|  |  |
| --- | --- |
| **Title:** | Auto Install and Configure FTPS (Windows 7) |
| Updated: | Jan 29th 2013 |
| Overview: | In this lab we will build an Auto-Install.bat script to automate installation and configuration of the following:   * IIS Management Tools & IIS FTP Server * Configuration of the Firewall for FTP SSL Access * Creation of Three User Accounts * Creation of a Site Instance and Binding in IIS * Generation of a Self-Signed SSL Certificate (Using SelfSSL7) * Assignment of SSL Certificate to FTP Site * NTFS Security Settings for User Accounts |
| Task/Activity: | 1. Use Deployment Image Servicing and Management (DISM) to Install FTP/IIS |
|  | 1. Use netsh to Configure the Firewall |
|  | 1. Add user accounts and set passwords |
|  | 1. Use appcmd to configure FTP site bindings, SSL policy, auth and permissions |
|  | 1. Generate self-signed certificate with SelfSSL7, use certutil and FOR loop to extract SSL hash, assign certificate using appcmd |
|  | 1. Assign NTFS security permissions with cacls |

1. **Use Deployment Image Servicing and Management (DISM) to Install FTP/IIS**

The Deployment Image Servicing and Management (DISM) tool performs all the functions of the short-lived pkgmgr, intlconfig, and peimg tools (which are in the process of being phased out by Microsoft). Its core function is to add, remove and query installed packages/drivers. You can also enable/disable windows features and prep Windows pre-installation images. For additional features I’ll defer you to TechNet for the full DISM functionality list.

1. Open a cmd shell on your Windows 7 system and enter the following DISM command to view the available /online options:

**C:\>dism /online /?**

1. Next run the following command to retrieve a list showing the status of the available features on your Windows system:

**C:\>dism /online /get-features**

It’s time to start building the Auto-Install script. In the first part of our FTPS install & configuration script we will launch the DISM tool using the START /WAIT command switch to ensure all packages finish installing before we continue on with the configuration of the services. Review the following command (which is all on one line) for the IIS and FTP feature(s) install:

*This command is all on line line*

**START /WAIT DISM /Online /Enable-Feature /FeatureName:IIS-Metabase /FeatureName:IIS-WMICompatibility /FeatureName:IIS-LegacyScripts /FeatureName:IIS-IIS6ManagementCompatibility /FeatureName:IIS-ManagementScriptingTools /FeatureName:IIS-WebServerManagementTools /FeatureName:IIS-WebServerRole /FeatureName:IIS-FTPExtensibility /FeatureName:IIS-FTPServer /FeatureName:IIS-FTPSvc /FeatureName:IIS-ManagementConsole /FeatureName:IIS-ManagementService**

1. **Use netsh to Configure the Firewall**

Network shell (netsh) is a command shell utility that gives us the ability to configure and display the status of network interface cards and a variety of other network-related server roles and components. The netsh also gives us full access to Windows network functionality (e.g. additional bridging, IPv6 options, RPC functions etc.) that cannot be configured from the MMC or other GUI interface.

1. Open a cmd shell on your Windows 7 system and enter the following netsh command to dump the current network configuration to a dat file. The dump command creates an executable netsh configuration script:

C:\>**netsh interface dump >%computername%-network-config.dat**

1. Open the **%computername%-network-config.dat** file you created using notepad and review the netsh configuration script:

**# ----------------------------------**

**# IPHTTPS Configuration**

**# ----------------------------------**

**pushd interface httpstunnel**

*pushd stores the name of the current directory(context)*

*popd changes to the directory(context) stored by pushd*

**reset**

**popd**

**# End of IPHTTPS configuration**

**# ----------------------------------**

**# IPv4 Configuration**

*static route*

**# ----------------------------------**

**pushd interface ipv4**

**reset**

**set global icmpredirects=enabled**

**add route prefix=10.10.200.0/24 interface="iftype0\_0" nexthop=10.10.201.2 metric=1 publish=Yes**

**add route prefix=0.0.0.0/0 interface="Local Area Connection" nexthop=192.168.10.1 publish=Yes**

**add address name="VMware Network Adapter VMnet1" address=192.168.80.1 mask=255.255.255.0**

**add address name="Local Area Connection" address=192.168.10.3 mask=255.255.255.0**

**popd**

**# End of IPv4 configuration**

1. It is not necessary to run the following command (our system are currently healthy), but if needed you can execute a netsh configuration script by using the netsh **exec** command as follows:

C:\>*netsh exec %computername%-network-config.dat*

To configure the firewall for the automated FTP Server installation we will use netsh to set and add new rule names and groups. To allow the FTP Service itself to accept incoming connections and allow inbound connections on TCP port 21, the two following netsh commands are executed:

C:\> **netsh advfirewall firewall set rule group="FTP Server" new enable="yes"**

C:\> **netsh advfirewall firewall add rule name="FTP SSL" action=allow protocol=TCP dir=in localport=21**

1. **Add user accounts and set passwords**

To add users in a domain environment you would typically use the **dsadd** tool available on 2K8\2K8R2\2K12 after installing the Active Directory server role. On stand-alone servers and workstations the conventional “net user” command provides an effective way to add and delete users in our scripts.

