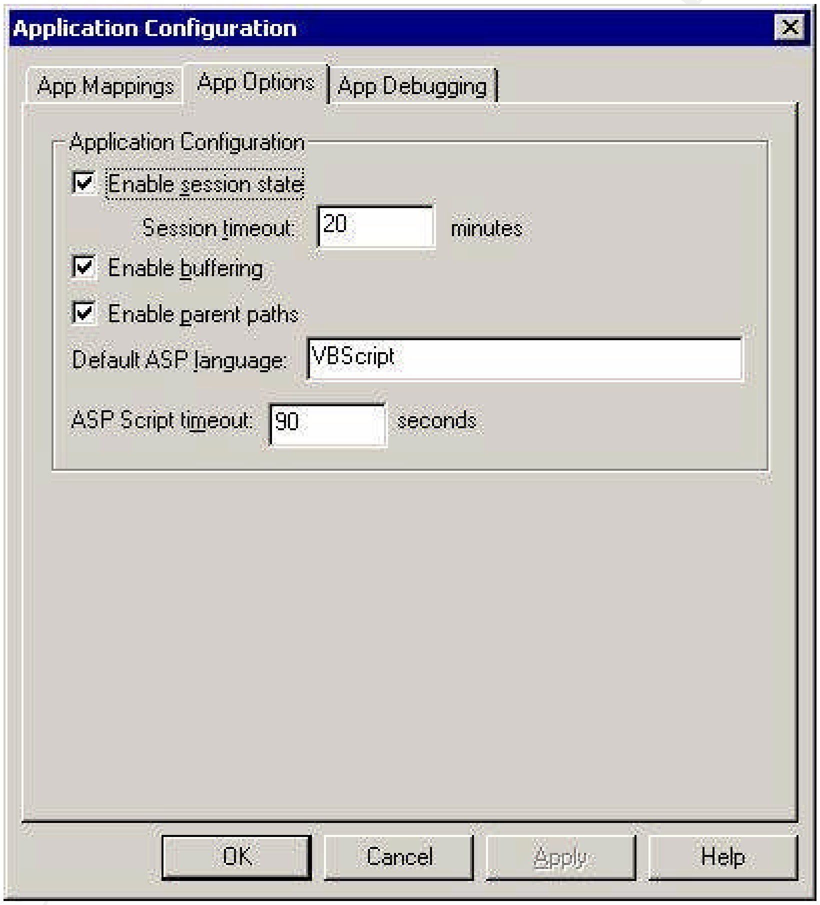
Next, the Lockdown tool will disable WebDAV which has also had vulnerabilities in the past and set more restrictive file permissions on the content files. This prevents the anonymous account from having write permission to the file system and having execute permission on system files. In addition, the tool also removes a number of virtual directories that contain examples and scripts that can be used by attackers to gain access to the server.

Inspecting IIS, notice that the Administrative Web Site has been removed and all that remains is the Default Web Site. Under the App Mappings, those app mappings mentioned above are mapped to `404.dll` which returns a 404 error when requesting a file of that type. It is possible to just delete the unneeded app mappings which eliminates any possibility of a corrupted `404.dll` causing undesirable effects.

Although the Lockdown tool provides excellent security over the base install of IIS, more can be done. Continuing under the Application Configuration tab, move from App Mapping to App Options. In this tab you will see several things pertaining to session state, buffering, parent paths, and script languages. The important pieces here are session state and parent paths. By enabling session state with a low session length of 5 minutes, the application can actually timeout and depending on the logic of the application, can require the user to login again. This is a nice security feature if your application is programmed to support it. More importantly, though, you will see that *Enable Parent Paths* is checked. This is another potential vulnerability. By allowing parent paths IIS will allow URL syntax like the following:

http://host.domainname.com/vdir/../../Winnt/cmd.exe

This can allow an attacker to potentially access a system file such as the Command prompt. Uncheck *Enable Parent Paths* to prevent the use of the "../" syntax. Figure 5 shows the setting on the App Options tab under Configuration.



Application Configuration

☑ Enable session state

Session timeout: 20 minutes

☑ Enable buffering

☑ Enable parent paths

Default ASP language: VBScript

ASP Script timeout: 90 seconds

Figure 5

A nice feature of IIS 5.0 is the ability to specify that applications run as isolated processes. This feature allows web applications to kick off their own separate `DLLHost.exe` process running without elevated privileges. This setting is located on the Home Directory tab under the IIS snap-in and has three potential settings:

- Low → this allows all applications to run under the IIS `inetinfo.exe` process

- Medium → a process called `DLLHost.exe` will kick off and all web applications will run under this single process
- High → this forces each web application to spawn its own `DLLHost.exe` in addition to the generic one created for IIS

The advantage to High would be that any given web application that is compromised will not give the attacker SYSTEM privileges and won't impact any other web application. This setting is not without penalty however as performance will take a hit by enabling each application to run as its own process. It is not advised that an application ever be set to run as Low due to security risks. Non-essential, non-critical applications can be run with Medium level to decrease the performance loss. Critical and sensitive applications should be run as High to take advantage of the separate `DLLHost.exe` processes. The Web Site level should be set to Medium and any virtual directories requiring High can override the Web Site inheritance.

The last IIS hardening step to be outlined by the paper will be reducing the user rights on the Web Site. IIS has two different settings which limit user rights and both are located under Web Site → Properties → Home Directory tab in the IIS snap-in (refer to Figure 4). The first is a series of checkboxes which define the rights assigned to the root directory and files of the website. The check boxes are:

- Script source access
- Read
- Write
- Directory browsing
- Log visits
- Index this resource

Script source access allows the user to gain access to script source such as ASP. Would be attackers can now see the source code running on the server. This is obviously not a good security practice, so this box should never be checked. *Read* access is necessary if anyone is to use the website so this setting should always be checked. *Write* access should be turned off unless there is an absolute reason to turn it on. With write access a user can create files and modify existing files under the root directory. *Directory browsing* allows the user to see everything in the root directory. Combined with allowing parent paths (see section above on disabling parent paths), a user can see the file and directory structure of the entire web server! This is definitely not a good thing so directory browsing should always be unchecked. *Log visits* functionality enables all traffic to the website to be logged in the IIS log. This should always be turned on to allow for forensics if needed as well as to monitor any potential attacks against the website. The *Index this resource* setting only applies if Microsoft Index server is used. If it is not used, this setting has no impact.

The second piece of user access configuration on the Home Directory tab is Execute Permissions under the Application settings. There are 3 potential settings: *None, Scripts only,* and *Scripts and Executables*. The choice here depends on the needs of the website. If static content is only to be served, then *None* is appropriate. If ASP or some other script language is used then *Scripts only* is appropriate. If CGI or some other executable is required then *Scripts and Executables* should be set. Generally, it is best to avoid using executables and set this to *Scripts only*.

The VirtualBank Web Sites will all use an *Application Protection* setting of “High”, *Read* and *Log visits* rights, and *Scripts only* execution.

Configuration of the Web Site

Much of the configuration of the Web Site has been completed by the hardening steps above. However some general things still remain to be completed. The goal behind configuring the Web Site is to change any defaults that IIS comes with to those that are appropriate for the server. This involves changing the location of the Web Site root, the configuration of log files, the default documents for the Web Site, and changes to the HTTP error messages.

Setting the location of the Web Site root is a simple configuration. This should be pointed to a root folder where default content would be stored. This does not limit the ability to point a virtual directory to any location on the server, but it gives a starting point for the Web Site. Typically, virtual directories would be setup below this root but again, that is not a requirement. This option is defined by the *Local Path* setting at Web Site → Properties → Home Directory in the IIS snap-in. The *Local Path* setting will be defined as follows for the *VirtualBank* web application:

- Internet Web-tier → %APPDIR%
- Intranet Web-tier → %APPDIR%
- Mid-tier → %APPDIR%

The next step would be to configure the log files. There are a number of settings that can be configured for the log files, however only a few are of significant importance to this paper. The logging settings are located under Web Site → Properties → Web Site. The majority of the settings are then located under the Properties button. The *Log file directory* setting will specify the location to store the logs generated. By default this is in the %WINDIR%\System32\LogFiles directory. It is advised to change this to a different folder to obfuscate the location of the IIS logs. Choose a location for IIS log files to be stored and ensure that it is locked down to only Administrators, SYSTEM, and anyone else who needs Read access to the log files. *VirtualBank* will use %LOGDIR%\Web.

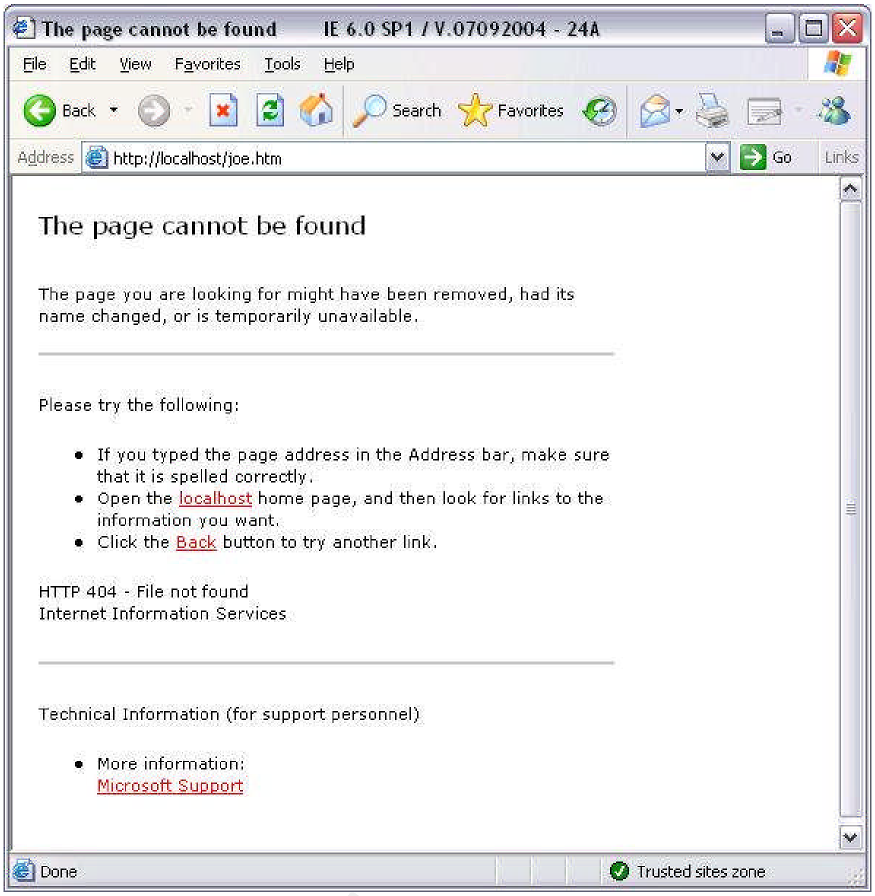
The *Extended Properties* tab allows additional logging information to be turned on as well. By default IIS will log the following items:

- Client IP Address → Logs the address the user is coming from
- User Name → Logs the user name that is authenticating to the server (anonymous user is not logged)
- Server IP Address → Logs the IP of the server
- Server Port → Logs the port that the request is coming into (80 for instance)
- Method → Logs the HTTP method used (GET, POST, etc.)
- URI Stem → Logs the resource requested
- URI Query → Logs any query the client tried to perform
- Protocol Status → Logs the HTTP code returned (200, 404, 401, etc.)
- User Agent → Logs the agent accessing the server (Internet Explorer, Mozilla, Netscape, etc.)

These defaults are adequate for our mock application. URI query is probably the least important for the *VirtualBank* application since it should usually log nothing. The rest are all important to both troubleshooting as well as security logging.

The Web Site → Properties → Documents tab allows a default document to be specified for the Web Site. This implies that a user does not need to specifically request a document when hitting the root of the Web Site. For instance, a browser query can be made to http://www.sans.org/. Although the request didn't specify it wanted index.html, that file was returned because it was defined as the default document. Typically this setting is set at the Web Site level but may be disabled or changed per virtual directory. For the Web Site in the *VirtualBank* example, `Default.htm` will be left as the default document. This setting will be changed for the virtual directory setup.

The last bit of Web Site configuration explored in this paper will be HTTP errors. By default, IIS comes installed with a number of default documents that are displayed by the web browser according to the HTTP error returned. Figure 6 shows the default 404 error returned. The problem with these default pages is that they clearly state that IIS is running on the web server. This may not be a problem for Intranet applications but it gives away information to potential attackers from the Internet that the web server is running IIS. Although it is possible to determine that IIS is running through other probing and investigation from attackers, it should not be as easy as requesting a document that does not exist. The documents that are returned are defined under Web Site → Properties → Custom Errors. Figure 7 shows this tab. By selecting the HTTP Error from the table, choosing *Edit Properties* button and selecting a message type of *Default,* the HTML returned will specify only the error and nothing about IIS. It is also possible to further obfuscate by creating custom files to return. This can be done by creating a file and specifying its location with the *File* setting under *Edit Properties*.


The page cannot be found IE 6.0 SP1 / V.07092004 - 24A
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost/joe.htm Go Links
Done Trusted sites zone

The page cannot be found

The page you are looking for might have been removed, had its name changed, or is temporarily unavailable.

Please try the following:

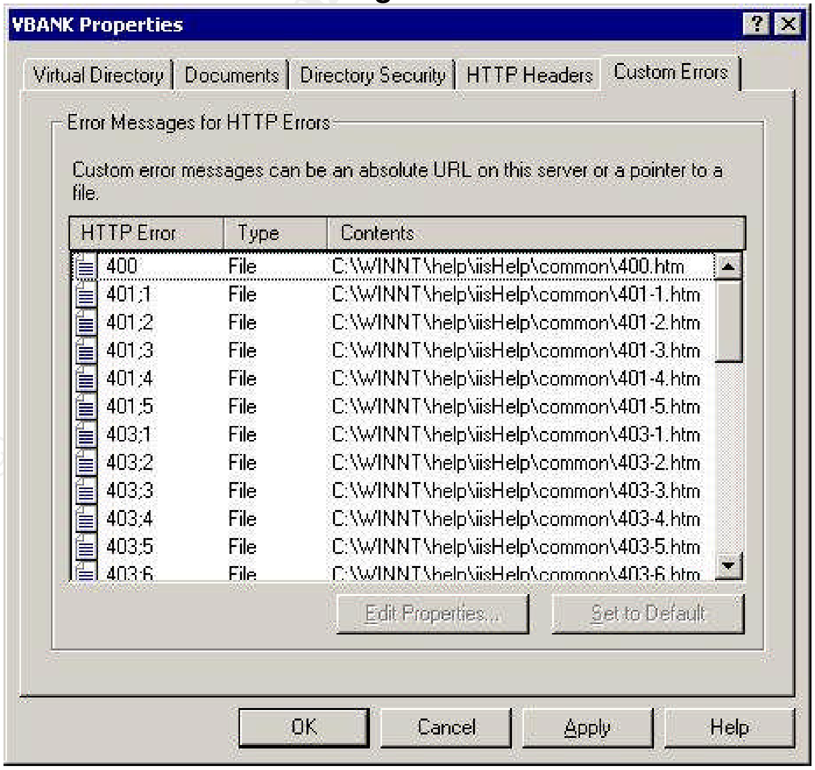
- If you typed the page address in the Address bar, make sure that it is spelled correctly.
- Open the localhost home page, and then look for links to the information you want.
- Click the Back button to try another link.

