Using OpenSSH in a Single Sign-On Corporate Environment with z/OS, Windows and Linux

Dovetailed Technologies
February 2016

Edition 2.0.0

For the latest version of this document, see
http://dovetail.com/docs/ssh/kerberos_sso.pdf

Contents

1 Overview

One of the vexing issues associated with large SSH networks is the management of user and host keys. Many corporate environments have strict policies regarding key expiration and revocation, and adapting standard SSH key management practices to conform to these policies can be difficult. IBM's z/OS OpenSSH implementation addresses some of these concerns by enabling the storage of OpenSSH keys in RACF Digital Certificates, but this capability does not extend beyond the z/OS environment, making it an incomplete solution.

Kerberos is an integral part of Windows Active Directory and is also available on z/OS and Linux systems. With the addition of Kerberos support (via the GSSAPI options) in IBM's OpenSSH implementation, single sign-on for SSH is achievable by extending an existing Windows Active Directory Domain to include Linux servers. z/OS servers can either be included in the Windows realm directly or indirectly by establishing a trust relationship between the Windows realm and a z/OS Kerberos realm. In addition to the benefit of providing users the convenience of single sign-on, a Kerberos based implementation eliminates SSH user and host private-public key pairs for all z/OS, Windows and Linux OpenSSH connections within that environment. Using Kerberos in a corporate environment simplifies the administration of SSH and reduces the need for a separate SSH key management solution.

Why define a separate realm for z/OS? While it is possible to add z/OS LPARs to an existing Windows realm (and in some cases, this may be desirable due to its simplicity), a local z/OS realm allows z/OS SAF users to be transparently authenticated to Kerberos without requiring a password (see the z/OS kinit command with the –s option). This is especially beneficial in a batch environment where interactive password entry is not feasible. Both methods are presented in this paper: Eliminating SSH keys with Kerberos (Single Realm) and Eliminating SSH keys with Kerberos (Cross Realm).

This paper assumes an existing Windows Active Directory presence (common in most corporations) and details the steps necessary to incorporate Kerberos enabled SSH in this environment. Before discussing the configuration changes required to support Kerberos, it will be useful to first review how SSH connections are established.

1.1 SSH Connection Review

To make an SSH connection, **two** authentications are required:

1. The client must identify and accept the server (server authentication)
2. The server must identify and accept the client (client authentication).

Server authentication is generally performed by the server presenting the client the public key portion of its "host" key pair. The client will check to see if this public key is in its `known_hosts` file. If a match is found, the server is accepted. If no match for the server's public key is found, SSH asks the user to verify the key based on its fingerprint. If the user accepts the key, it is added to the `known_hosts` file for future use. Proper host key verification can be tedious, and as a result, many users blindly accept host keys when presented. This is a potential security risk. In addition, if corporate policy requires that

host keys have an expiry, the process of removing the old keys and replacing them with the new versions can be difficult to manage efficiently.

Client authentication can take several forms: password, key, smart card, etc. Standard practice, in many corporate SSH environments, is to issue each user a public/private key pair for authentication. When a user establishes a connection to a server, the user's public key is presented to the server. The server then checks to see if this public key is valid by checking the user's `authorized_keys` file on the server. If a match is found, the user is authorized. If not, the user is denied. Proper management of user keys can be even more difficult than host keys: Individual users are required to maintain secure access to their private key data, and key expiry and replacement can be especially cumbersome in environments with large numbers of users.

IBM's OpenSSH implementation for z/OS allows host keys and user keys to be stored in RACF digital certificates, so this allows for more formal expiry policy management, but similar facilities are not generally available on Linux and/or Windows.

Note: For more information, see the following webinars (dovetail.com/webinars.html):
IBM Ported Tools for z/OS: OpenSSH - Using Key Rings
IBM Ported Tools for z/OS: OpenSSH - Key Authentication

Fortunately, it is possible to use an existing Windows AD/DC environment and its underlying Kerberos implementation to completely eliminate the requirement for SSH key pairs. In doing so, SSH user and host management can be handled using the existing AD environment which has well established policies and implementations for expiry and key security.

2 Sample Environment

To illustrate integrating SSH authentication and key exchange into an existing Windows AD Kerberos installation, we will start with the environment shown in Figure 1. The sample Kerberos environment contains the following servers and workstations configured with the listed software:

- Windows Server 2012 R2 Active Directory Domain Controller and DNS
- Windows Server with a SSH Server supporting GSSAPI Authentication and Key Exchange. Tested with: VanDyke Software VShell Server 4.1
- z/OS server v2r2 or higher with Kerberos enabled OpenSSH and z/OS Network Authentication Service
- Linux server with OpenSSH 5.6 or later and the Kerberos 5 client installed. To verify whether OpenSSH on your server supports the correct GSSAPI options, check your distro's `sshd_config` man page to confirm that the GSSAPIAuthentication and GSSAPIKeyExchange options are available.
- Windows client workstation with a SSH Client supporting GSSAPI Authentication and Key Exchange. Clients tested:
    - VanDyke Software SecureFX 7.3.4
    - PuTTY 0.64 -The standard version of PuTTY 0.64 supports GSSAPI Authentication but not Key Exchange. If PuTTY 0.64 is used as the SSH client, not all host keys can be eliminated from the sample environment. An updated open source version of PuTTY with Key Exchange support is available from https://marcussundberg.com/putty/.

Note: Microsoft has announced plans to provide ported versions of the OpenSSH server and client to its PowerShell product. See http://blogs.msdn.com/b/powershell/archive/2015/10/19/openssh-for-windows-update.aspx Early versions of this code can be downloaded from GitHub: https://github.com/PowerShell/Win32-OpenSSH

Figure 1 illustrates the starting point for the sample environment which uses SSH user private-public key pairs and host keys.