1. Open a cmd shell and type “net user /?” to view the syntax of the command:

C:\>**net user /?**

*Where are all the options?*

The syntax of this command is:

NET USER

[username [password | \*] [options]] [/DOMAIN]

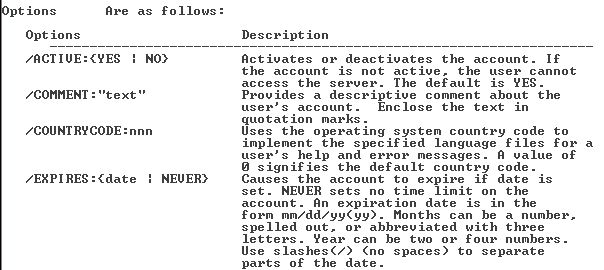
username {password | \*} /ADD [options] [/DOMAIN]

username [/DELETE] [/DOMAIN]

username [/TIMES:{times | ALL}]

1. The syntax provided by the “**/?**” in the net user command does not display any details of the options that are available for creating and editing user accounts. Try running the following command to retrieve a more detailed list of command options/parameters:

C:\>**net help user | more**



1. Our FTP Server installation scenario requires us to create three user accounts for bob, tom, and jay. Each FTP user must be prevented from uploading files between 10PM at night and 6AM in the morning. The FTP Server should also be inaccessible to the users on Sundays, a day reserved for backup activities and other administrative duties. The following three net user commands will create the accounts and set the appropriate logon times:

C:\> net user bob Student99 /add /fullname:"Bob Boberson" /comment:"Field Agent" /times:M-Sa,6AM-10PM

C:\> net user tom Student99 /add /fullname:"Tom Thompson" /comment:"Field Agent" /times:M-Sa,6AM-10PM

C:\> net user jay Student99 /add /fullname:"Jay Jaquish" /comment:"Field Agent" /times:M-Sa,6AM-10PM

1. **Use appcmd to configure FTP site bindings, SSL policy, auth and permissions**

The AppCmd.exe command is a general purpose cmd shell administration tool that allows you to configure services managed by Internet Information Service(s) (IIS). You can generate and delete sites, create apps within the sites (including application pools), and make virtual directories. It can also be used as a troubleshooting and administrative tool to stop/start sites, recycle application pools, list worker processes and view executing requests. Another key function of the AppCmd.exe tool is the ability to search, change, and export/import prior IIS & ASP.NET configurations; this includes creation/modification of policies.

1. The appcmd tool was installed earlier when we used DISM to install the FTP and IIS features; otherwise it would not be natively installed. There is no default path variable to map to the executable so we need to change into the **\windows\system32\inetsrv** directory before running the tool. Execute the following commands to view the **appcmd.exe** options:

C:\>**cd \windows\system32\inetsrv**

C:\Windows\System32\inetsrv>**appcmd /?**

<snip>

**APPCMD (command) (object-type) <identifier> </parameter1:value1 ...>**

<snip>

To setup and configure FTP to use SSL (FTPS) we will need to create the initial site, set the port binding, create the SSL policy, set the authentication type, and then configure authorization for the users (or groups) who are allowed access to the site. Review the following appcmd commands that perform these actions:

*Note: The following appcmd.exe commands are all executed with an absolute path of: c:\windows\system32\inetsrv\apcmd.exe. The following examples are truncated to just “appcmd.exe”.*

**appcmd.exe add site /name:"Drop" /bindings:ftp://\*:21 /physicalpath:c:\inetpub\ftproot**

Create a new site called “Drop” (short for dropbox) and bind to port 21 on all listening addresses. Set c:\inetpub\ftproot as the root directory of the FTP server.

**appcmd.exe set config -section:system.applicationHost/sites /[name='Drop'].ftpServer.security.ssl.controlChannelPolicy:"SslRequire"**

Set both the control channel and data channel policy to REQUIRE SSL. Assign this policy to the “Drop” site we created earlier.

**appcmd.exe set config -section:system.applicationHost/sites /[name='Drop'].ftpServer.security.ssl.dataChannelPolicy:"SslRequire"**

**appcmd.exe set config -section:system.applicationHost/sites /[name='Drop'].ftpServer.security.authentication.basicAuthentication.enabled:true**

Set the “Drop” site to use basic authentication (which will be protected via SSL) and disable anonymous access.