HTTP 404 - File not found
Internet Information Services

Technical Information (for support personnel)

- More information:
  Microsoft Support

Figure 6


VBANK Properties
Virtual Directory Documents Directory Security HTTP Headers Custom Errors
Edit Properties... Set to Default
OK Cancel Apply Help

Error Messages for HTTP Errors

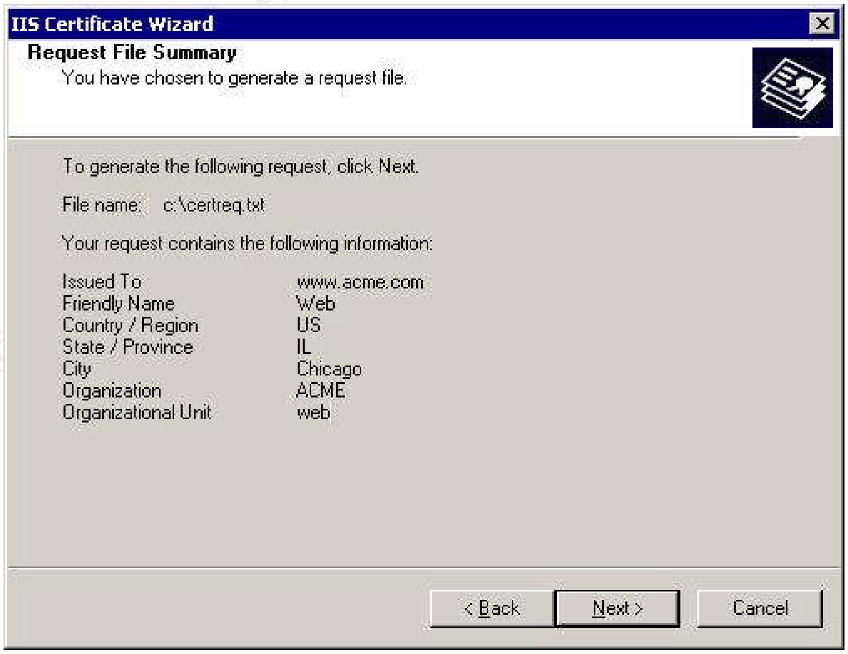
Custom error messages can be an absolute URL on this server or a pointer to a file.

HTTP Error	Type	Contents
400	File	C:\WINNT\help\iisHelp\common\400.htm
401;1	File	C:\WINNT\help\iisHelp\common\401-1.htm
401;2	File	C:\WINNT\help\iisHelp\common\401-2.htm
401;3	File	C:\WINNT\help\iisHelp\common\401-3.htm
401;4	File	C:\WINNT\help\iisHelp\common\401-4.htm
401;5	File	C:\WINNT\help\iisHelp\common\401-5.htm
403;1	File	C:\WINNT\help\iisHelp\common\403-1.htm
403;2	File	C:\WINNT\help\iisHelp\common\403-2.htm
403;3	File	C:\WINNT\help\iisHelp\common\403-3.htm
403;4	File	C:\WINNT\help\iisHelp\common\403-4.htm
403;5	File	C:\WINNT\help\iisHelp\common\403-5.htm
403;6	File	C:\WINNT\help\iisHelp\common\403-6.htm

Figure 7

SSL configuration
To this point, IIS has been installed, patched, secured, and the Web Site configured. There is only one thing at the server level to complete for IIS and that is the installation of an X.509 certificate and configuration of SSL. The SSL will be integral in our security design to provide integrity and confidentiality of customer data as well as to protect the plain text credentials sent via Basic Authentication in the HTTP header.

The first thing to do in order to get SSL up and working on the web server is to create a request for a certificate. The server must create a request key that will be submitted to the Certificate Authority. The Certificate Authority (CA) will then issue a certificate which contains the public key for the web server and a digital signature from the CA. If the client trusts the CA, then it will automatically know that the public key provided in the web server's certificate is authentic. In order to prepare the request, navigate to Web Site → Properties → Directory Security → Server Certificate… in the IIS snap-in. This will open the Web Server Certificate Wizard. Choose to "Create a new certificate" and then to "Create request now, but send it later." The next screens will ask for key length and some organizational information. The important part is the screen that asks for "Common Name." This is the name of the server on the certificate. This must match the server name exactly. The Internet presentation server will use www.acme.com in the mock example. After entering the common name, more information such as state and city will be needed to go on the certificate. Finally, save the request to a text file. The information in this text file will be used to create the certificate on the CA. Figure 8 shows the completed certificate request information.



Request File Summary
You have chosen to generate a request file.

To generate the following request, click Next.

File name: c:\certreq.txt

Your request contains the following information:

Issued To	www.acme.com
Friendly Name	Web
Country / Region	US
State / Province	IL
City	Chicago
Organization	ACME
Organizational Unit	web

Figure 8

Internet facing servers should receive certificates from external sources such as Entrust, Verisign, Thawte, etc. Most web browsers have copies of the root CA certificate for these sources and that way users will never see certificate trust prompts when they visit the site. Intranet presentation servers and mid-tier servers can make use of an internal certificate authority specific to the business and save money on certificate costs.

Figure 4 shows the contents of the certificate request text file. The information that is given to the CA would be the Base-64 encoded portion between the `-----BEGIN NEW CERTIFICATE REQUEST-----` and `-----END NEW CERTIFICATE REQUEST-----` lines.

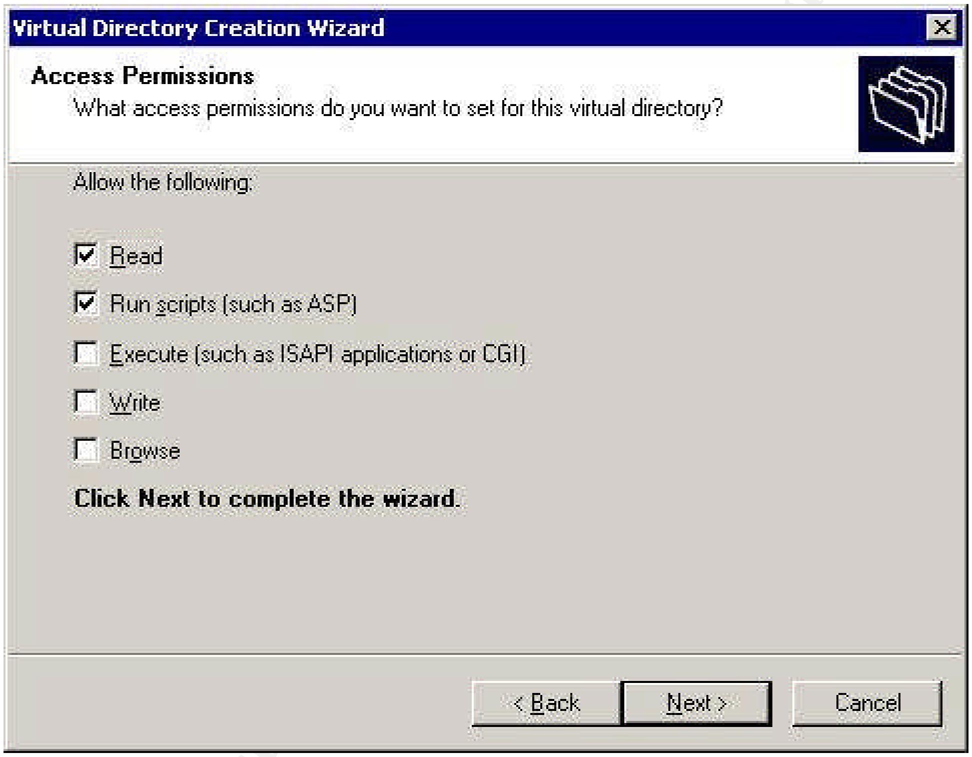
The CA will generate a .cer file and it will be provided for installation on the server. In order to install the certificate, return to Web Site → Properties → Directory Security → Server Certificate… in the IIS snap-in. This time select "Process the pending request and install the certificate" on the first screen. The next screen will ask for the location of the .cer file provided by the CA. This will import the certificate into the server and allow SSL to be used.

The last thing to setup on the server to utilize SSL is to specify a port for SSL. The default port for HTTPS is 443 so in order to enable clean, port free URLs, 443 should be set as the SSL port for the server. To do this, navigate to Web Site → Properties → Web Site and set the value for *SSL Port.* The server should now be ready to serve HTTPS traffic over port 443.

Virtual directory creation and setup
The last portion of the setup and securing of IIS involves creating a virtual directory for the web application and configuring it. A virtual directory is not absolutely needed but is useful if the Web Site intends to serve more than one web application. This section will go over the steps needed to create a virtual directory, change some of the configuration settings, and enforce some additional security.

The first step will be to create the virtual directory. In order to do this, a Web Site must already be configured. Virtual directories must be created within a Web Site. A virtual directory can be created by right-clicking the Web Site and choosing New → Virtual Directory. This kicks off a wizard to setup the preliminary settings for the virtual directory. The first screen will ask for the alias for the virtual directory. This is the portion tacked to the end of the Web Site name that will allow access to the virtual directory. For example an alias of *site* will produce a URL of http://www.servername.com/site/. The second screen will ask for the directory that the virtual directory should point to. Typically this is below the root Web Site directory but does not have to be. It can point to any directory on the server, to a share on another server, or even to another URL as a redirector. The final screen in the wizard will ask to setup the access permissions for the virtual directory. These permissions are exactly like the permissions that can be defined at the Web Site level. They are as follows (Figure 9 shows the screen shot):

- Read
- Run scripts (such as ASP)
- Execute (such as ISAPI applications or CGI)
- Write
- Browse



Access Permissions
What access permissions do you want to set for this virtual directory?

Allow the following:

☑ Read
☑ Run scripts (such as ASP)
□ Execute (such as ISAPI applications or CGI)
□ Write
□ Browse

Click Next to complete the wizard.

Figure 9

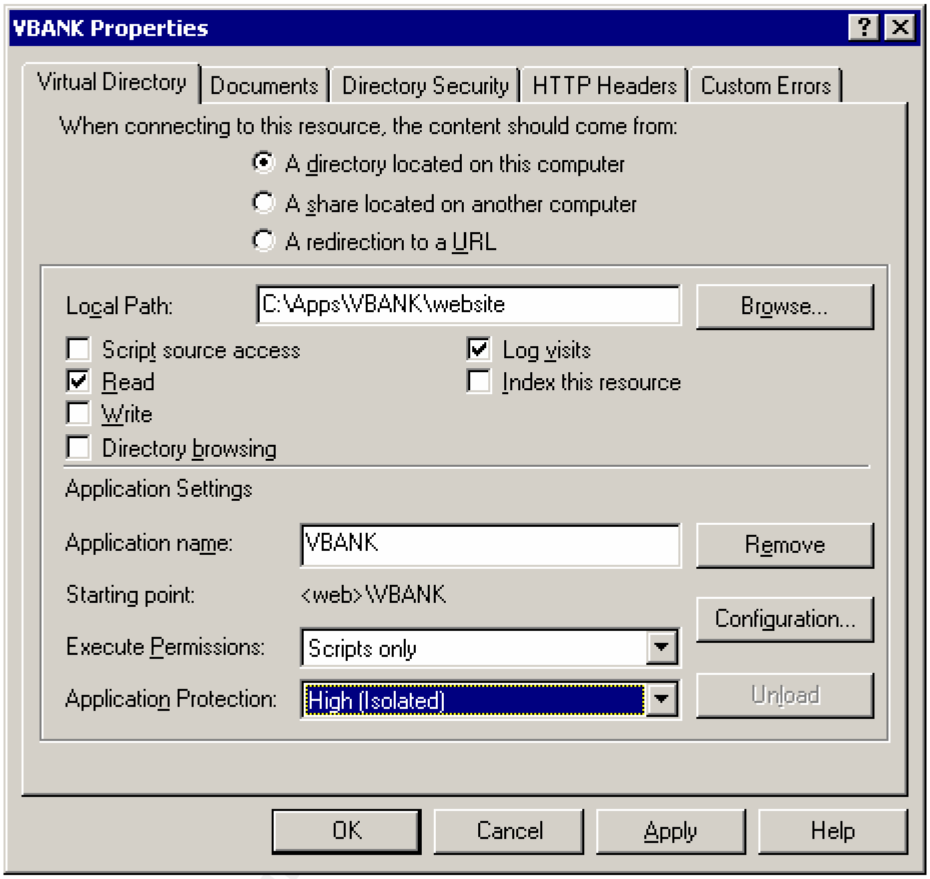
The virtual directory settings will override those that are defined by the Web Site. Refer to the Hardening IIS section for more information on each of these settings. The same security recommendations apply here as at the Web Site level. If more relaxed permissions are required, it is better practice to apply them at the virtual directory level than at the Web Site level.

In addition, the Mid-tier server needs a mapping for .wsdl files in order for SOAP requests to the .wsdl files to be processed using the ISAPI filter for SOAP provided in the SOAP toolkit. The mapping should use the `SOAPIS30.dll` ISAPI filter which is located in the SOAP toolkit install folder; most likely at `C:\Program Files\Common Files\MSSoap\Binaries`.

The VirtualBank virtual directories have the following settings defined (Figure 10):

- Virtual directory alias → VBank

- Local Path → `%APPDIR%\VBANK\website`
- Execute Permissions → Scripts only
- Application Protection → High
- Default Documents → `index.asp`



VBANK Properties

When connecting to this resource, the content should come from:

● A directory located on this computer
○ A share located on another computer
○ A redirection to a URL

Local Path: C:\Apps\VBANK\website

□ Script source access
☑ Read
□ Write
□ Directory browsing
☑ Log visits
□ Index this resource

Application Settings

Application name: VBANK

Starting point: <web>\VBANK

Execute Permissions: Scripts only

Application Protection: High (Isolated)

Figure 10

Now that the virtual directory is created it will show up under the Web Site and highlighting it will show all the folders and files accessible through the virtual directory. Navigating to the virtual directory → Properties will bring up a configuration screen similar to the Web Site → Properties screen. The virtual directory Properties has a subset of some settings from the Web Site level. Documents, Directory Security, HTTP Headers, and Custom Errors are all present at the Web Site level as well as the virtual directory level. The Virtual Directory tab is very similar to the Home Directory tab from the Web Site Properties. The settings at the virtual directory level are all inherited from the Web Site except those set in the wizard for the virtual directory. This allows the virtual directory to take advantage of the hardening that was performed for the Web Site.