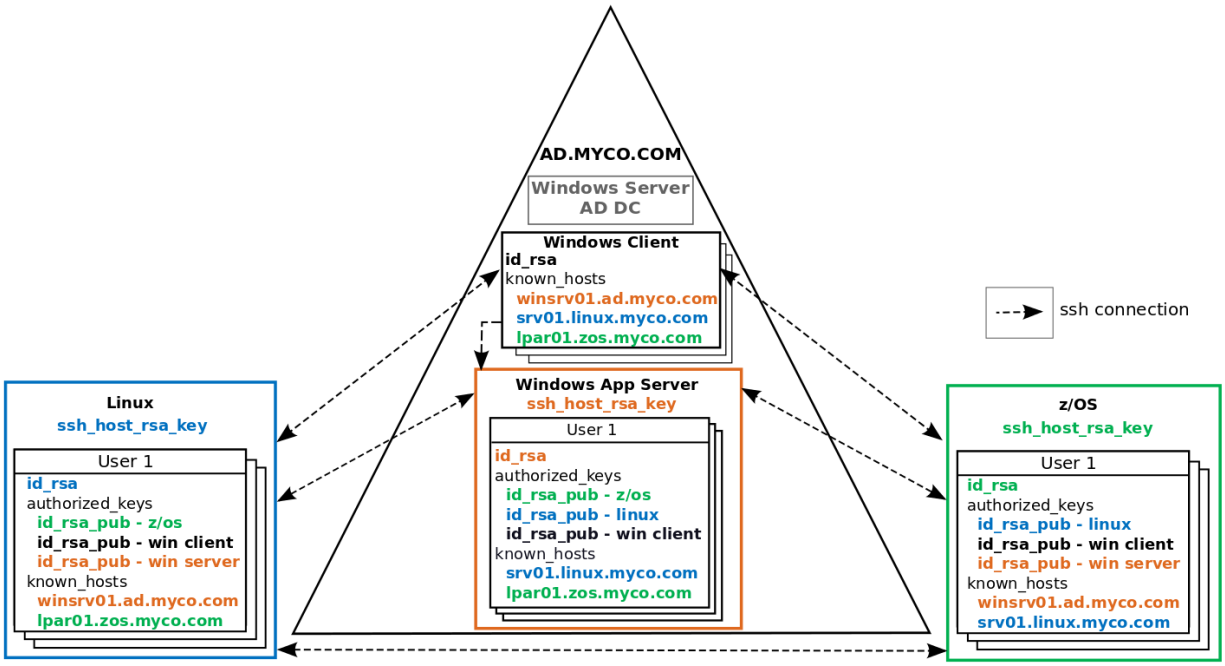


Figure 1 Infrastructure using SSH user private-public key pairs and host keys

In Figure 1, the triangle represents the Windows domain `AD.MYCO.COM`. The users in this environment have full SSH connectivity between the Windows application server, Linux server and z/OS server. SSH connections are established using private-public key pairs. Host keys are shown in the `known_host` file for each user. For simplicity, global known hosts files are not shown. Each Windows, Linux and z/OS user has a private key in their local `.ssh` directory. Each user's public key also exists in the user's `authorized_keys` file for each host the user connects to. Should a user's private key need to be expired due to a security vulnerability or company security policy, the diagram illustrates the cumbersome changes required due to the distributed nature of SSH key pairs.

2.1 Verify DNS Setup

Before starting with Kerberos configuration, make sure that the DNS/Resolver service is working properly and a fully qualified domain name is configured. In our sample environment, the z/OS command `hostname -r` returns `lpar01.zos.myco.com`. On Linux, `hostname -f` returns `srv01.linux.myco.com`. For a Windows 2012 Server, right click on Start and select System. The full computer name for the AD DC and the application server are `server.ad.myco.com` and `winsrv01.ad.myco.com`, respectively.

Kerberos clients depend on DNS lookups for building service principal names. Configuring Kerberos will be significantly easier if forward and reverse lookups for the servers are correct. Verifying the service principal name in Kerberos requests using either Kerberos tracing or a network packet analyzer can quickly identify the source of a problem.

3 Eliminating SSH keys with Kerberos (Single Realm)

Figure 2 shows the sample environment with Kerberos configured and the user and host keys removed by incorporating the Linux, z/OS and Windows servers in the same realm, AD.MYCO.COM.

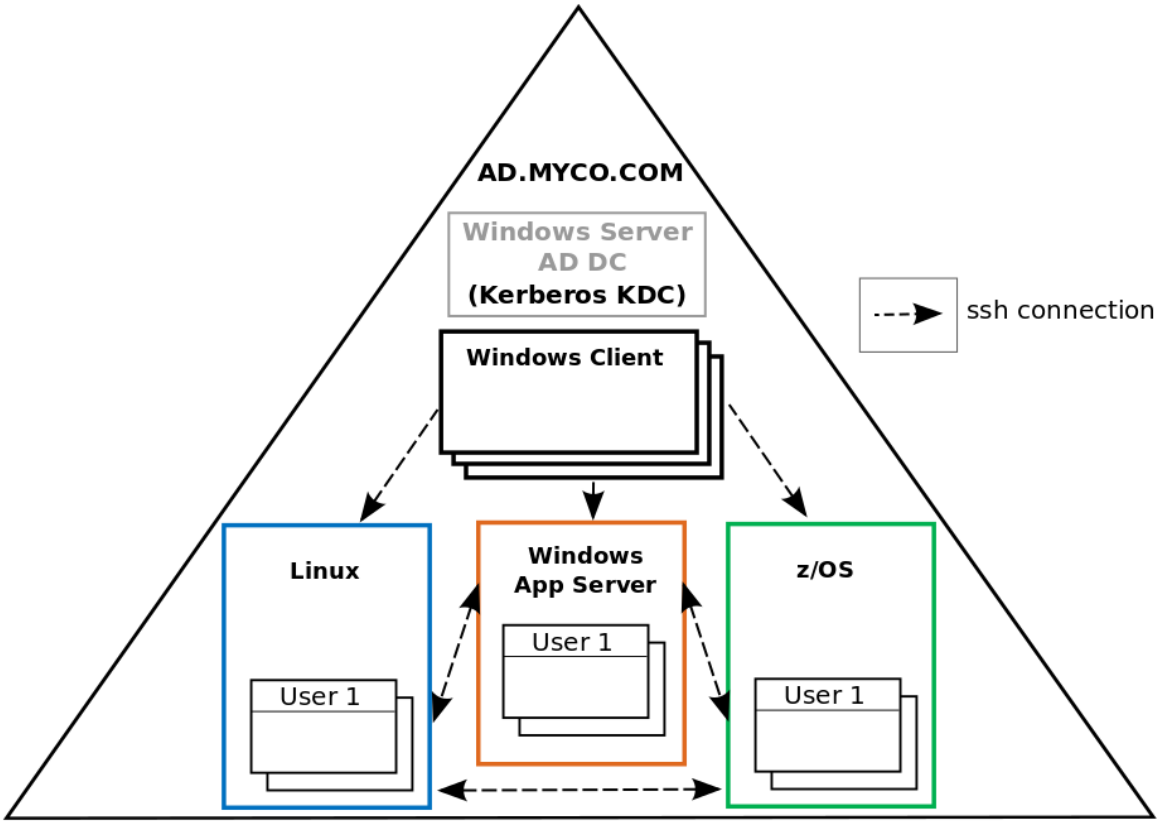


Figure 2 User and Host Keys Eliminated with Kerberso (single realm)

The Windows Server AD DC defines the windows Kerberos realm (`AD.MYCO.COM`). The Windows Key Distribution Center (KDC) is available as part of the Microsoft Security Support Provider Interface (SSPI).

The Linux server has been updated by installing the Kerberos client and configuring its default realm to be `AD.MYCO.COM`.

The z/OS server has been updated by configuring its default Kerberos realm to be `AD.MYCO.COM`.

Each SSH server (SSHD) and client configuration has been updated to use GSSAPIAuthentication and GSSAPIKeyExchange. The user's key pairs have been removed from user's control and are now managed by the Kerberos protocol in conjunction with the Windows KDC. Additionally, host keys have been removed by deleting `known_host` files.

The following sections describe in detail the software and configuration changes required. For simplicity, this example assumes that the username `aeneas` exists in the Windows domain and on the Linux server. The z/OS username is `hercules`.

The high level tasks for configuring this single realm Kerberos OpenSSH environment are:

- Configure the OpenSSH Servers on z/OS, Linux and Windows, enabling GSSAPI Authentication and Key Exchange
- Define Service Principals for the OpenSSH servers (SSHD) on z/OS, Linux and Windows
- Configure and test OpenSSH Client connections on z/OS, Linux and Windows

3.1 Kerberos Single Realm Configuration

This section configures the Kerberos client on the z/OS and Linux servers. No additional configuration is required for Windows servers.