**appcmd.exe set config -section:system.applicationHost/sites /[name='Drop'].ftpServer.security.authentication.anonymousAuthentication.enabled:false**

**appcmd.exe set config "Drop" -section:system.ftpServer/security/authorization /+"[accessType='Allow',users='bob',permissions='Read, Write']" /commit:apphost**

**appcmd.exe set config "Drop" -section:system.ftpServer/security/authorization /+"[accessType='Allow',users='tom',permissions='Read, Write']" /commit:apphost**

**appcmd.exe set config "Drop" -section:system.ftpServer/security/authorization /+"[accessType='Allow',users='jay',permissions='Read, Write']" /commit:apphost**

The last three commands exclusively set Read/Write permissions for

bob, tom, and jay to RW via the “Drop” FTP site. Note: These authorization settings are independent of any file system (e.g. NTFS) settings that may be applied to the root physical directory of the FTP site.

1. **Generate self-signed certificate with SelfSSL7, use certutil and FOR loop to extract SSL hash, assign certificate using appcmd**

SelfSSL is a tool from Microsoft that allows you to create self-signed SSL certificates for any SSL/TLS enabled IIS service. Self signed certificates are a low-cost and effective way of managing your own mini-“Public-Key-Infrastructure”, and also a good way to prototype and build SSL enabled systems prior to investing in a certificate from a trusted authority. Although the SelfSSL utility has the ability to generate and assign (add) an SSL binding to IIS (using the /S, /A, and /P parameters), we will use the certutil accompanied with a FOR loop to extract the hash and then assign the SSL certificate binding to IIS using appcmd.

The SelfSSL7 utility is expecting a Yes or No interactive response when executed. You can often pre-answer a Y/N prompt by piping the response character to the command when you run it. Review the SelfSSL7 command executed in the install script:

Echo the letter “Y” to the SelfSSL7 command. The “Y” will not be interpreted by the command until AFTER the SelfSSL7.exe executes.

**echo Y | %installdrive%\SelfSSL7 /N:CN=%computername% /K 2048 /V 3600 /T**

The “CN” is the common name. If necessary, you could set it to a FQDN.

/K **size** *Specifies the key length. Default is 1024*

/V **days** *Specifies the validity of the certificate in days. (default is 30 days)*

/T Adds the self-signed certificate to the user's "Trusted Certificates" list.

Certutil is a command-line program from Microsoft that allows you to manage and configure a variety of certificate and certificate store items using a variety of verbs. The SelfSSL7 utility generates a certificate and stores it locally. To assign the self-signed certificate to an IIS site using the appcmd tool we need the exact (in this case) SHA1 hash of the certificate. The certutil command displays the certificate in hex format with spaces in-between the hash values. To remove the spaces and use the native SHA1 value we will use a for loop.

1. Open a cmd shell and enter the following command to view the certutil command line options:

C:\>**certutil –v -?**

As you can tell by the output of the previous command, the certutil offers a wide variety of certificate management options. Because the cmd shell was not designed to be a full functioning shell environment, instead more of a tool shell with some BASIC ability, you may sometimes find it is more effective to store variables & stdout results in a text file vs. trying to extract and assign variables. Because the SHA1 hash needs to be extracted and then set to a variable, we will store the output of the certutil command in a text file. We can then read the text file into our FOR loop and extract/set the SSL hash using tokens and delims.

Review the following certutil command and FOR loop:

Use findstr to search for only the lines that have sha1 hashes in them.

The “my” store extracts the SHA1 hash from the “My user account”.

**certutil -store my | findstr /C:"Cert Hash(sha1):" >%tmp%\mycerthash-tmp.txt**

Store the results in a text file located in the systems temp directory.

Read in the text file results from the certutil command and parse out 22 tokens using “space” as a delimiting character. Set the mysslhash variable using tokens 3-22.

**for /f "tokens=1-22 delims= " %%a in (%tmp%\mycerthash-tmp.txt) do (**

**set mysslhash=%%c%%d%%e%%f%%g%%h%%i%%j%%k%%l%%m%%n%%o%%p%%q%%r%%s%%t%%u%%v**

**)**

Once we have the SHA1 hash of the SSL certificate parsed out and set as a variable we can assign the certificate using the appcmd command.

**c:\windows\system32\inetsrv\appcmd.exe set config -section:system.applicationHost/sites /[name='Drop'].ftpServer.security.ssl.serverCertHash:"%mysslhash%" /commit:apphost**

Pass the new %mysslhash% variable to the appcmd to assign the certificate to the “Drop” site.

1. **Assign NTFS security permissions with cacls**

Icacls is a newer command line tool from Microsoft that allows us to display, backup/import, and modify security descriptors (access control lists) on files and directories. Although this shiny new tool is quite capable, so is the legacy cacls.exe command. Cacls allows us to add, delete, and change ACLs on files and folders (including symbolic links and directory mounted volumes).

1. Open a cmd shell and enter the following command to view the options of the cacls command:

C:\>**cacls /?**

The following two cacls commands will edit the existing acl on the FTP root directory—granting the “Users” group WRITE and READ access to the directory. The bob, tom, and jay users are members of the local “Users” group.

**cacls c:\inetpub\ftproot /G Users:W /T /E**

**cacls c:\inetpub\ftproot /G Users:R /T /E**