Virtual directory authentication

Moving on the next step to setting security at the virtual directory will be to configure the virtual directory on each web server to use the proper authentication scheme. This will be as follows:

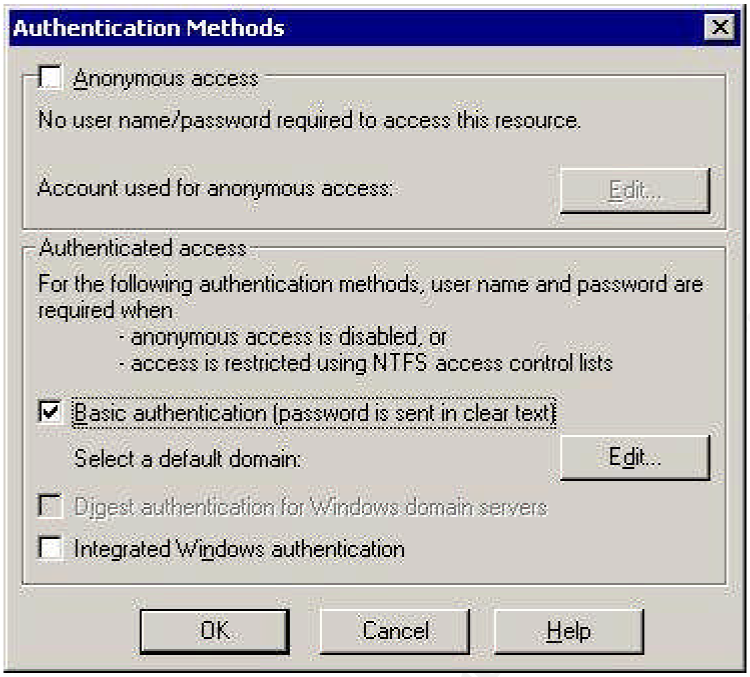
- Internet Presentation-tier server → Anonymous access
- Intranet Presentation-tier server → Integrated Windows authentication
- Mid-tier Business logic server → Basic Authentication

To set this up, navigate to virtual directory → Properties → Directory Security and click on the *Edit…* button under Anonymous access and authentication control. This brings up a box allowing several different authentication settings:

- Anonymous access → allows all incoming users to access resources without authentication; users will run with the IUSR_<computer> account (or whatever is specified by the *Edit…* button on the Anonymous access section)
- Basic authentication → Base-64 encoded ID and password (in the form ID:Password) are sent to the web server for authentication
- Integrated Windows authentication → uses NTLM or Kerberos to authenticate a user; this allows seamless authentication by passing the security token of a user that is logged into a Windows domain
- Digest authentication → this can be utilized in a Windows domain environment; it provides basic encryption but is not well supported by web browsers

First let us configure the web-tier servers. By default, IIS allows anonymous access, so nothing special needs to be configured for the *VirtualBank* Internet server. For the Intranet, anonymous access needs to be turned off and Integrated Windows authentication needs to be checked.

Lastly, the mid-tier needs to be configured to use Basic Authentication to authenticate the SOAP call. As stated before, Basic Authentication takes advantage of HTTP header variables to pass an ID and password via a Base-64 encoded string to the web server. Do not be fooled though, Base-64 encoded does not mean encrypted. It is clear text and can easily be decoded to reveal the ID and password. Basic authentication should not be used without SSL. A Windows domain can be specified for the ID and password passed by Basic Authentication. This can be set by the *Edit…* button in the Basic Authentication section under the Authentication screen. For the *VirtualBank* app, local accounts will be used so no domain will be specified. Figure 11 shows the Authentication screen.



☐ Anonymous access

No user name/password required to access this resource.

Account used for anonymous access:

Authenticated access

For the following authentication methods, user name and password are required when
- anonymous access is disabled, or
- access is restricted using NTFS access control lists

☑ Basic authentication (password is sent in clear text)

Select a default domain:

☐ Digest authentication for Windows domain servers

☐ Integrated Windows authentication

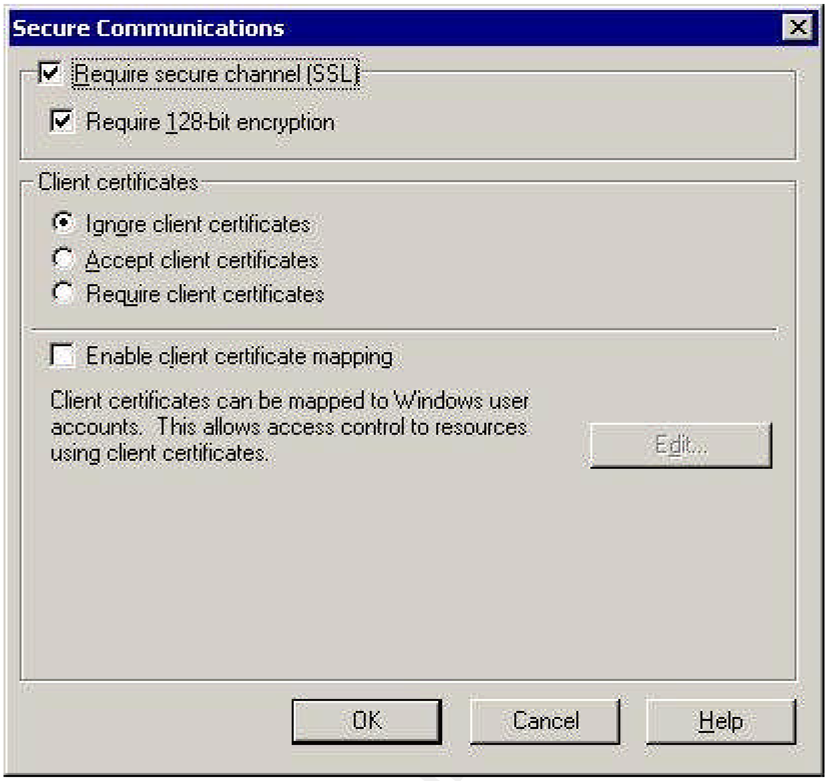
Figure 11

Virtual directory SSL

SSL is required to secure the Basic Authentication credentials sent from both Internet and Intranet web-tier servers to the mid-tier server. It is also required by the business requirements to protect the sensitive data transmitted over the wire by the *VirtualBank* application. SSL will need to be required at the virtual directories on all three servers.

IIS allows administrators to require that only HTTPS traffic be accepted by the server at either the Web Site level or at the virtual directory level. *VirtualBank* requires SSL so the virtual directory setup on all servers for *VirtualBank* needs to require SSL. This setting is located under the virtual directory → Properties → Directory Security tab, Secure Communications section *Edit…* button. This opens a window that displays configuration options for SSL. The very first option is *Require secure channel (SSL)*. This setting will require any connections to the virtual directory to use HTTPS. Any connections attempting to connect without SSL will receive a 403 HTTP error. This is desirable because it prevents all users from accidentally connecting via an unsecured port and passing credentials or sensitive information unencrypted.

In addition to requiring SSL, it is also possible to require the encryption strength be 128-bit. SSL also supports 40-bit encryption. This encryption is weak and can be broken fairly easily. E-commerce servers should use 128-bit SSL to protect data in transit. Figure 12 shows the *Secure Communications* screen.


Secure Communications

☑ Require secure channel (SSL)

☑ Require 128-bit encryption

Client certificates

● Ignore client certificates

○ Accept client certificates

○ Require client certificates

□ Enable client certificate mapping

Client certificates can be mapped to Windows user accounts. This allows access control to resources using client certificates.

Figure 12

The previous settings are the only ones that will need to be configured for the *VirtualBank* web application. The other settings defined in the Secure Communications window pertain to client certificates. These certificates can be used for authentication between client and server and eliminate the need for passwords.

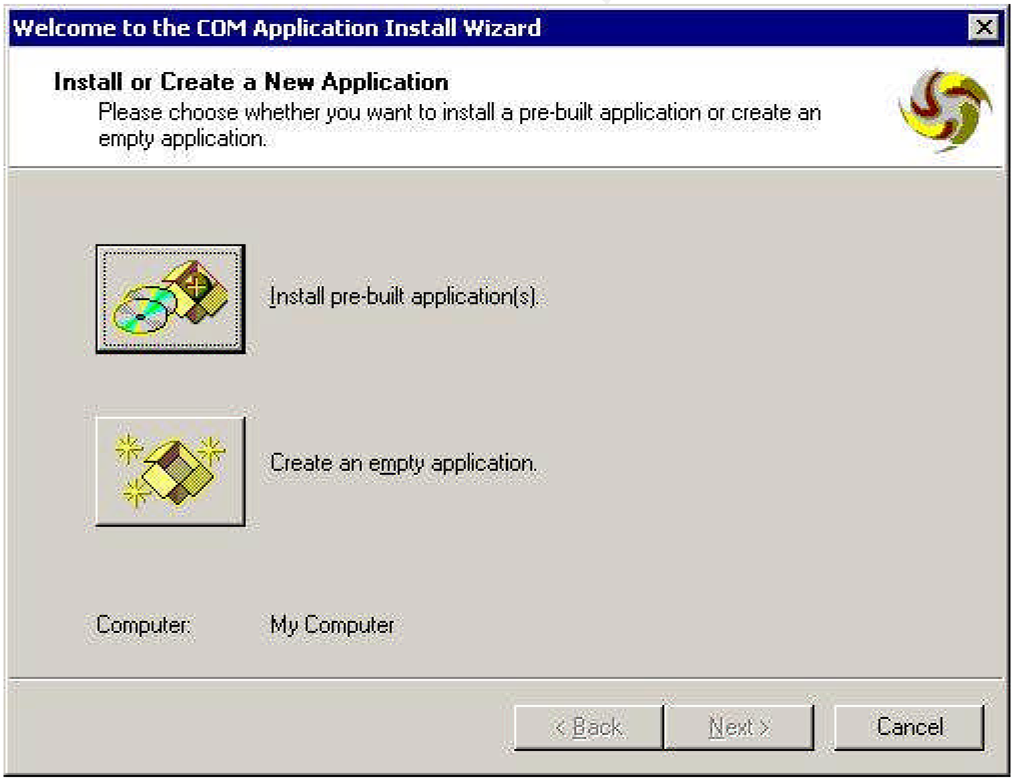
This completes the IIS setup, configuration, and security. The next section will discuss the final setup and security needed for the *VirtualBank* web application.

Installing and Securing the COM+ Application

Now that the application files have been installed and secured, IIS has been hardened and configured, the Web Site and virtual directory has been created and secured, and SSL has been deployed, the last piece to completing the deployment and securing of the *VirtualBank* web application is installing the COM+ pieces and securing them. The first piece will be the creation of the COM+ applications. The second piece will be the population of those applications with the appropriate components. Lastly, the applications and components will be secured using role-based security.

Creating the applications

Before COM+ can be used as an application and transaction server, the application must be created. This is easily accomplished through the Component Services snap-in. In the snap-in, the COM+ applications are grouped according to computer. All COM+ applications for the server are located under My Computer → COM+ Applications. In order to create a new application, highlight COM+ Applications and right click. Choose a New → Application. This executes the COM+ Application Install Wizard. The first screen (Figure 13) in the wizard presents a choice between using a pre-built application and creating an empty application. In order to use a pre-built application, an .msi file is needed. Since the *VirtualBank* web application uses .dll files, creating an empty application is the only option. The next screen will ask for the name of the application and a choice between which kind of application to create. The two choices of application are library and server. Library applications run in the process of the caller. This means that library applications will not do impersonation and in fact, will not have an identity of their own. Server applications, when invoked, will run in their own `dllhost.exe` process (or a shared server process depending on the value chosen for *Application Protection* under the Web Site or virtual directory). Unlike the library application, the server application will run as an identity. This can be specified as Interactive user, which means it will execute with the identity of a currently logged in user, or a specified user, in which case the process will always run as the specified user.



Welcome to the COM Application Install Wizard

Install or Create a New Application

Please choose whether you want to install a pre-built application or create an empty application.

Install pre-built application(s).

Create an empty application.

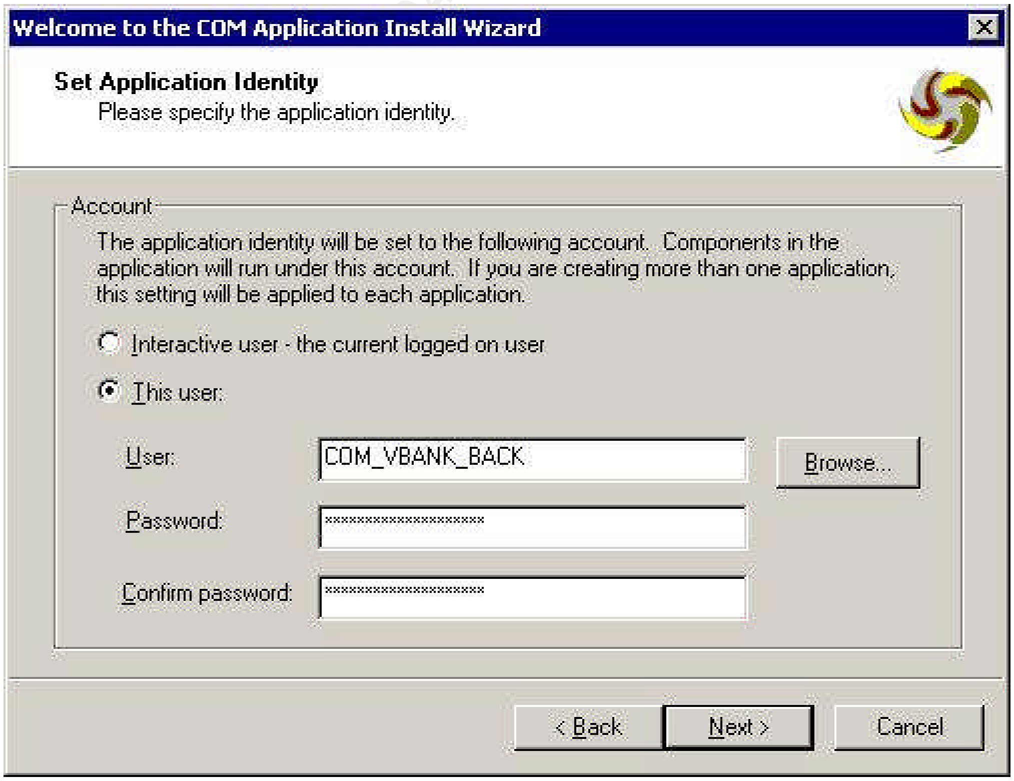
Computer: My Computer

Figure 13

The advantage to library applications is that they run considerably faster since they do not need to kick off a separate process. In addition, since they are running in the process that invoked them they can access any resources that the calling process ID can. The downfall is that since they cannot run as their own ID, they could be invoked by something running as SYSTEM and if the library application was exploited via buffer overflow or a similar exploit, the hacker would gain the context of the process. In this case, that would be SYSTEM; this is obviously bad.

Server applications sacrifice some performance in order to utilize better security. Since the server application can be specified to run as a non-privileged user account, buffer overflow type exploits result in a process running with a non-privileged context to be exploited. The end result is better security since the hacker is limited to working with a non-privileged account. Obviously this requires COM+ administrators to create accounts with no privilege for the applications to run under. Best practices would indicate that each application should run with a unique account. This prevents all applications from being vulnerable if an account is compromised. The *Interactive user* should be avoided because if an administrator is logged into the server when the application runs, it will run as that ID. This defeats the advantage of a server application. Also, if no one is logged in, the application will fail. Using a specific account allows the application to run without having anyone logged into the server.

VBank and *VBankBackend* will be running as server applications with respective account IDs of COM_VBANK and COM_VBANK_BACK. Those accounts should be added on the third screen of the wizard with their respective passwords (see Figure 14). *SoapUtils* will run as a library application and will not require any identity configuration in the wizard.



Welcome to the COM Application Install Wizard

Set Application Identity

Please specify the application identity.

Account

The application identity will be set to the following account. Components in the application will run under this account. If you are creating more than one application, this setting will be applied to each application.