3.1.1 z/OS Server

On z/OS, Kerberos is part of the Network Authentication Service component of Integrated Security Services.

3.1.1.1 Define the krb5.conf

Create `/etc/skrb/krb5.conf` setting the default realm to `AD.MYCO.COM`. Note the following:

- In the `libdefaults` section of `/etc/skrb/krb5.conf`, set `kdc_use_tcp = 1`. This option configures Kerberos to use TCP rather than UDP when communicating with the KDC. Without this option, the following error may occur when getting a ticket granting ticket from the windows KDC.

```
EUVF06014E Unable to obtain initial credentials.
     Status 0x96c73a34 - Response too large for datagram.
```

- Configure the encryption types in `/etc/skrb/krb5.conf` to be compatible with Windows Servers. For Windows Server 2012, the following is recommended.

```
[libdefaults]
    default_realm = AD.MYCO.COM
    kdc_use_tcp = 1
    default_tkt_enctypes = aes256-cts-hmac-sha1-96,des-cbc-md5,des-cbc-crc
    default_tgs_enctypes = aes256-cts-hmac-sha1-96,des-cbc-md5,des-cbc-crc

[realms]
   AD.MYCO.COM = {
        kdc = server.ad.myco.com:88
        admin_server = server.ad.myco.com:749
   }

[domain_realm]
.ad.myco.com      = AD.MYCO.COM
ad.myco.com       = AD.MYCO.COM
.linux.myco.com   = AD.MYCO.COM
linux.myco.com    = AD.MYCO.COM
.zos.myco.com     = AD.MYCO.COM
zos.myco.com      = AD.MYCO.COM
```

3.1.1.2 Map Windows and z/OS Userids

For each Windows domain user that will SSH to a z/OS SSH server, define a mapping between the Windows user name and the z/OS user name. In the following example, `aeneas` is the windows domain user name and `hercules` is the z/OS user name.

```
RDEFINE KERBLINK /.../AD.MYCO.COM/aeneas APPLDATA('hercules')
```

3.1.2 Linux Server

Next, the Linux server is added to the Windows realm. The Linux server will be able to access the z/OS realm due to the trust relationship between the two realms.

3.1.2.1 Install the Kerberos Client

Depending on the Linux distribution of the server, use one of the following to install the Kerberos client.

```
yum install krb5-workstation
```

or

```
apt-get install krb5-user
```

3.1.2.2 Configure krb5.conf

After installing the Kerberos client on the Linux server, create `/etc/krb5`. Note that the encryption types in `/etc/krb5.conf` must be compatible with Windows Servers. For Windows Server 2012, the following is recommended.

```
[libdefaults]
   default_realm = AD.MYCO.COM
   default_tkt_enctypes=aes256-cts-hmac-sha1-96,des-cbc-md5,des-cbc-crc
   default_tgs_enctypes=aes256-cts-hmac-sha1-96,des-cbc-md5,des-cbc-crc
   permitted_enctypes=aes256-cts-hmac-sha1-96,des-cbc-md5,des-cbc-crc


[realms]
   AD.MYCO.COM = {
      kdc = server.ad.myco.com:88
      admin_server = server.ad.myco.com:749
      default_domain = ad.myco.com
   }

[domain_realm]
.ad.myco.com      = AD.MYCO.COM
ad.myco.com       = AD.MYCO.COM
.linux.myco.com   = AD.MYCO.COM
linux.myco.com    = AD.MYCO.COM
.zos.myco.com     = AD.MYCO.COM
zos.myco.com      = AD.MYCO.COM
```

3.1.3 Windows Server

Nothing specific is required because Kerberos is an integral part of Windows authentication.

3.2 OpenSSH GSSAPI Authentication and Key Exchange Configuration

On each server, modify the OpenSSH client and server configuration to use GSSAPI.

3.2.1 z/OS OpenSSH Server

For z/OS, update the `/etc/ssh/sshd_config` file setting the following.

```
GSSAPIAuthentication yes
GSSAPIKeyExchange yes
GSSAPICleanupCredentials yes
```

After making these configuration changes, restart SSHD.

3.2.2 z/OS OpenSSH Client

For z/OS, update the /etc/ssh/ssh_config file setting the following:

```
GSSAPIAuthentication yes
GSSAPIKeyExchange yes
GSSAPIDelegateCredentials yes
```

If global configuration is not desired, this option can either be added to the user's `.ssh/ssh_config` file or on the ssh command line. For example:

```
ssh -oGSSAPIAuthentication=yes -oGSSAPIKeyExchange=yes aeneas@srv01.linux.myco.com
```

3.2.3 Linux OpenSSH Server

For Linux, the `/etc/ssh/sshd_config` file settings are the same as the z/OS settings above.

3.2.4 Linux OpenSSH Client

For Linux, the `/etc/ssh/ssh_config` file settings are the same as the z/OS settings above.

3.2.5 Windows SSH Server

On the Windows Server, the SSH Server must support GSSAPI Authentication and Key Exchange. The following product was tested:

- VanDyke Software VShell 4.1 server enables GSSAPI Authentication by default. Kerberos must be enabled for Key Exchange by selecting the Kerberos algorithms on the SSH2->Key Exchange configuration page.

Note: Microsoft has announced plans to provide ported version of the OpenSSH server and client to its PowerShell product. See http://blogs.msdn.com/b/powershell/archive/2015/10/19/openssh-for-windows-update.aspx Early versions of this code can be downloaded from GitHub: https://github.com/PowerShell/Win32-OpenSSH

3.2.6 Windows SSH Client

For each windows domain user, use the Active Directory Administrative Center to **enable AES encryption options (this is required)**. This setting is under Encryption Options. Additionally, Windows clients must have SSH Client software installed. The following clients were tested:

- The standard distribution of the PuTTY 0.64 client supports GSSAPI Authentication, but not Key Exchange. An updated open source version of PuTTY with GSSAPI Key Exchange support is available from https://marcussundberg.com/putty/.
- The VanDyke Software SecureFX 7.3.4 client supports GSSAPI Authentication and Key Exchange.

3.3 Service Principal Configuration

A service principal name (SPN) uniquely defines an instance of a service within a network. Each SSH Server running on z/OS, Linux and Windows is assigned an SPN. The SPN name has the following format:

```
host/fully.qualified.hostname@REALM
```

Each server has a different way to configure a service principal name.

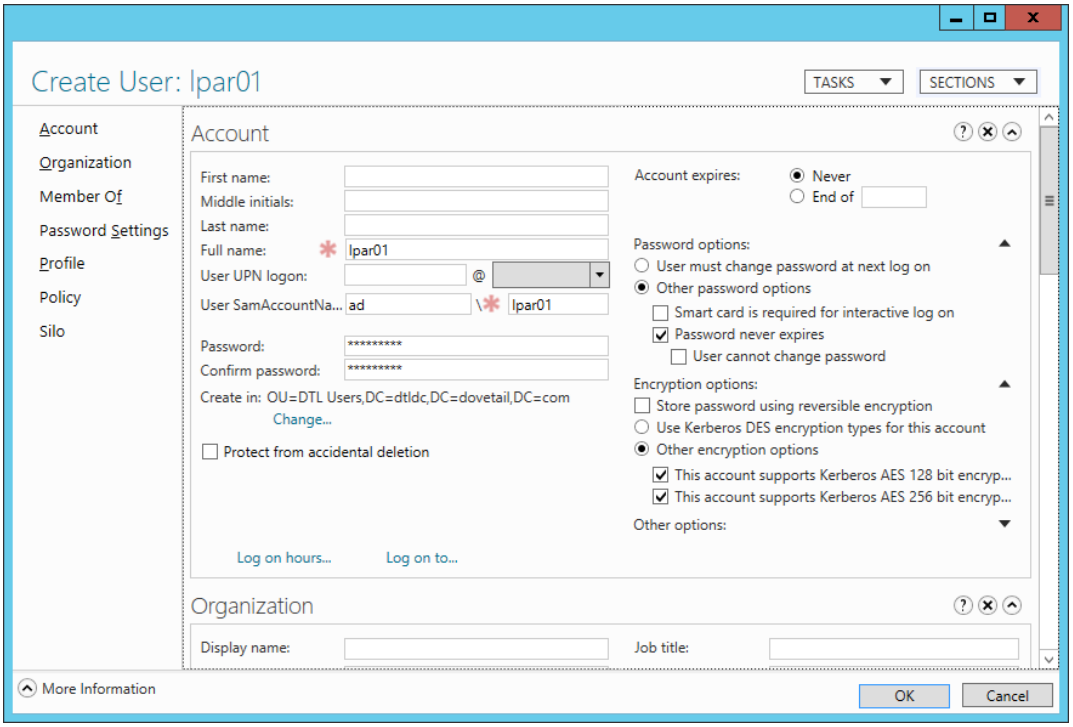
3.3.1 z/OS OpenSSH Server Service Principal

In this scenario, the z/OS server is being added to the Windows realm, so creating a service principal requires steps to be executed on the Windows Server as well as on z/OS.

3.3.1.1 Create an Active Directory User for the z/OS Service Principal

On the Windows Server, create an Active Directory user for the host (for example: `lpar01`. Create the user using the Active Directory Administrative Center user interface (Server Manager ->tools-> Active Directory Administrative Center). The screenshot below shows the minimum information required:

- Full name
- User SamAccountName
- Password and confirmed password
- Password never expires
- Encryption options: select both AES options *(Note: this setting is required for Kerberos to work with z/OS.)*



3.3.1.2 Configure the Service Principal Name and Generate a keytab

After the user has been created, execute the `ktpass` command to map the service principal to the user and to generate a keytab file.

Execute the following on the Windows domain controller from the Windows PowerShell.

```
ktpass princ host/lpar01.zos.myco.com@AD.MYCO.COM mapuser ad\lpar01
   -crypto all -pass password -ptype KRB5_NT_PRINCIPAL out lpar01.keytab
```

The command output will look like the following. Note specifically, the message in bold below.

```
PS C:\Users\Administrator> ktpass princ host/lapr01.zos.myco.com@AD.MYCO.COM mapuser
ad\lpar01 -crypto all -pass xxxxxxx -ptype KRB5_NT_PRINCIPAL out lpar01.keytab
Targeting domain controller: server.ad.myco.com
Using legacy password setting method
Successfully mapped host/lpar01.zos.myco.com to lpar01.
Key created.
Key created.
Key created.
Key created.
Key created.
Output keytab to lpar01.keytab:
Keytab version: 0x502
keysize 72 host/lpar01.zos.myco.com@AD.MYCO.COM ptype 1 (KRB5_NT_PRINCIPAL) vno 3 etype
0x1 (DES-CBC-CRC) keylength 8 (0xec409bbf1ad3e670)
keysize 72 host/lpar01.zos.myco.com@AD.MYCO.COM ptype 1 (KRB5_NT_PRINCIPAL) vno 3 etype
0x3 (DES-CBC-MD5) keylength 8 (0xec409bbf1ad3e670)
keysize 80 host/lpar01.zos.myco.com@AD.MYCO.COM ptype 1 (KRB5_NT_PRINCIPAL) vno 3 etype
0x17 (RC4-HMAC) keylength 16 (0x48a0917d773a94229c6b3231429349ab)
keysize 96 host/lpar01.zos.myco.com@AD.MYCO.COM ptype 1 (KRB5_NT_PRINCIPAL) vno 3 etype
0x12 (AES256-SHA1) keylength 32
(0xbb3709357b595cb15c6975d8964b74303b1f93237effe99ed7619cb4058b9fcb)
keysize 80 host/lpar01.zos.myco.com@AD.MYCO.COM ptype 1 (KRB5_NT_PRINCIPAL) vno 3 etype
0x11 (AES128-SHA1) keylength 16 (0xd309d8c89f8e218a3dd7705787856611)
PS C:\Users\Administrator>
```

To verify that the `ktpass` command worked correctly, select the new user in the Active Directory Administrative Center and click on properties. The account should have been updated with the User UPN logon field set to `host/lpar01.zos.myco.com@AD.MYCO.COM`.

3.3.1.3 Merge the generated keytab file with the z/OS krb5.keytab

Transfer the keytab file to the z/OS server as a **binary** file. On z/OS, use the `keytab` command to merge the service principal into `/etc/skrb/krb5.keytab`.

For example:

```
IBMUSER:/etc/skrb: >/usr/lpp/skrb/bin/keytab merge lpar01.keytab
EUVF06152E Principal host/lpar01.zos.myco.com@AD.MYCO.COM version 3 exists in keytab
/etc/skrb/krb5.keytab.
EUVF06154I Encryption type 23 is not supported.
```

Note: Keytab files should be treated securely. Once the file has been merged into the z/OS `keytab.krb5` file, copies on the Windows and z/OS server should be removed in order to prevent unauthorized access.

3.3.2 Linux OpenSSH Server Service Principal

The Linux service principle is created the same way as the SPN for the z/OS server. The only difference is with the command used on Linux to merge the keytab file with the `/etc/krb5.keytab`.

3.3.2.1 Create an Active Directory User for the Linux Service Principal

On the Windows Server, create an Active Directory user for the host using the same steps shown above for a z/OS host. For this example environment, the user name is `penguin`.

3.3.2.2 Configure the Service Principal Name and Generate a keytab

After the user has been created, execute the `ktpass` command to map the service principal to the user and to generate a keytab file.

Execute the following on the Windows domain controller from the Windows PowerShell.

```
ktpass princ host/srv01.linux.myco.com@AD.MYCO.COM mapuser ad\penguin
   -crypto all -pass password -ptype KRB5_NT_PRINCIPAL out penguin.keytab
```

3.3.2.3 Merge the generated keytab file with the Linux krb5.keytab

Transfer the keytab file to the Linux server as a binary file. On Linux, use the `ktutil` command to merge the service principal into `/etc/krb5.keytab`.

For example:

```
sudo ktutil
ktutil: rkt penguin.keytab
ktutil:  list
slot KVNO Principal
---- ---- ---------------------------------------------
1    8 host/srv01.linux.myco.com@AD.MYCO.COM
2    8 host/srv01.linux.myco.com@AD.MYCO.COM
3    8 host/srv01.linux.myco.com@AD.MYCO.COM
4    8 host/srv01.linux.myco.com@AD.MYCO.COM
5    8 host/srv01.linux.myco.com@AD.MYCO.COM
ktutil: wkt /etc/krb5.keytab
ktutil: q
```

Note: The `penquin.keytab` file should be treated securely. Once the file has been merged into the Linux `keytab.krb5` file, copies on the Windows and Linux server should be removed in order to prevent unauthorized access.