○ Interactive user - the current logged on user

● This user:

User: COM_VBANK_BACK

Password: ********************

Confirm password: ********************

Figure 14

Populating the components

With the applications created, it is time to populate them with components from the .dll files. Right-clicking on the newly created application and selecting New → Component, will bring up the *COM+ Component Install Wizard*. Through this wizard components can be added to the application. The first screen will ask whether to install new components, install components that are already registered (through REGSVR32), or install new event classes. All of the applications in the *VirtualBank* application will use the *Install new components* option. Selecting *the Install new components* button will open a browse window where the .dll file containing the components for the application should be opened. Selecting the .dll will populate *Components found* list for the application. This is all that is needed to add the components to the applications. The following shows the COM+ application and corresponding .dll for the *VirtualBank* web application:

- *VBank* → `VBank.dll`
- *VBankBackend* → `VBankBackend.dll`
- *SoapUtils* → `SoapUtils.dll`

Implementing the COM+ security

The applications have now been created in COM+. By default, applications start off with security checking turned off. This means that any process can make a call and initiate a component in an application. This is particularly bad if the application has components which can make business transactions and update critical information in a database. The authorization for the security checks is based on the roles defined for the application. Roles will be discussed a little later.

First, consider the two server applications in the *VirtualBank* web application. The first step is to turn security checking on. None of the other security features in COM+ can be utilized without turning the checking on. COM+ performs security checks at three distinct locations: before a server process is started, when a call enters a process, and when a call enters an application. The first check is always done when a server process starts. If a process is already running (in the case of a library application initiating), then this check is not done. The second check is always completed when security checks are turned on. The third check occurs every time a call moves from one application to another within in the same process. The third check is determined by the *Security level* setting under the COM+ application Properties → Security tab. If this is set to "Perform access checks only at the process level," then the third check will not occur. If it is set to "Perform access checks at the process and component level," then this check will occur. As long as calls between components stay within the context of a single process, then the third check is all that occurs between applications. If a component makes a call to a separate process, all three calls (or two if the *Security level* is set to only check at the process level) will be performed.

Moving further down the screen of Properties → Security under the COM+ application, there is a listbox that defines the *Authentication level for calls* setting. This setting is only defined for server applications. It defines how often the application requests authentication from the caller. The setting options are:

- None → never ask for authentication (this option effectively disables authorization checks as well)
- Connect → only ask for authentication when a connection occurs (the first call to the application)
- Call → ask for authentication with each separate call to a component in the application
- Packet → ask for authentication with each packet sent
- Packet Integrity → perform a check for integrity, via checksum, of the packet data in addition to checking authentication with each packet sent
- Packet Privacy → encrypt the data in each packet in addition to the integrity check and checking authentication with each packet sent

These settings have more performance impact as the authentication level moves from “None” to “Packet Privacy.”

The last setting for server applications on the Properties → Security tab under the application is Impersonation level. In MTS there were only two impersonation levels defined: Identity and Impersonate. In COM+ two additional impersonation levels have been defined: Anonymous and Delegate. Impersonation is the ability for the server application to access resources using the security context of the caller process. The goal is to allow the server application the ability to access resources in much the same manner as a library application would, using the process ID of the caller. Anonymous means that the identity of the caller is ignored and never added to the context of the server application. Identity means that the identity is added to the context, but no impersonation is taking place. Impersonate means that the identity is added to the security context and the server application will access resources on the local server using the context of the caller. Delegate is the ultimate use of impersonation and means that the server application will make all calls local and over the network using the context of the caller. Much of this functionality must be coupled with programmatic settings in the components to make full use of the impersonation. In addition, delegation has several additional requirements:

- The account of the server application must have the “Trusted for delegation” property set
- The account of the caller cannot have the “Account is sensitive and cannot be delegated” property set
- Both accounts must be member of the same Windows® 2000 domain

The settings that have been mentioned thus far are set purely at the application level. COM+ also enables authorization checks to be set at the component, interface, and method levels. As mentioned before, calls between components within a single application will not have any authorization checks performed since all components in an application are running with the same security context. Component, interface, and method security is defined in the Properties → Security tab of each, respectively. Roles defined for the application can be assigned to individual components, interfaces, or methods. Authorization checks must be defined at all levels above any one of these in order for the caller to be authorized.

The effective security settings for each of the *VirtualBank* server applications are as follows:

- *VBank*
  - Enforce security checks → ON
  - Perform access checks at the process and component levels → ON
  - Authentication level → Packet Integrity
  - Impersonation level → Impersonate
  - Component level security → OFF for all
  - Interface level security → OFF for all
  - Method level security → OFF for all

- *VBankBackend*
  - Enforce security checks → ON
  - Perform access checks at the process and component levels → ON
  - Authentication level → Packet Integrity
  - Impersonation level → Impersonate
  - Component level security → ON for all
  - Interface level security → OFF for all
  - Method level security → OFF for all

Notice that component level checking has been turned on for all components within the *VBankBackend* application. This will enable the role-based security to properly authorize Customers, Tellers, and Managers to the appropriate components for their respective functionality. The functionality of VBank is purely to authenticate and pass on the information to the backend. For this reason, application security is ample. The `IsCallerInRole()` programmatic check is still successful. Since security checks have now been turned on, no one will be able to access the components unless roles are defined and set appropriately to the components. The roles for each application and the appropriate component security are as follows:

VBank Roles

Role	Members
Customer	IUSER_<computer>
Manager	VBANK_MGR_DLG

Teller	VBANK_TLR_DLG

VBankBackend Roles

Role	**Members**
Customer	SOAP_VBANK_CUST_L
Manager	SOAP_VBANK_MGR_L
Teller	SOAP_VBANK_TLR_L

VBankBackend

Component	**Roles assigned**
VBankBackend.authorizeLoan	Manager
VBankBackend.Deposit	Teller
VBankBackend.Withdrawal	Teller, Customer
VBankBackend.retrieveBalance	Teller, Customer, Manager

The server applications have been secured, so it is time to secure the library application. In MTS, there was no way to secure a library package. Library packages would run within the same process of the caller and assume the security context associated with that process. No security checks would be performed. In COM+ however, security checks are possible even though the library application is running in the same process as the caller.

The application Propeties → Security tab once again controls the security for the application. The security settings are a bit different here than with server applications. The Authorization and Security level are the same, but library applications cannot do any impersonation and can only specify whether to perform authentication or not. They are, by default, specified to not perform access checks.

Since the *SoapUtils* component merely makes SOAP calls to a specified service using a specific ID and password that it receives from the caller, it does not need to be secured. It will run in the context of the process that invokes it and does not need to perform any security checks. Since no security checks are being performed, no roles need to be defined for *SoapUtils*.

The COM+ security is now setup. The entire *VirtualBank* application has been setup and secured from files to IIS to COM+. This process has been a complicated one and definitely labor intensive. The goal of the next section is to reduce the confusion and labor providing a quick and simple automated solution for administrators to execute when setting up the *VirtualBank* application.

Scripting the Solution

The purpose of this section is to give a brief supplement to the scripts which are added at the end of this paper. The goal was to come up with a scripted solution which would perform the majority of the configuration and security setup required in the solution outlined above. A few pieces, such as the SSL certificate installation and the IIS Lockdown tool execution, are not scripted and need to be performed manually.

The scripting will utilize batch files when possible to increase performance and ease of coding. VBScript will be used when batch scripting is inadequate. The code will have comments for each section; much of the documentation of what each script does will be included in those comments. The scripts are partitioned according to functionality to improve readability of the code and simplify debugging. Each script will perform environment checks to implement only the changes needed in the server it is run on (Mid-Tier, Internet Web-Tier, or Intranet Web-Tier).

The final scripted solution will contain a number of scripts which can be applied separately if desired, however there will also be a single "driver" script which can be executed to kick off each of the other scripts in the proper order. The solution will be setup so that a single CD will contain all the scripts and source files for the *VirtualBank* application. Administrators need only run the driver script and all scripts will run. The .dll files obviously cannot be included but all script file source is included at the end of the paper.

Scripting the ID creation
This portion will be handled primarily using `local_ids.bat` with the `NET USER` command. The `NET USER` command will create all the local IDs. The `NET LOCALGROUP` will then create the local groups and populate them with local IDs.

Domain accounts will already be in place for the users. A single VBS, `domain_grps.vbs`, will handle creating the global groups, domain local groups, and populating the global groups into the domain local groups.

Scripting the file install and security
Creating the directory structure, installing the files, and securing the files will all be handled by `%SERVER%_files.bat`. This means each server type (INTRA, INTER, and MIDTIER) will have its own designated file. The `MD` command will create directories. The `COPY` command and `CACLS` executable will be used to copy the files from the CD and apply the proper permissions.

Scripting IIS setup
The IIS setup will all be contained within two scripts: `website_config.vbs` and `vdir_create.vbs`. Both scripts make changes directly to the IIS Metabase. Each script is well documented with the settings that are being changed.

Scripting the COM+ application install
The application installs are relatively easy scripts that utilize the COMAdmin object. Each application has a corresponding installation script. They are: `vbank_app_create.vbs` for the *VBank* application,

`vbankback_app_create.vbs` for the *VBankBackend* application, and `soaputils_app_create.vbs` for the *SoapUtils* application. The roles for each application are defined in these install scripts as well.

Scripting the COM+ security

The final scripts will be the COM+ security scripts for the *VBank* and *VBankBackend* applications; `vbank_app_sec.vbs` sets the *VBank* application security and `vbankback_app_sec.vbs` sets the *VBankBackend* application and component security. These scripts will set the identity and password of the applications as well as the impersonation level, authentication level, access checking level, and access checks flag at the application level. `vbankback_app_sec.vbs` sets the component security for each of the components in *VBankBackend*. The scripts also utilize the COMAdmin object to set the properties at both the application and component levels.

Testing

Historically, security testing has been vastly under-utilized. The majority of the time, application testing only ensures that the correct ID has access, not that unauthorized accounts cannot perform the actions. It is impossible to test all potential combinations of IDs and passwords in the application flow. A good way to verify the security is to go into the interfaces and perform a check out of the security settings. This involves knowing exactly what security should exist and verifying that only those settings are in place. A thorough security test plan will include some end to end application tests and the following manual verification tests:

Test Case 1

Verify that the .ASP and .DLL files on the Presentation servers have the correct ACLs as defined in the solution.

- `VBank.dll` (on DMZ servers) → Administrators:F, SYSTEM:F, Authenticated Users:RE
- `VBank.dll` (on Intranet servers) → Administrators:F, SYSTEM:F, VBANK_MGR_DLG:RE, VBANK_TLR_DLG:RE

Test Case 2

Verify that the .WSDL and .DLL files on the Mid-Tier servers have the correct ACLs as defined in the solution.

- `VBankBackend.dll` → Administrators:F, SYSTEM:F, SOAP_VBANK_L:R
- `authorizeLoan.wsdl` and `authorizeLoan.wsml` → Administrators:F, SYSTEM:F, SOAP_VBANK_MGR_L:R
- `Deposit.wsdl` and `Deposit.wsml` → Administrators:F, SYSTEM:F, SOAP_VBANK_TLR_L:R
- `Withdrawal.wsdl` and `Withdrawal.wsml` → Administrators:F, SYSTEM:F, SOAP_VBANK_TLR_L:R, SOAP_VBANK_CUST_L:R

- `retrieveBalance.wsdl` and `retrieveBalance.wsml` → Administrators:F, SYSTEM:F, SOAP_VBANK_L:R

Test Case 3
Verify that the following groups and IDs exist on the Mid-Tier servers:

- COM_VBANK_BACK
- SOAP_VBANK_TLR
- SOAP_VBANK_CUST
- SOAP_VBANK_MGR
- SOAP_VBANK_L → contains SOAP_VBANK_CUST, SOAP_VBANK_TLR, SOAP_VBANK_MGR
- SOAP_VBANK_TLR_L → contains SOAP_VBANK_TLR
- SOAP_VBANK_MGR_L → contains SOAP_VBANK_MGR
- SOAP_VBANK_CUST_L → contains SOAP_VBANK_CUST

Verify that the following account exists on the Presentation servers:

- COM_VBANK

Verify that the following domain groups exist on the Mid-tier Active Directory domain:

- VBANK_MGR_G
- VBANK_TLR_G
- VBANK_MGR_DLG → VBANK_MGR_G should be a member
- VBANK_TLR_DLG → VBANK_TLR_G should be a member

Test Case 4
Verify that the following settings are in place for IIS on the Presentation servers:

WebSite

- IIS logging directory → `%LOGDIR%\Web`
- Default Document → `Default.htm`
- Execute permissions → Scripts only; Read checked and log visits checked
- Application protection → Medium
- Local path → `%APPDIR%`
- Parent paths → OFF
- Enable Sessions → ON
- Session Timeout → 5 minutes

Virtual Directory

- Default Document → `index.asp`
- Authentication → Anonymous on Internet and Windows Integrated on Intranet
- Application protection → High
- Local path → `%APPDIR%\VBANK\website`

Test Case 5
Verify that the following settings are in place for IIS on the Mid-Tier servers:

WebSite

- IIS logging directory → `%LOGDIR%\Web`
- Default Document → `Default.htm`
- Execute permissions → Scripts only; Read checked and log visits checked
- Application protection → Medium
- Local path → `%APPDIR%`
- Parent paths → OFF
- Enable Sessions → ON
- Session Timeout → 5 minutes

Virtual Directory

- Default Document → `index.asp`
- Authentication → Basic Authentication
- Application protection → High
- Local path → `%APPDIR%\VBANK\website`
- .WSDL application mapping

Test Case 6
Verify that the following COM+ security settings are in place on the *VBank* COM+ application:

- Application Level
    - Identity → COM_VBANK
    - Password set
    - Impersonation Level → Impersonate
    - Authentication Level → Packet Integrity
    - Access checks set to Process and Component level
    - Access checks turned on

Test Case 7
Verify that the following COM+ security settings are in place on the *VBankBackend* COM+ application:

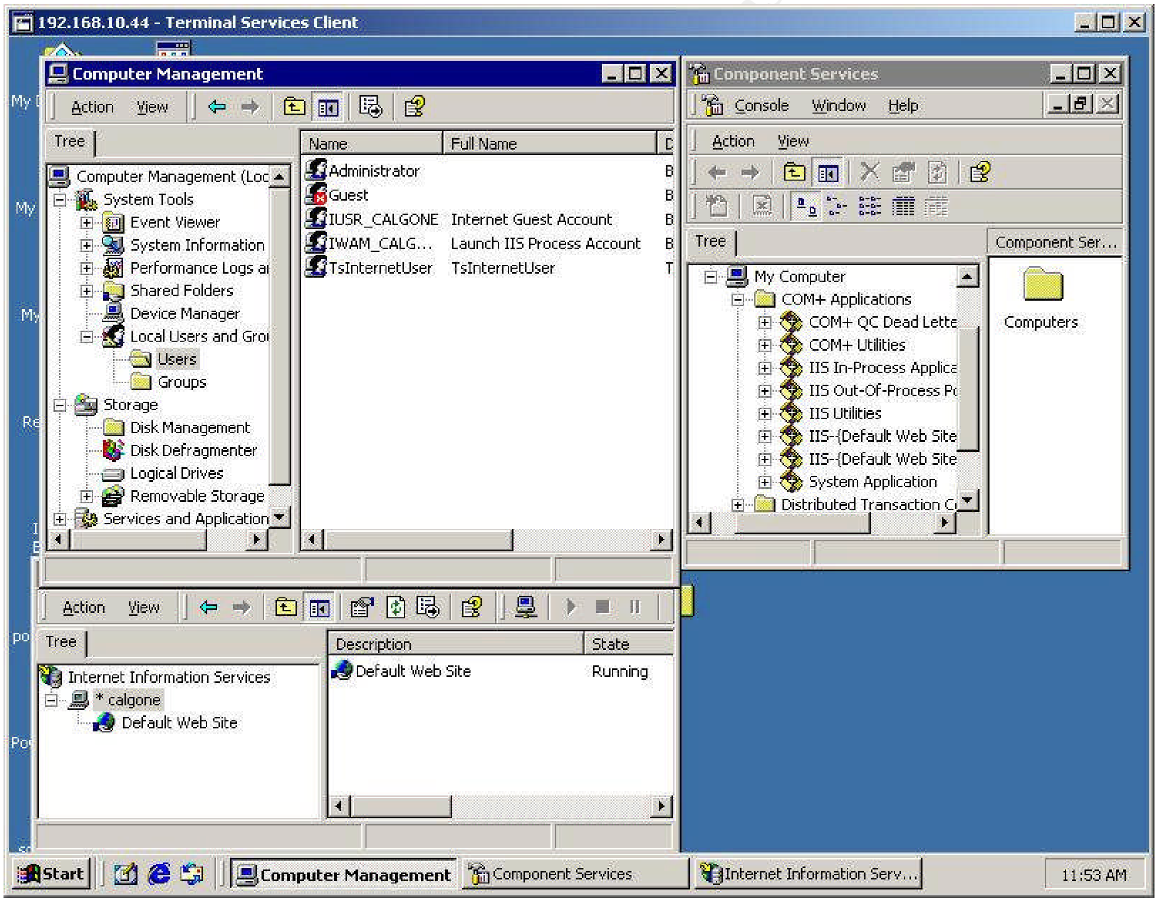
- Application Level
    - Identity → COM_VBANK_BACK
    - Password set
    - Impersonation Level → Impersonate
    - Authentication Level → Packet Integrity
    - Access checks set to Process and Component level
    - Access checks turned on
    - Customer role populated with SOAP_VBANK_CUST_L
    - Manager role populated with SOAP_VBANK_MGR_L
    - Teller role populated with SOAP_VBANK_TLR_L

- Component Level
  - Deposit → Component security on and Teller role set
  - Withdrawal → Component security on and Teller, Customer roles set
  - retrieveBalance → Component security on and Teller, Customer, Manager roles set
  - authorizeLoan → Component security on and Manager role set

Script Validation

This section will display the screenshots taken before, during, and after script execution. Due to the large number of settings that are set using the included scripts, only a portion of the settings can be validated using the following screen shots.

Before execution



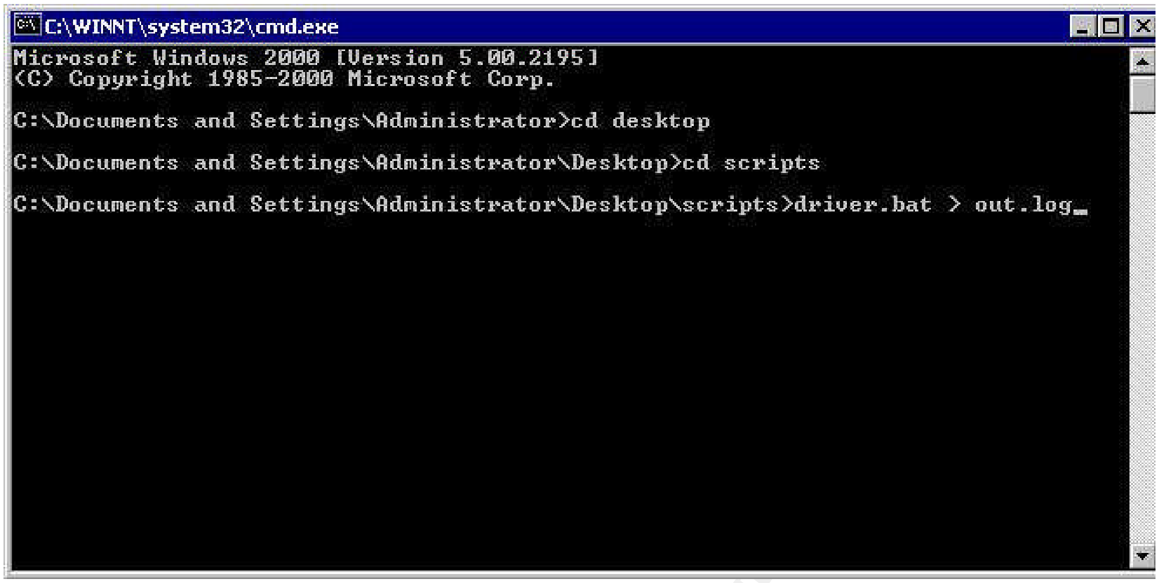
Name	Full Name
Administrator	
Guest	
IUSR_CALGONE	Internet Guest Account
IWAM_CALG...	Launch IIS Process Account
TsInternetUser	TsInternetUser

Description	State
Default Web Site	Running

Computer Management, COM+, and IIS

During execution

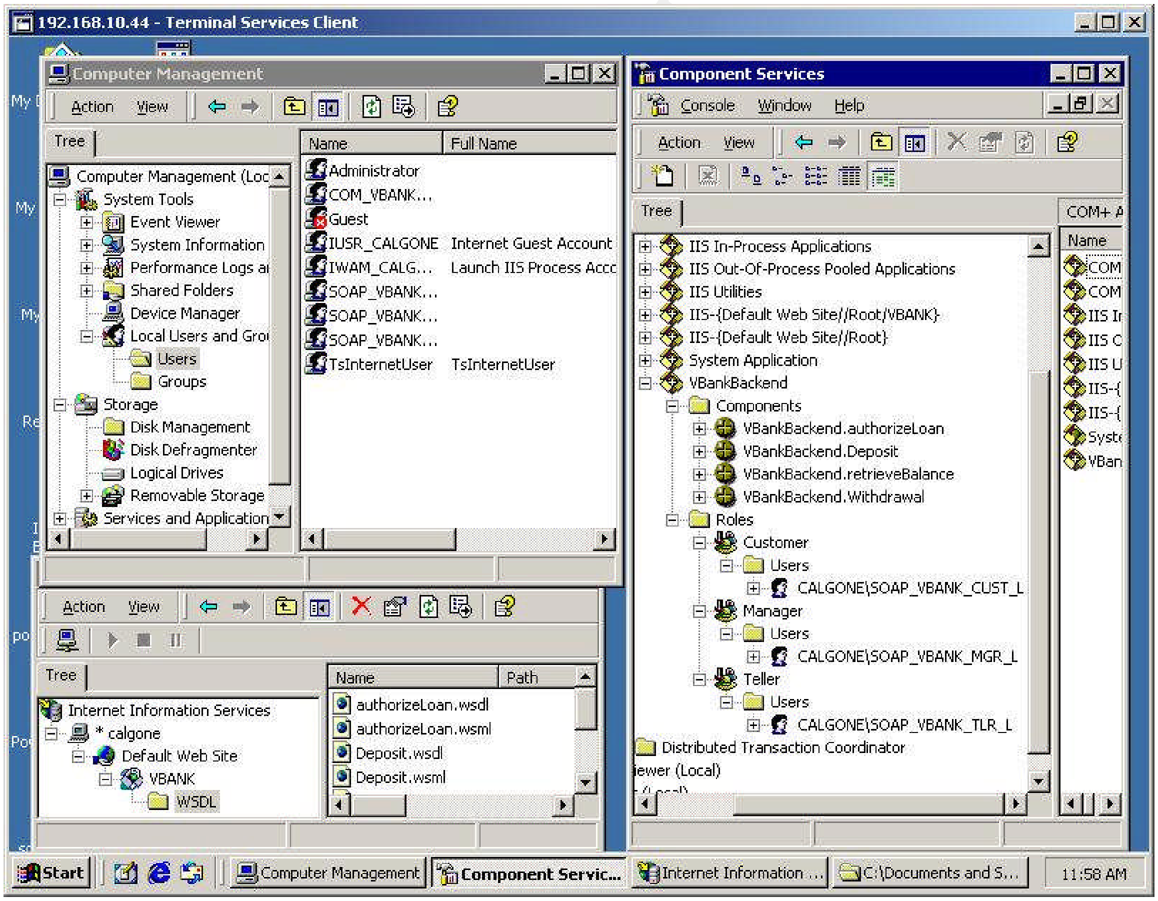
See `out.log` appendix for the script execution log output.



```
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

C:\Documents and Settings\Administrator>cd desktop

C:\Documents and Settings\Administrator\Desktop>cd scripts

C:\Documents and Settings\Administrator\Desktop\scripts>driver.bat > out.log_
```

Execution of driver.bat

After execution



Computer Management (Loc
- System Tools
  - Event Viewer
  - System Information
  - Performance Logs ar
  - Shared Folders
  - Device Manager
  - Local Users and Gro
    - Users
    - Groups
- Storage
  - Disk Management
  - Disk Defragmenter
  - Logical Drives
  - Removable Storage
- Services and Application

Name	Full Name
Administrator	
COM_VBANK...	
Guest	
IUSR_CALGONE	Internet Guest Account
IWAM_CALG...	Launch IIS Process Acc
SOAP_VBANK...	
SOAP_VBANK...	
SOAP_VBANK...	
TsInternetUser	TsInternetUser

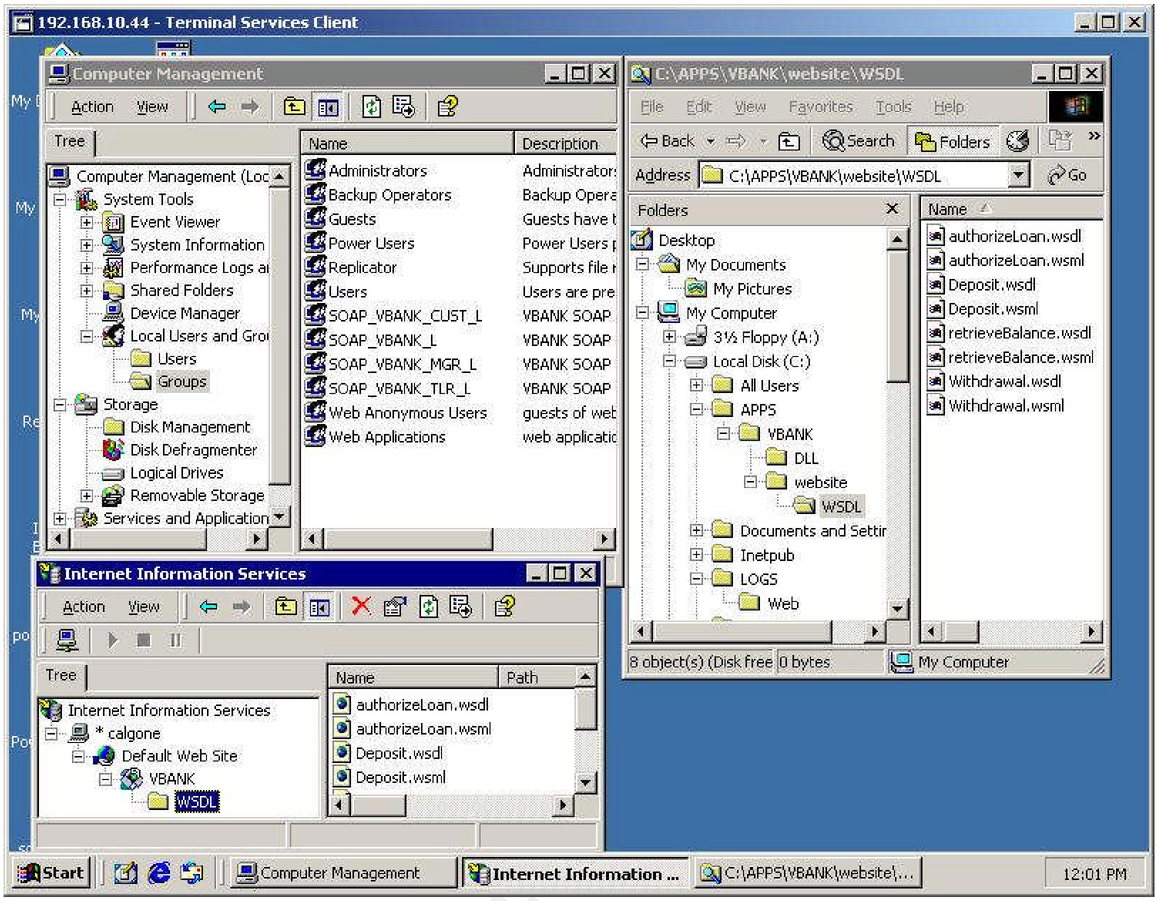
- IIS In-Process Applications
- IIS Out-Of-Process Pooled Applications
- IIS Utilities
- IIS-{Default Web Site//Root/VBANK}
- IIS-{Default Web Site//Root}
- System Application
- VBankBackend
  - Components
    - VBankBackend.authorizeLoan
    - VBankBackend.Deposit
    - VBankBackend.retrieveBalance
    - VBankBackend.Withdrawal
  - Roles
    - Customer
      - Users
        - CALGONE\SOAP_VBANK_CUST_L
    - Manager
      - Users
        - CALGONE\SOAP_VBANK_MGR_L
    - Teller
      - Users
        - CALGONE\SOAP_VBANK_TLR_L
- Distributed Transaction Coordinator

iewer (Local)

Internet Information Services
- * calgone
  - Default Web Site
    - VBANK
      - WSDL

Name	Path
authorizeLoan.wsdl	
authorizeLoan.wsml	
Deposit.wsdl	
Deposit.wsml	

Computer Management Users, COM+, and IIS



Name	Description
Administrators	Administrators
Backup Operators	Backup Opera
Guests	Guests have t
Power Users	Power Users p
Replicator	Supports file r
Users	Users are pre
SOAP_VBANK_CUST_L	VBANK SOAP
SOAP_VBANK_L	VBANK SOAP
SOAP_VBANK_MGR_L	VBANK SOAP
SOAP_VBANK_TLR_L	VBANK SOAP
Web Anonymous Users	guests of web
Web Applications	web applicatio

Computer Management Groups, Windows Explorer, and IIS

Challenges

There are challenges that still exist in the role-based security model as it is implemented by Microsoft in COM+. Currently, individual components cannot have their own unique ID. They are forced to run under the ID of the application to which they belong. This allows two components within the same application to access each other without any security check. This forces system designers to create separate applications whenever a security check is desired between components. At times this does not logically make sense, prevents the granularity needed, and can cause performance issues.

In addition, impersonation by COM+ does not provide a valuable auditing system. Each COM+ application must make OriginalCaller coding calls in order to find out which ID started the call chain. This is cumbersome and consistency cannot be guaranteed across all applications. In addition, if the call from the COM+ application makes a jump to a non-Microsoft backend product, the audit trail stops there.

Although much of the solution is scripted, it is not entirely self-executing; an administrator still must run the script. Ideally, a service such as Microsoft's SMS would allow for more seamless automation, particularly to a large number of servers. SMS can be configured to push issuances so that no interaction from an administrator is needed. In addition, the user passwords are currently in plain text in the script files. Using SMS, it is possible to create an executable which bundles all the scripts. This would prevent anyone from seeing the password in the script files.