3.3.3 Windows SSH Server Service Principal

On the Windows Server, the service principal `host/winsrv01.ad.myco.com@AD.MYCO.COM` already exists. No additional configuration is required, because Kerberos is tightly integrated with Windows.

3.4 Verify the SSH Connections

The following tests all combinations of connections. Note that the z/OS and Linux clients must enter a Windows password in order to authenticate in the AD.MYCO.COM realm. See section 3.4.7 for information on optionally configuring a user keytab to eliminate the need to enter the Windows password.

3.4.1 z/OS Client SSH Connection to Linux

To connect to the Linux server from z/OS, first execute the `kinit aeneas` and provide the user's Windows password. This command will obtain a ticket granting ticket for the default realm, `AD.MYCO.COM`.

On the SSH command, specify the `GSSAPIAuthentication=yes` and `GSSAPIKeyExchange=yes` options if not already enabled in your `ssh_config`.

```
kinit aeneas
ssh -oGSSAPIAuthentication=yes -oGSSAPIKeyExchange=yes aeneas@srv01.linux.myco.com
```

Note: The `kdestroy` command is used to clear cached credentials.

3.4.2 z/OS Client SSH Connection to Windows

The commands to connect to the Windows SSH Server are the same as shown above for the Linux SSH server connection.

```
kinit aeneas
ssh -oGSSAPIAuthentication=yes -oGSSAPIKeyExchange=yes aeneas@winsrv01.ad.myco.com
```

3.4.3 Linux Client SSH Connection to z/OS

To connect to the z/OS server from Linux, first execute the `kinit` command. You will be prompted for your Windows password. This command will obtain a ticket granting ticket for the default realm, `AD.MYCO.COM`. On the SSH command, specify the `GSSAPIAuthentication=yes` and `GSSAPIKeyExchange=yes` options if not already enabled in your `ssh_config`. The credential presented to the z/OS server is `aeneas@AD.MYCO.COM`. The following KERBLINK configuration that was setup earlier on z/OS will allow login for the z/OS user id `HERCULES`:

```
RDEFINE KERBLINK /.../AD.MYCO.COM/aeneas APPLDATA('hercules')
```

```
kinit aeneas
ssh -oGSSAPIAuthentication=yes -oGSSAPIKeyExchange=yes hercules@lpar01.zos.myco.com
```

3.4.4 Linux Client SSH Connection to Windows

To connect to the Windows server from Linux, first execute the `kinit` command. You will be prompted for your Windows password. This command will obtain a ticket granting ticket for the default realm, `AD.MYCO.COM`. On the SSH command, specify the `GSSAPIAuthentication=yes` and `GSSAPIKeyExchange=yes` options if not already enabled in your `ssh_config`.

```
kinit aeneas
ssh -oGSSAPIAuthentication=yes -oGSSAPIKeyExchange=yes aeneas@winsrv01.ad.myco.com
```

3.4.5 Windows Client SSH Connection to Linux

When connecting from a Windows client, there is no step to obtain a ticket granting ticket because this was done during the Windows login process. The `klist` command can be used to review the list of currently cached tickets.

When connecting with PuTTY 0.64, the GSSAPI authentication option is enabled under Connection -> SSH -> Auth -> GSSAPI. Selecting the checkbox labeled Attempt GSSAPI authentication (SSH-2 only) is all that is required to connect with Kerberos. Because PuTTY, does not support GSSAPI Key Exchange, Windows workstations must continue to accept host keys.

If using the updated open source version of PuTTY with GSSAPI Key Exchange support available from https://marcussundberg.com/putty/, move the 'GSSAPI group exchange' algorithm to the top in the selection policy list on the Connection->SSH->Kex configuration page in addition to the GSSAPI configuration required for the standard version of PuTTY.

When connecting with VanDyke Software SecureFX 7.3.4, check the following settings in addition to specifying the hostname, port and username:

- In the Authentication box, select GSSAPI.
- In the Key exchange box, select Kerberos (Group Exchange). Note that this option must be the first enabled option in the list.

3.4.6 Windows Client SSH Connection to z/OS

The SSH client settings for connecting to z/OS are the same as listed above for a connection to a Linux SSH server. The credential presented to the z/OS server is `aeneas@AD.MYCO.COM`. The following KERBLINK configuration that was setup earlier on z/OS will convert the principal to allow login for the z/OS user id `HERCULES`: `RDEFINE KERBLINK /.../AD.MYCO.COM/aeneas APPLDATA('hercules')`

3.4.7 User keytab to eliminate Windows Password

To generate a keytab for a user, execute the ktpass command on the Windows domain controller from the Windows PowerShell.

The command output will look like the following. Note that the messages in bold can be ignored.

```
PS C:\Users\Administrator> ktpass princ aeneas@AD.MYCO.COM -crypto all
-pass password -ptype KRB5_NT_PRINCIPAL out aeneas.keytab
NOTE: creating a keytab but not mapping principal to any user.
For the account to work within a Windows domain, the
principal must be mapped to an account, either at the
domain level (with /mapuser) or locally (using ksetup)
If you intend to map aeneas@AD.MYCO.COM to an account through other means
or don't need to map the user, this message can safely be ignored.
WARNING: pType and account type do not match. This might cause problems.
Key created.
Key created.
Key created.
```

```
Key created.
Key created.
Output keytab to aeneas.keytab:
Keytab version: 0x502
keysize 51 aeneas@DTLDC.DOVETAIL.COM ptype 1 (KRB5_NT_PRINCIPAL) vno 1 etype 0x1 (DES-
CBC-CRC) keylength 8 (0x9da4c458f4d5755e)
keysize 51 aeneas@DTLDC.DOVETAIL.COM ptype 1 (KRB5_NT_PRINCIPAL) vno 1 etype 0x3 (DES-
CBC-MD5) keylength 8 (0x9da4c458f4d5755e)
keysize 59 aeneas@DTLDC.DOVETAIL.COM ptype 1 (KRB5_NT_PRINCIPAL) vno 1 etype 0x17 (RC4-
HMAC) keylength 16 (0x48a0917d773a94229c6b3231429349ab)
keysize 75 aeneas@DTLDC.DOVETAIL.COM ptype 1 (KRB5_NT_PRINCIPAL) vno 1 etype 0x12
(AES256-SHA1) keylength 32
(0x91881ecd255d95ae9e2100328cb7ac8a0f7ae2f5bafb75094e3eefee40b1b3c1)
keysize 59 aeneas@DTLDC.DOVETAIL.COM ptype 1 (KRB5_NT_PRINCIPAL) vno 1 etype 0x11
(AES128-SHA1) keylength 16 (0x095b7e75bc2e1086418fd433f1ca3816)
```

On completion, transfer (in binary) the keytab to the user's home directory on the z/OS and Linux servers as needed. The keytab file should be writable and readable only by the user:

```
chmod 600 aeneas.keytab
```

With this file, executing `kinit –t aeneas.keytab` will authenticate the user to the AD.MYCO.COM realm without requiring the user to enter the Windows password. When the user's Windows password changes, the keytab will need to be regenerated and transferred to remote servers.

4 Eliminating SSH keys with Kerberos (Cross Realm)

Figure 3 shows the sample environment with the user and host keys removed by defining two Kerberos realms with a cross realm trust relationship.

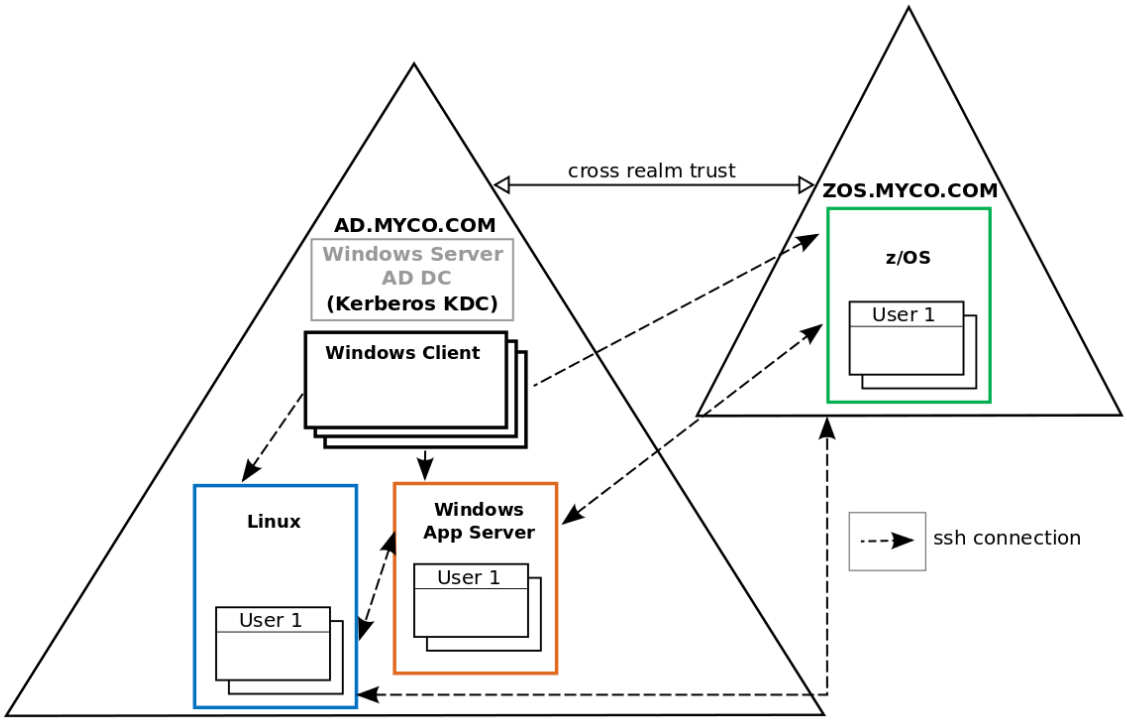


Figure 3 User and Host Keys Eliminated with Kerberos (cross realm)

The Windows Server AD DC defines the windows Kerberos realm (`AD.MYCO.COM`). The Windows Key Distribution Center (KDC) is available as part of the Microsoft Security Support Provider Interface (SSPI).

The Linux server has been updated by installing the Kerberos client and configuring its default realm to be `AD.MYCO.COM`.

The z/OS Network Authentication Service has been installed on the z/OS server defining a KDC and the z/OS realm `ZOS.MYCO.COM`.

Each SSH server (SSHD) and client configuration has been updated to use GSSAPIAuthentication and GSSAPIKeyExchange. The user's key pairs have been removed from user's control and are now managed by the Kerberos protocol in conjunction with the KDCs. Additionally, host keys have been removed by deleting `known_host` files.

Finally, the Windows and z/OS realms are configured with a trust (cross-realm) relationship.

The following sections describe in detail the software and configuration changes required. For simplicity, this example assumes that the username `aeneas` exists in the Windows domain and on the

Linux server. The z/OS username is `hercules`. This will show how to map usernames between the z/OS and Windows realms.

The high level tasks for configuring Kerberos for OpenSSH in our sample corporate environment are:

- Configure the z/OS Network Authentication Service (i.e. z/OS KDC)
- Configure a cross-realm (trust) relationship between the z/OS and Windows realms
- Configure the OpenSSH Servers on z/OS, Linux and Windows, enabling GSSAPI Authentication and Key Exchange
- Define Service Principals for the OpenSSH servers (SSHD) on z/OS, Linux and Windows
- Configure and test OpenSSH Client connections on z/OS, Linux and Windows

4.1 z/OS Network Authentication Configuration (SKRBKDC)

The primary reference for configuring the z/OS KDC is the following: z/OS V2R2 Integrated Security Services Authentication Services Administration - SC23-6786-01.

In order to configure and start a KDC for the sample environment, complete the steps from two sections in Chapter 2. Because RACF is the external security manager in our environment, steps related to NDBM are skipped. Additionally, steps specific to DCE are skipped because DCE is not installed. In our sample environment, the following are the two sections required for a minimum configuration:

Chapter 2 “Configuring Network Authentication Service”

- Make the program operational
- Security server configuration/Configuring the primary security server for the realm

Below are section/step specific notes for the sample environment.

Note: Generally, some or all z/OS LPARs sharing the same RACF database can be part of the same z/OS realm. In our sample environment, z/OS LPARs sharing our RACF database can be part of the z/OS realm, `ZOS.MYCO.COM`, if the LPAR is configured with a `krb5.conf` file matching the `krb5.conf` defined on `lpar01.zos.myco.com`. The SKRBKDC service must be running on each LPAR.

4.1.1.1 Make the program operational

For step 1 under the “Make the program operational” heading in Chapter 2:

b. When creating the `/etc/skrb/krb5.conf`, note the following:
   - In the `libdefaults` section of `/etc/skrb/krb5.conf`, set `kdc_use_tcp = 1`. This option configures Kerberos to use TCP rather than UDP when communicating with the KDC.
   - Configure the encryption types in `/etc/skrb/krb5.conf` to be compatible with Windows Servers. For Windows Server 2012, the following is recommended.

   Use the following contents when creating `/etc/skrb/krb5.conf`:

```
[libdefaults]
    default_realm = ZOS.MYCO.COM
```

```
    kdc_use_tcp = 1
    default_tkt_enctypes = aes256-cts-hmac-sha1-96,des-cbc-md5,des-cbc-crc
    default_tgs_enctypes = aes256-cts-hmac-sha1-96,des-cbc-md5,des-cbc-crc

[realms]
   ZOS.MYCO.COM = {
       kdc = lpar01.zos.myco.com:88
       admin_server = lpar01.zos.myco.com:749
   }

[domain_realm]
.ad.myco.com      = AD.MYCO.COM
ad.myco.com       = AD.MYCO.COM
.linux.myco.com   = AD.MYCO.COM
linux.myco.com    = AD.MYCO.COM
.zos.myco.com     = ZOS.MYCO.COM
zos.myco.com      = ZOS.MYCO.COM
```

c. When configuring the SKRBKDC environment variables, the default values can be used except for the encryption types. Change the following setting making encryption types supported by Windows available.

```
SKDC_TKT_ENCTYPES=aes256-cts-hmac-sha1-96,des-cbc-md5,des-cbc-crc
```

4.1.1.2 Security server configuration/Configuring the primary security server for the realm

For step 1 under "Configuring the primary security server for the realm":

b. Skip this section for now. The cross-realm configuration will be covered in the next section.