Conclusion

The goal of this paper was to describe the steps necessary to securely deploy a web application developed and implemented using Microsoft's Windows, IIS, and COM+ platforms. Microsoft systems often take a beating from those individuals critical of its security issues. If nothing else, this paper has shown that it is possible to implement web applications securely using Microsoft products. Although they may appear more vulnerable "out of the box," no knowledgeable security administrator would run any system in an "out of the box" configuration. By configuring IIS, COM+ and Windows, the end result is a system that is just as secure by most standards as any Linux or UNIX system.

A second goal of the paper was to convey the importance of web applications. Web applications are an integral part of the business world today. Without them, businesses across the world would be unable to sell products, hire employees, or coordinate communication. Security has been slow to evolve with the web but great strides have been made in the past decade. Securing web applications is not so much a science as an art. The decision of whether to implement security controls in a web application and to what extent lies solely in the risk analysis and criticality of the data involved. Two identical applications may have completely different security requirements. Security administrators need to perform the necessary risk assessments with business sponsors in order to provide the level of security that is appropriate.

The last goal of this paper was to provide a basis for future readers to implement their own security controls. Learning by example is an excellent way to see concepts portrayed in a semi-real world environment. The use of the mock application was included to draw a connection between general security guidelines and actual business requirements. Most applications deployed follow some sort of business requirements or else there would be no reason to deploy them. Exploring the security controls implemented side-by-side with business and functionality requirements provides an invaluable example. The most important point to take from this paper, besides the Microsoft security solution, is that business requirements should always dictate the security controls, not the other way around.

Acronyms

***ACL** – Access Control List*

***ADO** – ActiveX Data Object*

***ASP** – Active Server Page*

***B2C** – Business to Customer*

***CA** – Certificate Authority*

***CGI** – Common Gateway Interface*

***COM** – Component Object Model*

***DLG** – Domain Local Group*

***DMZ** – De-militarized Zone*

***GPO** – Group Policy Object*

***HTTP** – Hyper Text Transport Protocol*

***HTTPS** – Hyper Text Transport Protocol Secured*

***IIS** – Internet Information Services*

***IP** – Internet Protocol*

***ISAPI** – Internet Server Application Program Interface*

***LDAP** – Lightweight Directory Access Protocol*

***MTS** – Microsoft Transaction Service*

***NTFS** – NT File System*

***NTLM** – NT LAN Manager (Hash)*

***OS** – Operating System*

***OSI** – Open System Interconnection*

***OU** – Organizational Unit*

***SMS** – Software Management System*

***SOAP** – Simple Object Access Protocol*

***SQL** – Structured Query Language*

***SSL** – Secure Sockets Layer*

***URI** – Uniform Resource Identifier*

***URL** – Uniform Resource Locator*

***WSDL** – Web Services Description Language*

***WSML** – Web Services Meta Language*

***XML** – Extensible Markup Language*

driver.bat Script

```
REM *** This file serves as the single click driver for driver.vbs
***
REM *** Users need only run this file to execute all scripts in
***
REM *** correct order
***
REM *** Created by: Derek Lawless 7/20/04
***
REM ***
***

cscript driver.vbs
```

driver.vbs Script

```
' True driver script
'
' Created by Derek Lawless 7/22/04
'

Dim Shell
Dim Exec
Dim Layer

If Environ("SERVER") = "MIDTIER" Then
   Layer = "BUS"
ElseIf Environ("SERVER") = "INTER" Then
   Layer = "WEB"
ElseIf Environ("SERVER") = "INTRA" Then
   Layer = "WEB"
End If

Set Shell = WScript.CreateObject("WScript.Shell")

Wscript.echo "Running ID creation script"
Wscript.echo "*******************************"

If Environ("SERVER") = "MIDTIER" Then
   Set Exec = Shell.Exec("cmd.exe /c ids.bat")
   Do While Not Exec.StdOut.AtEndofStream
      output = output & Exec.StdOut.ReadLine() & vbCrLf
   Loop
   Wscript.echo output
   output = ""
ElseIf Environ("SERVER") = "INTRA" Then
   Set Exec = Shell.Exec("cmd.exe /c cscript domain_grps.vbs")
   Do While Not Exec.StdOut.AtEndofStream
      output = output & Exec.StdOut.ReadLine() & vbCrLf
   Loop
   Wscript.echo output
   output = ""
End If
Wscript.echo "*******************************"

Wscript.echo "Running File install and security script"
Wscript.echo "*******************************"
Set Exec = Shell.Exec("cmd.exe /c " + Environ("SERVER") + "_files.bat")
Do While Not Exec.StdOut.AtEndofStream
      output = output & Exec.StdOut.ReadLine() & vbCrLf
Loop
Wscript.echo output
output = ""
Wscript.echo "*******************************"

Wscript.echo "Running Website Config script"
Wscript.echo "*******************************"
Set Exec = Shell.Exec("cmd.exe /c cscript website_config.vbs")
```

```
Do While Not Exec.StdOut.AtEndofStream
      output = output & Exec.StdOut.ReadLine() & vbCrLf
Loop
Wscript.echo output
output = ""
Wscript.echo "*******************************"

Wscript.echo "Running Virtual Directory creation script"
Wscript.echo "*******************************"
Set Exec = Shell.Exec("cmd.exe /c cscript vdir_create.vbs")
Do While Not Exec.StdOut.AtEndofStream
      output = output & Exec.StdOut.ReadLine() & vbCrLf
Loop
Wscript.echo output
output = ""
Wscript.echo "*******************************"

Wscript.echo "Running COM+ application creation scripts"
Wscript.echo "*******************************"
If Layer = "WEB" Then
   Set Exec = Shell.Exec("cmd.exe /c cscript vbank_app_create.vbs")
   Do While Not Exec.StdOut.AtEndofStream
      output = output & Exec.StdOut.ReadLine() & vbCrLf
   Loop
   Wscript.echo output
   output = ""
   Set Exec = Shell.Exec("cmd.exe /c cscript soaputils_app_create.vbs")
   Do While Not Exec.StdOut.AtEndofStream
      output = output & Exec.StdOut.ReadLine() & vbCrLf
   Loop
   Wscript.echo output
   output = ""
ElseIf Layer = "BUS" Then
   Set Exec = Shell.Exec("cmd.exe /c cscript vbankback_app_create.vbs")
   Do While Not Exec.StdOut.AtEndofStream
      output = output & Exec.StdOut.ReadLine() & vbCrLf
   Loop
   Wscript.echo output
   output = ""
End If
Wscript.echo "*******************************"

Wscript.echo "Running COM+ application security scripts"
Wscript.echo "*******************************"
If Layer = "WEB" Then
   Set Exec = Shell.Exec("cmd.exe /c cscript vbank_app_sec.vbs")
   Do While Not Exec.StdOut.AtEndofStream
      output = output & Exec.StdOut.ReadLine() & vbCrLf
   Loop
   Wscript.echo Layer
   Wscript.echo output
   output = ""
ElseIf Layer = "BUS" Then
   Set Exec = Shell.Exec("cmd.exe /c cscript vbankback_app_sec.vbs")
   Do While Not Exec.StdOut.AtEndofStream
      output = output & Exec.StdOut.ReadLine() & vbCrLf
   Loop
   Wscript.echo output
   output = ""
End If
Wscript.echo "*******************************"

Wscript.echo "Driver script finished."

Function Environ(var)
   Dim shell
   Set shell = wscript.CreateObject("Wscript.Shell")
   Environ = shell.ExpandEnvironmentStrings("%" + var + "%")
   Set shell = Nothing
End Function
```

domain_grps.vbs Script

```
' VBANK Domain group creation script
'
' Created by Derek Lawless 7/18/04
'
' Assume acme.org is the name of the internal domain

Dim Domain
Dim Group

' Get the domain object
Set Domain = GetObject("WinNT://acme.org")


'**************************************************************
'************** Create Manager global group *******************
'**************************************************************

' Create new group
Set Group = Domain.Create("group", "VBANK_MGR_G")

' Set group as global type
Group.GroupType = 2

' Set description
Group.Description = "Global group for Managers"

' Add users (this line of code commented out for mock application)
' Add more Group.Add statements for each user to add
' It is assumed that the user account is already created
' Group.Add "WinNT://acme.org/joemanager.User"

' Update group info
Group.SetInfo

'**************************************************************
'************** Create Teller global group ********************
'**************************************************************

' Create new group
Set Group = Domain.Create("group", "VBANK_TLR_G")

' Set group as global type
Group.GroupType = 2

' Set description
Group.Description = "Global group for Tellers"

' Add users (this line of code commented out for mock application)
' Add more Group.Add statements for each user to add
' It is assumed that the user account is already created
' Group.Add "WinNT://acme.org/janeteller.User"
' Update group info
Group.SetInfo

'**************************************************************
'************** Create Manager domain local group *************
'**************************************************************

' Create new group
Set Group = Domain.Create("group", "VBANK_MGR_DLG")

' Set description
Group.Description = "Domain Local group for Managers"

' Set group as domain local type
Group.GroupType = 4

' Update prior to adding group
Group.SetInfo

' Add the global group
Group.Add "WinNT://acme.org/VBANK_MGR_G"

' Update group info
Group.SetInfo

'**************************************************************
'************** Create Teller domain local group **************
'**************************************************************

' Create new group
Set Group = Domain.Create("group", "VBANK_TLR_DLG")

' Set description
Group.Description = "Domain Local group for Tellers"

' Set group as domain local type
Group.GroupType = 4

' Update prior to adding group
Group.SetInfo

' Add the global group
Group.Add "WinNT://acme.org/VBANK_TLR_G"

' Update group info
Group.SetInfo

' Update Domain info
Domain.SetInfo

' Cleanup memory
Set Domain = Nothing
Set Group  = Nothing
```

ids.bat Script

```
REM *** This file creates all local IDs for VBANK    ***
REM *** Created by: Derek Lawless 7/20/04            ***
REM ***                                              ***

echo *** Creating local IDs ***

if %SERVER%==INTER NET USER COM_VBANK "8yEf7d(3dsx09l" /ADD

if %SERVER%==INTRA NET USER COM_VBANK "8yEf7d(3dsx09l" /ADD

if %SERVER%==MIDTIER NET USER COM_VBANK_BACK "0kr!45xqzuUvt8" /ADD
if %SERVER%==MIDTIER NET USER SOAP_VBANK_CUST o*hkS#xtrP91!c /ADD
if %SERVER%==MIDTIER NET USER SOAP_VBANK_TLR xpCATr#tpq!(zs /ADD
if %SERVER%==MIDTIER NET USER SOAP_VBANK_MGR *ck#t*wmnGiO73 /ADD

if %SERVER%==MIDTIER NET LOCALGROUP SOAP_VBANK_L /COMMENT:"VBANK SOAP
group" /ADD
if %SERVER%==MIDTIER NET LOCALGROUP SOAP_VBANK_CUST_L /COMMENT:"VBANK
SOAP group" /ADD
if %SERVER%==MIDTIER NET LOCALGROUP SOAP_VBANK_TLR_L /COMMENT:"VBANK
SOAP group" /ADD
if %SERVER%==MIDTIER NET LOCALGROUP SOAP_VBANK_MGR_L /COMMENT:"VBANK
SOAP group" /ADD

if %SERVER%==MIDTIER NET LOCALGROUP SOAP_VBANK_CUST_L SOAP_VBANK_CUST
/ADD
if %SERVER%==MIDTIER NET LOCALGROUP SOAP_VBANK_TLR_L SOAP_VBANK_TLR
/ADD
if %SERVER%==MIDTIER NET LOCALGROUP SOAP_VBANK_MGR_L SOAP_VBANK_MGR
/ADD
if %SERVER%==MIDTIER NET LOCALGROUP SOAP_VBANK_L SOAP_VBANK_CUST /ADD
if %SERVER%==MIDTIER NET LOCALGROUP SOAP_VBANK_L SOAP_VBANK_TLR /ADD
if %SERVER%==MIDTIER NET LOCALGROUP SOAP_VBANK_L SOAP_VBANK_MGR /ADD

echo *** Done. ***
```

INTER_files.bat Script

```
REM *** This file installs files on the Internet     ***
REM *** Web-tier server and secures the files        ***
REM *** Created by: Derek Lawless 7/20/04            ***
REM ***                                              ***

echo "Creating directory structure for %SERVER% server..."
md %APPDIR%
md %APPDIR%\VBANK
md %APPDIR%\VBANK\DLL
md %APPDIR%\VBANK\website

md %LOGDIR%
md %LOGDIR%\Web

md %APPDIR%\WSDL

echo "Done."
echo ""
echo "Installing files on %SERVER% server..."

COPY source\index.asp %APPDIR%\VBANK\website
COPY source\SoapUtils.dll %APPDIR%\VBANK\DLL
COPY source\VBank.dll %APPDIR%\VBANK\DLL

echo "Done."

echo ""
echo "Setting file security..."

ECHO Y|CACLS %APPDIR%\VBANK\website\index.asp /C /P "Authenticated
Users:R" Administrators:F SYSTEM:F
ECHO Y|CACLS %APPDIR%\VBANK\DLL /C /P /T "Authenticated Users":R
Administrators:F SYSTEM:F
ECHO Y|CACLS %APPDIR%\WSDL /C /T /P COM_VBANK:R Administrators:F
SYSTEM:F

echo "Done."
```

INTRA_files.bat Script

```
REM *** This file installs files on the Intranet     ***
REM *** Web-tier server and secures the files        ***
REM *** Created by: Derek Lawless 7/20/04            ***
REM ***                                              ***

echo "Creating directory structure for %SERVER% server..."
md %APPDIR%
md %APPDIR%\VBANK
md %APPDIR%\VBANK\DLL
md %APPDIR%\VBANK\website

md %LOGDIR%
md %LOGDIR%\Web

md %APPDIR%\WSDL

echo "Done."
echo ""
echo "Installing files on %SERVER% server..."

COPY source\index.asp %APPDIR%\VBANK\website
COPY source\SoapUtils.dll %APPDIR%\VBANK\DLL
COPY source\VBank.dll %APPDIR%\VBANK\DLL

echo "Done."

echo ""
echo "Setting file security..."

ECHO Y|CACLS %APPDIR%\VBANK\website\index.asp /C /P VBANK_TLR_DLG:RE
VBANK_MGR_DLG:RE Administrators:F SYSTEM:F
ECHO Y|CACLS %APPDIR%\VBANK\DLL /C /T /P VBANK_TLR_DLG:RE
VBANK_MGR_DLG:RE Administrators:F SYSTEM:F
ECHO Y|CACLS %APPDIR%\WSDL /C /T /P COM_VBANK:R Administrators:F
SYSTEM:F

echo "Done."
```