Step 3 can be skipped for this minimal configuration.

After completing the steps in these sections, check the system log to verify that the KDC was successfully started. Messages similar to the following should be displayed:

```
EUVF04001I Security server version 3.23, Service level OA46245.
EUVF04002I Security runtime version 3.23, Service level OA40957.
EUVF04158I Kerberos KDC services are enabled.
EUVF04018I Security server initialization complete.
EUVF04022I Security server start command processed.
EUVF04069I Listening for requests on network interface 172.31.2.86.
EUVF04058I System SERVER has joined the Kerberos security server group.
```

4.2 Kerberos Cross-Realm Configuration

In our sample environment, a cross-realm trust between the z/OS realm and the Windows realm will be established. Unlike the single realm case, with a local z/OS realm, z/OS SAF users can transparently be authenticated to Kerberos without requiring a password or user keytab file (see the z/OS kinit command with the –s option). Cross-realm configuration for z/OS, Linux and the Windows AD DC are summarized in the following sections.

4.2.1 z/OS Server

On z/OS, Kerberos is part of the Network Authentication Service component of Integrated Security Services. Three types of changes are required on z/OS to configure a cross-realm relationship with a Windows realm:

4.2.1.1 Create Peer Trust Relationship

Execute the following RACF commands to create a peer trust relationship between `AD.MYCO.COM` and `ZOS.MYCO.COM`. The password specified will also be used when configuring the trust on the Windows server. Note that the syntax showing `/.../` is exact.

```
RDEFINE REALM /.../ZOS.MYCO.COM/krbtgt/AD.MYCO.COM KERB(PASSWORD('trust-password'))
RALTER REALM /.../ZOS.MYCO.COM/krbtgt/AD.MYCO.COM
    KERB(ENCRYPT(NODES NODESD NODES3 AES128 AES256))

RDEFINE REALM /.../AD.MYCO.COM/krbtgt/ZOS.MYCO.COM KERB(PASSWORD('trust-password'))
RALTER REALM /.../AD.MYCO.COM/krbtgt/ZOS.MYCO.COM
    KERB(ENCRYPT(NODES NODESD NODES3 AES128 AES256))
```

4.2.1.2 Map Windows and z/OS Userids

For each Windows domain user that will SSH to a z/OSS SSHD server, define a mapping between the Windows user name and the z/OS user name. In the following example, `aeneas` is the windows domain user name and `hercules` is the z/OS user name.

```
RDEFINE KERBLINK /.../AD.MYCO.COM/aeneas APPLDATA('hercules')
```

4.2.1.3 Add Cross-realm to krb5.conf

Update the `/etc/skrb/krb5.conf` adding `AD.MYCO.COM` to the realm section and defining the cross-realm authentication paths between `ZOS.MYCO.COM` and `AD.MYCO.COM` in the `capath` section. The following is the completed z/OS `krb5.conf` file for the sample environment.

```
[libdefaults]
    default_realm = ZOS.MYCO.COM
    kdc_use_tcp = 1
    default_tkt_enctypes = aes256-cts-hmac-sha1-96,des-cbc-md5,des-cbc-crc
    default_tgs_enctypes = aes256-cts-hmac-sha1-96,des-cbc-md5,des-cbc-crc

[realms]
   AD.MYCO.COM = {
        kdc = server.ad.myco.com:88
        admin_server = server.ad.myco.com:749
   }

   ZOS.MYCO.COM = {
        kdc = lpar01.zos.myco.com:88
        admin_server = lpar01.zos.myco.com:749
   }

[capaths]
   ZOS.MYCO.COM = {
        AD.MYCO.COM = .
```

```
    }
    AD.MYCO.COM = {
        ZOS.MYCO.COM = .
    }
[domain_realm]
.ad.myco.com       = AD.MYCO.COM
ad.myco.com        = AD.MYCO.COM
.linux.myco.com    = AD.MYCO.COM
linux.myco.com     = AD.MYCO.COM
.zos.myco.com      = ZOS.MYCO.COM
zos.myco.com       = ZOS.MYCO.COM
```

4.2.2 Linux Server

Next, the Linux server is added to the Windows realm. The Linux server will be able to access the z/OS realm due to the trust relationship between the two realms.

4.2.2.1 Install the Kerberos Client

Depending on the Linux distribution of the server, use one of the following to install the Kerberos client.

```
yum install krb5-workstation
```

or

```
apt-get install krb5-user
```

4.2.2.2 Configure krb5.conf

After installing the Kerberos client on the Linux server, update `/etc/krb5.conf` with the cross-realm configuration. Based on our sample environment, the configuration looks like the following. The following is the completed Linux `krb5.conf` file for the sample environment.

```
[libdefaults]
   default_realm = AD.MYCO.COM
   default_tkt_enctypes=aes256-cts-hmac-sha1-96,des-cbc-md5,des-cbc-crc
   default_tgs_enctypes=aes256-cts-hmac-sha1-96,des-cbc-md5,des-cbc-crc
   permitted_enctypes=aes256-cts-hmac-sha1-96,des-cbc-md5,des-cbc-crc

[realms]
   AD.MYCO.COM = {
      kdc = server.ad.myco.com:88
      admin_server = server.ad.myco.com:749
      default_domain = ad.myco.com
      auth_to_local = RULE:[1:$1@$0](.*@ZOS.MYCO.COM)s/@.*//
      auth_to_local = DEFAULT
   }

   ZOS.MYCO.COM = {
      kdc = lpar01.zos.myco.com:88
      admin_server = lpar01.zos.myco.com:749
      default_domain = zos.myco.com
   }

[capaths]
```

```
    ZOS.MYCO.COM = {
        AD.MYCO.COM = .
    }

    AD.MYCO.COM = {
        ZOS.MYCO.COM = .
    }

[domain_realm]
.ad.myco.com       = AD.MYCO.COM
ad.myco.com        = AD.MYCO.COM
.linux.myco.com    = AD.MYCO.COM
linux.myco.com     = AD.MYCO.COM
.zos.myco.com      = ZOS.MYCO.COM
zos.myco.com       = ZOS.MYCO.COM
```

Note that the `realms` and `capaths` sections of the Linux `krb5.conf` file are almost identical to the z/OS `/etc/skrb/krb5.conf` file. The differences are two additional lines, `auth_to_local,` in the `AD.MYCO.COM` realm section. The `auth_to_local` setting provides a general rule for mapping principal names from the `ZOS.MYCO.COM` realm to local user names in the `AD.MYCO.COM` realm. The same mapping was done in reverse for each user on z/OS using `RDEFINE KERBLINK` commands in section 4.2.1.2 .

4.2.3 Windows Server

On the Windows Server, connect the z/OS KDC to the Windows realm.

4.2.3.1 Configure forward and reverse lookup zones for the z/OS KDC

In our sample environment, the Windows AD DC also is a DNS. Execute the commands listed below in the Windows Power Shell of the Windows Server 2012. Alternatively, the DNS configuration can be completed using the DNS Manager user interface on the Windows Server 2012.