MIDTIER_files.bat Script

```
REM *** This file installs files on the Mid-tier     ***
REM *** server and secures the files                 ***
REM *** Created by: Derek Lawless 7/20/04            ***
REM ***                                              ***

echo "Creating directory structure for %SERVER% server..."
md %APPDIR%
md %APPDIR%\VBANK
md %APPDIR%\VBANK\DLL
md %APPDIR%\VBANK\website

md %LOGDIR%
md %LOGDIR%\Web

md %APPDIR%\VBANK\website\WSDL

echo "Done."
echo ""
echo "Installing files on %SERVER% server..."

rem *** Install Mid-Tier files ***
COPY source\VBankBackend.dll %APPDIR%\VBANK\DLL
COPY source\authorizeLoan.ws* %APPDIR%\VBANK\website\WSDL
COPY source\Deposit.ws* %APPDIR%\VBANK\website\WSDL
COPY source\Withdrawal.ws* %APPDIR%\VBANK\website\WSDL
COPY source\retrieveBalance.ws* %APPDIR%\VBANK\website\WSDL

echo "Done."

echo ""
echo "Setting file security..."

ECHO Y|CACLS %APPDIR%\VBANK\DLL /C /T /P SOAP_VBANK_L:R
Administrators:F SYSTEM:F
ECHO Y|CACLS %APPDIR%\VBANK\website\WSDL /C /T /P SOAP_VBANK_L:R
Administrators:F SYSTEM:F
ECHO Y|CACLS %APPDIR%\VBANK\website\WSDL\authorizeLoan.ws* /C /T /P
SOAP_VBANK_MGR_L:R Administators:F SYSTEM:F
ECHO Y|CACLS %APPDIR%\VBANK\website\WSDL\Deposit.ws* /C /T /P
SOAP_VBANK_TLR_L:R Administators:F SYSTEM:F
ECHO Y|CACLS %APPDIR%\VBANK\website\WSDL\Withdrawal.ws* /C /T /P
SOAP_VBANK_TLR_L:R SOAP_VBANK_CUST_L:R Administators:F SYSTEM:F
ECHO Y|CACLS %APPDIR%\VBANK\website\WSDL\retrieveBalance.ws* /C /T /P
SOAP_VBANK_L:R Administators:F SYSTEM:F

echo "Done."
```

website_config.vbs Script

```
' Default WebSite Configuration Script
'
' Modified from script by James R. Fallon
' Modified by Derek Lawless on 07/18/04
'

Dim vDefWebSite, vW3SVC

' First setup logfile location because it must be defined at server
level
Set vW3SVC = GetObject("IIS://LocalHost/W3SVC")

vW3SVC.LogFileDirectory = Environ("LOGDIR") + "\Web"

vW3SVC.SetInfo

' Cleanup memory
Set vW3SVC = Nothing

' Modifies Default Web Site.
' To set or modify settings for a different Web site,
' modify the GetObject calls to connect to a site with a
' different instance ID.  1 is the Default site, 2 is the
' Administration site, and as more sites are added to a
' server, each is associated with an instance ID.

Set vDefWebSite = GetObject("IIS://LocalHost/W3SVC/1/Root")

vDefWebSite.DefaultDoc = "Default.htm"

vDefWebSite.AuthFlags = 4
'  Authentication
'  AuthFlags = 4 (Challenge Response)
'  AuthFlags = 5 (Allow Anonymous)
'  AuthFlags = 6 (Basic Authentication)

vDefWebSite.AccessFlags = 513
' AccessFlags = Execute Permissions + Read check box + Write check box
+ Script Source box
' Execute permissions None                   = 0
'                     Script Only            = 512
'                     Script and Executables = 516
' Read box            Unchecked = 0
'                     checked   = 1
' Write box           unchecked = 0
'                     checked   = 1
' Script source box   unchecked = 0
'                     checked   = 16
'

vDefWebSite.AppIsolated = 2
' AppIsolated (Application Protection) = 0 (low (IIS Process))
'                                    1 (high (Isolated))
'                              2 (Medium (pooled))(Our Default)

' Set local path
VDefWebSite.Path = Environ("APPDIR")

' Parent paths off
vDefWebSite.AspEnableParentPaths = "False"

' Enable sessions
vDefWebSite.AspAllowSessionState = "True"
vDefWebSite.AspSessionTimeout = 5

' Disable indexing
vDefWebSite.ContentIndexed = "False"

' Update Metabase
vDefWebSite.SetInfo

' Cleanup memory
Set vDefWebSite = Nothing

Function Environ(var)
   Dim shell
   Set shell = wscript.CreateObject("Wscript.Shell")
   Environ = shell.ExpandEnvironmentStrings("%" + var + "%")
   Set shell = Nothing
End Function
```

vdir_create.vbs Script

```
'VBank virtual directory creation and configuration script
'
' Created by Derek Lawless 7/18/04
'

Option Explicit

Dim WebDirName
Dim WebDirPath
Dim allowAnonymousAccess
Dim enableChallengeResponse
Dim allowBasicAuthentication
Dim WSHShell
Dim vRoot
Dim vDir
Dim sMap

'Configuration variables
WebDirName = "VBANK"
WebDirPath = Environ("APPDIR") + "\VBANK\website"

' This variable will collect output from this script for debugging
' Add logic to check SERVER variable to make configuration changes
' per INTER, INTRA, or MIDTIER (espcially for authentication)

' Set authentication per environment
If Environ("SERVER") = "MIDTIER" Then
   allowBasicAuthentication = "True"
   allowAnonymousAccess = "False"
   enableChallengeResponse = "False"
Else
   If Environ("SERVER") = "INTER" Then
      allowBasicAuthentication = "False"
      allowAnonymousAccess = "True"
      enableChallengeResponse = "False"
   Else
      allowBasicAuthentication = "False"
      allowAnonymousAccess = "False"
      enableChallengeResponse = "True"
   End If
End If

Set WSHShell = CreateObject("WScript.shell")
Set vRoot = GetObject("IIS://LocalHost/W3SVC/1/Root")

'Create the new virtual directory
Set vDir = vRoot.Create("IIsWebVirtualDir", WebDirName)

'Set the new virtual directory settings
vDir.AccessFlags = 513
vDir.EnableDefaultDoc = "True"
vDir.EnableDirBrowsing = "False"
vDir.DefaultDoc = "index.asp"
vDir.AppPackageName = WebDirName
vDir.AppFriendlyName = WebDirName

vDir.AuthNTLM = enableChallengeResponse
vDir.AuthAnonymous = allowAnonymousAccess
vDir.AuthBasic = allowBasicAuthentication

vDir.ASPScriptLanguage = "VBScript"
vDir.ASPSessionTimeout = 5
vDir.ASPScriptTimeout = 90
vDir.AspAllowSessionState = "True"
vDir.AccessSSL = "True"
vDir.AccessSSL128 = "True"
vDir.AppCreate False
vDir.Path = WebDirPath


' Add script mappings here
sMap = vRoot.Get("ScriptMaps")

' Add WSDL mapping on MIDTIER server
If Environ("SERVER") = "MIDTIER" Then
   sMap(UBound(sMap)) = ".wsdl,C:\Program Files\Common
Files\MSSoap\Binaries\SOAPIS30.dll,5,GET,HEAD,POST"
End If

vDir.Put "ScriptMaps", sMap

' Update metabase
vDir.SetInfo

' Cleanup memory
Set vDir = Nothing


Function Environ(var)
   Dim shell
   Set shell = wscript.CreateObject("Wscript.Shell")
   Environ = shell.ExpandEnvironmentStrings("%" + var + "%")
   Set shell = Nothing
End Function
```

soaputils_app_create.vbs Script

```
' SoapUtils COM+ application creation and configuration script
'
' Created by Derek Lawless 7/18/04
'

Dim Catalog
Dim AppCollection
Dim SoapUtils

' Get catalog
Set Catalog = CreateObject("COMAdmin.COMAdminCatalog")
' Get the Applications collection
Set AppCollection = Catalog.GetCollection("Applications")
' Add a new COM+ application
Set SoapUtils = AppCollection.Add

' Set name of new COM+ application
SoapUtils.Value("Name") = "SoapUtils"

' Set as a library application
SoapUtils.Value("Activation") = 0

' Save changes to collection
AppCollection.SaveChanges

Catalog.InstallComponent "SoapUtils", Environ("APPDIR") +
"\VBANK\DLL\SoapUtils.dll","",""

Function Environ(var)
   Dim shell
   Set shell = wscript.CreateObject("Wscript.Shell")
   Environ = shell.ExpandEnvironmentStrings("%" + var + "%")
   Set shell = Nothing
End Function
```

vbank_app_create.vbs Script

```
' VBank COM+ application creation and configuration script
'
' Created by Derek Lawless 7/18/04
'

Dim Catalog
Dim AppCollection
Dim VBank

' Get catalog
Set Catalog = CreateObject("COMAdmin.COMAdminCatalog")
' Get the Applications collection
Set AppCollection = Catalog.GetCollection("Applications")
' Add a new COM+ application
Set VBank = AppCollection.Add

' Set name of new COM+ application
VBank.Value("Name") = "VBank"

' Save changes to collection
AppCollection.SaveChanges

Catalog.InstallComponent "VBank", Environ("APPDIR") +
"\VBANK\DLL\VBank.dll","",""

Function Environ(var)
   Dim shell
   Set shell = wscript.CreateObject("Wscript.Shell")
   Environ = shell.ExpandEnvironmentStrings("%" + var + "%")
   Set shell = Nothing
End Function
```

vbank_app_sec.vbs Script

```
' VBank COM+ application security script
'
' Created by Derek Lawless 7/20/04
'
Dim Catalog
Dim AppCollection
Dim App
Dim VBank
Dim Users
Dim Roles
Dim Manager,Teller,Customer
Dim User1
Dim CompCollection
Dim Comp, CompRole1, CompRole2, CompRole3

' Get catalog
Set Catalog = CreateObject("COMAdmin.COMAdminCatalog")
' Get the Applications collection
Set AppCollection = Catalog.GetCollection("Applications")
' Get the application
AppCollection.Populate

' Grab VBank
For Each App in AppCollection
    If App.Name = "VBank" Then
        Exit For
    End If
Next

'***************************************
' First set application level security
'***************************************

' Set Process and Component Level checking
App.Value("ApplicationAccessChecksEnabled") = True
App.Value("AccessChecksLevel") = 1

' Set Identity of application
App.Value("Identity") = "COM_VBANK"
App.Value("Password") = "8yEf7d(3dsx09l"

' Set Impersonation level (1-4), 3 is Impersonate
App.Value("ImpersonationLevel") = 3

' Set Authentication level (1-6), 5 is for Packet Integrity
App.Value("Authentication") = 5

'*********
' ROLES
'*********

Set Roles = AppCollection.GetCollection("Roles", App.Key)
Set Manager = Roles.Add
Set Teller  = Roles.Add
Set Customer = Roles.Add

Manager.Value("Name")  = "Manager"
Teller.Value("Name")   = "Teller"
Customer.Value("Name") = "Customer"

' Update roles
Roles.SaveChanges

Set Users = Roles.GetCollection("UsersInRole", Manager.Key)

Set User1 = Users.Add
User1.Value("User") = "Test"
Users.SaveChanges

Set Users = Roles.GetCollection("UsersInRole", Customer.Key)

Set User1 = Users.Add
User1.Value("User") = "IUSR_" + Environ("COMPUTERNAME")
Users.SaveChanges

Set Users = Roles.GetCollection("UsersInRole", Teller.Key)

Set User1 = Users.Add
User1.Value("User") = "joe"
Users.SaveChanges

Roles.SaveChanges

'Update Application collection
AppCollection.SaveChanges

'***************************************
'
' Set Turn component checks off
'
Set CompCollection = AppCollection.GetCollection("Components", App.Key)

' Populate the components
CompCollection.Populate

' Add roles for components
For Each Comp in CompCollection
    If Comp.Name = "VBank.UserAuth" Then
       ' Enable component checks
       Comp.Value("ComponentAccessChecksEnabled") = False

       ' Save updates
       CompCollection.SaveChanges
    ElseIf Comp.Name = "VBank.UserReg" Then
      ' Disable component checks
      Comp.Value("ComponentAccessChecksEnabled") = False

      ' Save updates
       CompCollection.SaveChanges
    End If
Next

'****************************************
Function Environ(var)
   Dim shell
   Set shell = wscript.CreateObject("Wscript.Shell")
   Environ = shell.ExpandEnvironmentStrings("%" + var + "%")
   Set shell = Nothing
End Function
```

vbankback_app_create.vbs Script

```
' VBankBackend COM+ application creation and configuration script
'
' Created by Derek Lawless 7/18/04
'

Dim Catalog
Dim AppCollection
Dim VBankBack

' Get catalog
Set Catalog = CreateObject("COMAdmin.COMAdminCatalog")
' Get the Applications collection
Set AppCollection = Catalog.GetCollection("Applications")
' Add a new COM+ application
Set VBankBack = AppCollection.Add

' Set name of new COM+ application
VBankBack.Value("Name") = "VBankBackend"

' Save changes to collection
AppCollection.SaveChanges

Catalog.InstallComponent "VBankBackend", Environ("APPDIR") +
"\VBANK\DLL\VBankBackend.dll","",""

Function Environ(var)
   Dim shell
   Set shell = wscript.CreateObject("Wscript.Shell")
   Environ = shell.ExpandEnvironmentStrings("%" + var + "%")
   Set shell = Nothing
End Function
```

vbankback_app_sec.vbs Script

```
' VBankBackend COM+ application security script
'
' Created by Derek Lawless 7/20/04
'

Dim Catalog
Dim AppCollection
Dim App
Dim VBankBackend
Dim Users
Dim Roles
Dim Manager, Teller, Customer
Dim User1
Dim CompCollection
Dim Comp
Dim CompRole1, CompRole2, CompRole3

' Get catalog
Set Catalog = CreateObject("COMAdmin.COMAdminCatalog")
' Get the Applications collection
Set AppCollection = Catalog.GetCollection("Applications")
' Get the application
AppCollection.Populate

' Grab VBankBackend
For Each App in AppCollection
    If App.Name = "VBankBackend" Then
        Exit For
    End If
Next

'**************************************
' First set application level security
'**************************************

' Set Process and Component Level checking
App.Value("ApplicationAccessChecksEnabled") = True
App.Value("AccessChecksLevel") = 1

' Set Identity of application
App.Value("Identity") = "COM_VBANK_BACK"
App.Value("Password") = "0kr!45xqzuUvt8"

' Set Impersonation level (1-4), 3 is Impersonate
App.Value("ImpersonationLevel") = 3

' Set Authentication level (1-6), 5 is for Packet Integrity
App.Value("Authentication") = 5

AppCollection.SaveChanges

'*********
```

```
' ROLES
'*********

Set Roles = AppCollection.GetCollection("Roles", App.Key)

Set Manager = Roles.Add
Set Teller  = Roles.Add
Set Customer = Roles.Add

Manager.Value("Name")  = "Manager"
Teller.value("Name")   = "Teller"
Customer.value("Name") = "Customer"

' Save role changes
Roles.SaveChanges

' Set Manager role
Set Users = Roles.GetCollection("UsersInRole", Manager.Key)

Set User1 = Users.Add
User1.value("User") = "SOAP_VBANK_MGR_L"
Users.SaveChanges

' Set Customer role
Set Users = Roles.GetCollection("UsersInRole", Customer.Key)

Set User1 = Users.Add
User1.value("User") = "SOAP_VBANK_CUST_L"
Users.SaveChanges

' Set Teller role
Set Users = Roles.GetCollection("UsersInRole", Teller.Key)

Set User1 = Users.Add
User1.value("User") = "SOAP_VBANK_TLR_L"
Users.SaveChanges

' Save role changes
Roles.SaveChanges

'***************************************
'
' Set individual component roles
'
Set CompCollection = AppCollection.GetCollection("Components", App.Key)

' Populate the components
CompCollection.Populate

' Add roles for components
For Each Comp in CompCollection
    If Comp.Name = "VBankBackend.authorizeLoan" Then
       ' Enable component checks
       Comp.Value("ComponentAccessChecksEnabled") = True

       ' Save updates
       CompCollection.SaveChanges
       ' Add the role(s) to component
       Set Roles = CompCollection.GetCollection("RolesForComponent",
Comp.Key)
       Set CompRole1 = Roles.Add
       CompRole1.Value("Name") = "Manager"
       Roles.SaveChanges
    ElseIf Comp.Name = "VBankBackend.Deposit" Then
       ' Enable component checks
       Comp.Value("ComponentAccessChecksEnabled") = True

       ' Save updates
       CompCollection.SaveChanges

       ' Add the role(s) to component
       Set Roles = CompCollection.GetCollection("RolesForComponent",
Comp.Key)
       Set CompRole2 = Roles.Add
       CompRole2.Value("Name") = "Teller"
       Roles.SaveChanges
    ElseIf Comp.Name = "VBankBackend.Withdrawal" Then
       ' Enable component checks
       Comp.Value("ComponentAccessChecksEnabled") = True

       ' Save updates
       CompCollection.SaveChanges

       ' Add the role(s) to component
       Set Roles = CompCollection.GetCollection("RolesForComponent",
Comp.Key)
       Set CompRole2 = Roles.Add
       CompRole2.Value("Name") = "Teller"
       Set CompRole3 = Roles.Add
       CompRole3.Value("Name") = "Customer"
       Roles.SaveChanges
    ElseIf Comp.Name = "VBankBackend.retrieveBalance" Then
       ' Enable component checks
       Comp.Value("ComponentAccessChecksEnabled") = True

       ' Save updates
       CompCollection.SaveChanges

       ' Add the role(s) to component
       Set Roles = CompCollection.GetCollection("RolesForComponent",
Comp.Key)
       Set CompRole1 = Roles.Add
       CompRole1.Value("Name") = "Manager"
       Set CompRole2 = Roles.Add
       CompRole2.Value("Name") = "Teller"
       Set CompRole3 = Roles.Add
       CompRole3.Value("Name") = "Customer"
       Roles.SaveChanges
    End If
Next

' Update Component collection
CompCollection.SaveChanges
```

```
' Commit changes
AppCollection.SaveChanges

Function Environ(var)
   Dim shell
   Set shell = wscript.CreateObject("Wscript.Shell")
   Environ = shell.ExpandEnvironmentStrings("%" + var + "%")
   Set shell = Nothing
End Function
```

out.log file

```
C:\Documents and Settings\Administrator\Desktop\scripts>REM *** This
file serves as the single click driver for driver.vbs             ***

C:\Documents and Settings\Administrator\Desktop\scripts>REM *** Users
need only run this file to execute all scripts in correct order  ***

C:\Documents and Settings\Administrator\Desktop\scripts>REM *** Created
by: Derek Lawless 7/20/04                                      ***

C:\Documents and Settings\Administrator\Desktop\scripts>REM ***
***

C:\Documents and Settings\Administrator\Desktop\scripts>cscript
driver.vbs
Microsoft (R) Windows Script Host Version 5.6
Copyright (C) Microsoft Corporation 1996-2001. All rights reserved.

Running ID creation script
*******************************

C:\Documents and Settings\Administrator\Desktop\scripts>REM *** This
file creates all local IDs for VBANK    ***

C:\Documents and Settings\Administrator\Desktop\scripts>REM *** Created
by: Derek Lawless 7/20/04            ***

C:\Documents and Settings\Administrator\Desktop\scripts>REM ***
***

C:\Documents and Settings\Administrator\Desktop\scripts>echo ***
Creating local IDs ***
*** Creating local IDs ***

C:\Documents and Settings\Administrator\Desktop\scripts>if MIDTIER ==
INTER NET USER COM_VBANK "8yEf7d(3dsx09l" /ADD

C:\Documents and Settings\Administrator\Desktop\scripts>if MIDTIER ==
INTRA NET USER COM_VBANK "8yEf7d(3dsx09l" /ADD

C:\Documents and Settings\Administrator\Desktop\scripts>if MIDTIER ==
MIDTIER NET USER COM_VBANK_BACK "0kr!45xqzuUvt8" /ADD
The command completed successfully.



C:\Documents and Settings\Administrator\Desktop\scripts>if MIDTIER ==
MIDTIER NET USER SOAP_VBANK_CUST o*hkS#xtrP91!c /ADD
The command completed successfully.



C:\Documents and Settings\Administrator\Desktop\scripts>if MIDTIER ==
MIDTIER NET USER SOAP_VBANK_TLR xpCATr#tpq!(zs /ADD
The command completed successfully.



C:\Documents and Settings\Administrator\Desktop\scripts>if MIDTIER ==
MIDTIER NET USER SOAP_VBANK_MGR *ck#t*wmnGiO73 /ADD
The command completed successfully.



C:\Documents and Settings\Administrator\Desktop\scripts>if MIDTIER ==
MIDTIER NET LOCALGROUP SOAP_VBANK_L /COMMENT:"VBANK SOAP group" /ADD

C:\Documents and Settings\Administrator\Desktop\scripts>if MIDTIER ==
MIDTIER NET LOCALGROUP SOAP_VBANK_CUST_L /COMMENT:"VBANK SOAP group"
/ADD

C:\Documents and Settings\Administrator\Desktop\scripts>if MIDTIER ==
MIDTIER NET LOCALGROUP SOAP_VBANK_TLR_L /COMMENT:"VBANK SOAP group"
/ADD

C:\Documents and Settings\Administrator\Desktop\scripts>if MIDTIER ==
MIDTIER NET LOCALGROUP SOAP_VBANK_MGR_L /COMMENT:"VBANK SOAP group"
/ADD

C:\Documents and Settings\Administrator\Desktop\scripts>if MIDTIER ==
MIDTIER NET LOCALGROUP SOAP_VBANK_CUST_L SOAP_VBANK_CUST /ADD
The command completed successfully.



C:\Documents and Settings\Administrator\Desktop\scripts>if MIDTIER ==
MIDTIER NET LOCALGROUP SOAP_VBANK_TLR_L SOAP_VBANK_TLR /ADD
The command completed successfully.



C:\Documents and Settings\Administrator\Desktop\scripts>if MIDTIER ==
MIDTIER NET LOCALGROUP SOAP_VBANK_MGR_L SOAP_VBANK_MGR /ADD
The command completed successfully.



C:\Documents and Settings\Administrator\Desktop\scripts>if MIDTIER ==
MIDTIER NET LOCALGROUP SOAP_VBANK_L SOAP_VBANK_CUST /ADD
The command completed successfully.



C:\Documents and Settings\Administrator\Desktop\scripts>if MIDTIER ==
MIDTIER NET LOCALGROUP SOAP_VBANK_L SOAP_VBANK_TLR /ADD
The command completed successfully.



C:\Documents and Settings\Administrator\Desktop\scripts>if MIDTIER ==
MIDTIER NET LOCALGROUP SOAP_VBANK_L SOAP_VBANK_MGR /ADD
```

```
The command completed successfully.



C:\Documents and Settings\Administrator\Desktop\scripts>echo *** Done.
***
*** Done. ***

*******************************
Running File install and security script
*******************************

C:\Documents and Settings\Administrator\Desktop\scripts>REM *** This
file installs files on the Mid-tier     ***

C:\Documents and Settings\Administrator\Desktop\scripts>REM *** server
and secures the files                 ***

C:\Documents and Settings\Administrator\Desktop\scripts>REM *** Created
by: Derek Lawless 7/20/04            ***

C:\Documents and Settings\Administrator\Desktop\scripts>REM ***
***

C:\Documents and Settings\Administrator\Desktop\scripts>echo "Creating
directory structure for MIDTIER server..."
"Creating directory structure for MIDTIER server..."

C:\Documents and Settings\Administrator\Desktop\scripts>md C:\APPS

C:\Documents and Settings\Administrator\Desktop\scripts>md
C:\APPS\VBANK

C:\Documents and Settings\Administrator\Desktop\scripts>md
C:\APPS\VBANK\DLL

C:\Documents and Settings\Administrator\Desktop\scripts>md
C:\APPS\VBANK\website

C:\Documents and Settings\Administrator\Desktop\scripts>md C:\LOGS

C:\Documents and Settings\Administrator\Desktop\scripts>md C:\LOGS\Web

C:\Documents and Settings\Administrator\Desktop\scripts>md
C:\APPS\VBANK\website\WSDL

C:\Documents and Settings\Administrator\Desktop\scripts>echo "Done."
"Done."

C:\Documents and Settings\Administrator\Desktop\scripts>echo ""
""

C:\Documents and Settings\Administrator\Desktop\scripts>echo
"Installing files on MIDTIER server..."
"Installing files on MIDTIER server..."
```

```
C:\Documents and Settings\Administrator\Desktop\scripts>rem *** Install
Mid-Tier files ***

C:\Documents and Settings\Administrator\Desktop\scripts>COPY
source\VBankBackend.dll C:\APPS\VBANK\DLL
        1 file(s) copied.

C:\Documents and Settings\Administrator\Desktop\scripts>COPY
source\authorizeLoan.ws* C:\APPS\VBANK\website\WSDL
source\authorizeLoan.wsdl
source\authorizeLoan.wsml
        2 file(s) copied.

C:\Documents and Settings\Administrator\Desktop\scripts>COPY
source\Deposit.ws* C:\APPS\VBANK\website\WSDL
source\Deposit.wsdl
source\Deposit.wsml
        2 file(s) copied.

C:\Documents and Settings\Administrator\Desktop\scripts>COPY
source\Withdrawal.ws* C:\APPS\VBANK\website\WSDL
source\Withdrawal.wsdl
source\Withdrawal.wsml
        2 file(s) copied.

C:\Documents and Settings\Administrator\Desktop\scripts>COPY
source\retrieveBalance.ws* C:\APPS\VBANK\website\WSDL
source\retrieveBalance.wsdl
source\retrieveBalance.wsml
        2 file(s) copied.

C:\Documents and Settings\Administrator\Desktop\scripts>echo "Done."
"Done."

C:\Documents and Settings\Administrator\Desktop\scripts>echo ""
""

C:\Documents and Settings\Administrator\Desktop\scripts>echo "Setting
file security..."
"Setting file security..."

C:\Documents and Settings\Administrator\Desktop\scripts>ECHO Y  | CACLS
C:\APPS\VBANK\DLL /C /T /P SOAP_VBANK_L:R Administrators:F SYSTEM:F
Are you sure (Y/N)?processed dir: C:\APPS\VBANK\DLL
processed file: C:\APPS\VBANK\DLL\VBankBackend.dll

C:\Documents and Settings\Administrator\Desktop\scripts>ECHO Y  | CACLS
C:\APPS\VBANK\website\WSDL /C /T /P SOAP_VBANK_L:R Administrators:F
SYSTEM:F
Are you sure (Y/N)?processed dir: C:\APPS\VBANK\website\WSDL
processed file: C:\APPS\VBANK\website\WSDL\authorizeLoan.wsdl
processed file: C:\APPS\VBANK\website\WSDL\authorizeLoan.wsml
processed file: C:\APPS\VBANK\website\WSDL\Deposit.wsdl
processed file: C:\APPS\VBANK\website\WSDL\Deposit.wsml
processed file: C:\APPS\VBANK\website\WSDL\retrieveBalance.wsdl
processed file: C:\APPS\VBANK\website\WSDL\retrieveBalance.wsml
processed file: C:\APPS\VBANK\website\WSDL\Withdrawal.wsdl
processed file: C:\APPS\VBANK\website\WSDL\Withdrawal.wsml

C:\Documents and Settings\Administrator\Desktop\scripts>ECHO Y  | CACLS
C:\APPS\VBANK\website\WSDL\authorizeLoan.ws* /C /T /P
SOAP_VBANK_MGR_L:R Administators:F SYSTEM:F

C:\Documents and Settings\Administrator\Desktop\scripts>ECHO Y  | CACLS
C:\APPS\VBANK\website\WSDL\Deposit.ws* /C /T /P SOAP_VBANK_TLR_L:R
Administators:F SYSTEM:F

C:\Documents and Settings\Administrator\Desktop\scripts>ECHO Y  | CACLS
C:\APPS\VBANK\website\WSDL\Withdrawal.ws* /C /T /P SOAP_VBANK_TLR_L:R
SOAP_VBANK_CUST_L:R Administators:F SYSTEM:F

C:\Documents and Settings\Administrator\Desktop\scripts>ECHO Y  | CACLS
C:\APPS\VBANK\website\WSDL\retrieveBalance.ws* /C /T /P SOAP_VBANK_L:R
Administators:F SYSTEM:F

C:\Documents and Settings\Administrator\Desktop\scripts>echo "Done."
"Done."

*******************************
Running Website Config script
*******************************
Microsoft (R) Windows Script Host Version 5.6
Copyright (C) Microsoft Corporation 1996-2001. All rights reserved.


*******************************
Running Virtual Directory creation script
*******************************
Microsoft (R) Windows Script Host Version 5.6
Copyright (C) Microsoft Corporation 1996-2001. All rights reserved.


*******************************
Running COM+ application creation scripts
*******************************
Microsoft (R) Windows Script Host Version 5.6
Copyright (C) Microsoft Corporation 1996-2001. All rights reserved.


*******************************
Running COM+ application security scripts
*******************************
Microsoft (R) Windows Script Host Version 5.6
Copyright (C) Microsoft Corporation 1996-2001. All rights reserved.


*******************************
Driver script finished.
```

References

"A Brief History of the Internet." URL: http://www.isoc.org/internet/history/brief.shtml (22 July 2004).

"A Brief History of SOAP." URL: http://webservices.xml.com/pub/a/ws/2001/04/04/soap.html (22 July 2004).

"About Applications." URL: http://www.microsoft.com/windows2000/en/server/iis/default.asp?url=/windows2000/en/server/iis/htm/core/iiwarndc.htm (22 July 2004).

Bowne, Ben. "A Guide to Building Secure and Scalable Web Applications Using COM+." URL: http://www.giac.org/practical/GSEC/Ben_Bowne_GSEC.pdf (22 July 2004).

Brown, Keith. Programming Windows Security. Boston: Addison-Wesley, November 2000.

Clarke, Todd. "MTS/COM+ Security Design." URL: http://www.giac.org/practical/GSEC/Todd_Clarke_GSEC.pdf (22 July 2004).

Fallon, James R. "Script the Configuration of Your Default Web Site." URL: http://www.winnetmag.com/Web/Article/ArticleID/24840/24840.html (22 July 2004).

"IIS 5 Administration – Home Directory." URL: http://www.webhelpinghand.com/iis_administration_home.htm (22 July 2004).

"IIS Programmatic Administration Reference." URL: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/iissdk/iis/iis_programmatic_administration_reference.asp (22 July 2004).

"Internet Domain Survey, Jan 2004." URL: http://www.isc.org/ (22 July 2004).

"Introductory Example Using the COM+ Administration Catalog." URL: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cossdk/htm/pgautomatingadmin_2j51.asp (22 July 2004).

"Navigating the COM+ Collection Hierarchy." URL: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cossdk/htm/pgautomatingadmin_6e49.asp (22 July 2004).

Saltzer, Jerome H., Schroeder D., Michael. "The Protection of Information in Computer Systems." URL: http://cap-lore.com/CapTheory/ProtInf/ (22 July 2004).