4.2.3.1.1 Configure a forward lookup zone

First create a new zone. The zone name for the sample environment is `zos.myco.com`.

```
dnscmd /zoneadd zos.myco.com /dsprimary
```

Complete the new zone, by making the following additional changes:

- Create a new host entry. `server` is the hostname of the z/OS KDC in the sample. The IP address of the z/OS KDC server is `172.31.2.86`.

  ```
  dnscmd /recordadd zos.myco.com server A 172.31.2.86
  ```

- Create a new text entry that defines the name of the Kerberos realm. Specify `_kerberos` as the Record Name. The realm name is `ZOS.MYCO.COM` in our sample.

  ```
  dnscmd /recordadd zos.myco.com _kerberos TXT lpar01.zos.myco.com
  ```

- Finally, create two service location records, one for protocol `_tcp` and `_udp`. Specify `_kerberos` for the service. The host name in our sample is `lpar01.zos.myco.com`.

```
dnscmd /recordadd zos.myco.com _kerberos._tcp SRV 0 0 88 lpar01.zos.myco.com

dnscmd /recordadd zos.myco.com _kerberos._udp SRV 0 0 88  lpar01.zos.myco.com
```

4.2.3.1.2 Configure a reverse lookup zone

Create a reverse look up zone. The Zone Name for the sample environment is `2.31.172.in-addr.arpa`.

```
dnscmd /zoneadd 2.31.172.in-addr.arpa /primary
```

Create a new pointer record to associate the z/OS host name with its IP address, `172.31.2.86` in our sample environment.

```
dnscmd /recordadd 2.31.172.in-addr.arpa 86 PTR lpar01.zos.myco.com
```

4.2.3.2 Configure a domain trust

This step is the Windows equivalent of the z/OS section 4.2.1.1, Create Peer Trust Relationship on z/OS.

Execute the commands listed below in the Windows Power Shell of the Windows Server. Alternatively, the domain trust can be configured with the Active Directory Domains and Trusts user interface.

The `netdom` command defines a non-transitive, two way, realm trust. The password specified must match the password provided when defining the cross-realm relationship to the z/OS server.

```
netdom trust ad.myco.com /d:ZOS.MYCO.COM /add /realm /twoway
  /passwordt:trust-password
```

The `ksetup` command is used to set the encryption type for communication with the z/OS realm. AES encryption is required in order to connect from the `AD.MYCO.COM` domain to the `ZOS.MYCO.COM` domain. Without this setting, the SSH client will report the following error: KDC has no support for encryption type.

```
ksetup /setenctypeattr ZOS.MYCO.COM AES256-CTS-HMAC-SHA1-96
```

Note that this command does not set the “The other domain supports Kerberos AES Encryption” checkbox in the Active Directory Domains and Trusts user interface. However, executing the command and/or setting the checkbox in the user interface will work.

4.2.4 Windows Workstation Configuration (run as Administrator)

The information captured in the `krb5.conf` on Linux and z/OS is configured using the Windows `ksetup` command which writes to the Widows registry. This configuration must be completed on each workstation with users that will connect to servers in the z/OS realm.

- Define the z/OS KDC to the workstation

```
ksetup /addkdc ZOS.MYCO.COM lpar01.zos.myco.com
```

- Force Windows to use TCP rather than UDP for KDC communication

```
ksetup /setrealmflags ZOS.MYCO.COM tcpsupported
```

- Define host to realm mappings

```
ksetup /addhosttorealmmap lpar01.zos.myco.com  ZOS.MYCO.COM
```

4.3 OpenSSH Server GSSAPI Authentication and Key Exchange Configuration

On each server, modify the OpenSSH client and server configuration to use GSSAPI.

4.3.1 z/OS OpenSSH Server

For z/OS, update the `/etc/ssh/sshd_config` file setting the following.

```
GSSAPIAuthentication yes
GSSAPIKeyExchange yes
GSSAPICleanupCredentials yes
```

In this example, the SSH server (SSHD) is assumed to be running under the SSHDAEM userid. If you have not already defined the SSHDAEM userid, use the following commands:

```
ADDUSER SSHDAEM DFLTGRP(SYS1) PASSWORD(tmppass)
     OMVS(UID(0) PROGRAM('/bin/sh') HOME('/') SHARED)
PERMIT BPX.DAEMON CLASS(FACILITY) ID(SSHDAEM) ACCESS(READ)
SETROPTS RACLIST(FACILITY) REFRESH
```

Additionally, because the KDC is local in the sample environment, ensure that the SSHD server process has the following variable set:

```
export KRB5_SERVER_KEYTAB=2
```

After making these configuration changes, restart SSHD.

4.3.2 z/OS OpenSSH Client

Configuring Kerberose for the OpenSSH client on z/OS requires two steps:

4.3.2.1 Update the SSH client configuration file

For z/OS, update the `/etc/ssh/ssh_config` file setting the following.

Optionally, enable the following in `ssh_config`:

```
GSSAPIAuthentication yes
GSSAPIKeyExchange yes
GSSAPIDelegateCredentials yes
```

If global configuration is not desired, this option can either be added to the user's `.ssh/ssh_config` file or on the ssh command line. For example:

```
ssh -oGSSAPIAuthentication=yes -oGSSAPIKeyExchange=yes aeneas@srv01.linux.myco.com
```

4.3.2.2 Map z/OS userids to Kerberos principal names (Windows userids)

Alter any existing z/OS userids that will be used to run SSH clients with Kerberos. In the example below, `HERCULES` is the z/OS userid and `aeneas` is the Windows userid.

```
ALTUSER HERCULES PASSWORD(newpassword) NOEXPIRED KERB(KERBNAME('aeneas'))
```

Note: The user's password must change in order to ensure a Kerberos secret key is generated.

4.3.3 Linux OpenSSH Server

For Linux, update the `/etc/ssh/sshd_config` file setting the following. After making the configuration change, restart SSHD.

```
GSSAPIAuthentication yes
GSSAPIKeyExchange yes
GSSAPICleanupCredentials yes
```

4.3.4 Linux OpenSSH Client

For Linux, update the `/etc/ssh/ssh_config`:

```
GSSAPIAuthentication yes
GSSAPIKeyExchange yes
GSSAPIDelegateCredentials yes
```

If global configuration is not desired, this option can either be added to the user's `.ssh/ssh_config` file or on the `ssh` command line. For example:

```
ssh -oGSSAPIAuthentication=yes -oGSSAPIKeyExchange=yes aeneas@winsrv01.ad.myco.com
```

4.3.5 Windows SSH Server

On the Windows Server, the SSH Server must support GSSAPI Authentication and Key Exchange. The following product was tested:

- VanDyke Software VShell 4.1 server enables GSSAPI Authentication by default. Kerberos must be enabled for Key Exchange by selecting the Kerberos algorithms on the SSH2->Key Exchange configuration page.

Note: Microsoft has announced plans to provide ported version of the OpenSSH server and client to its PowerShell product. See http://blogs.msdn.com/b/powershell/archive/2015/10/19/openssh-for-windows-update.aspx Early versions of this code can be downloaded from GitHub: https://github.com/PowerShell/Win32-OpenSSH

4.3.6 Windows SSH Client

For each windows domain user, use the Active Directory Administrative Center to **enable AES encryption options (this is required)**. This setting is under Encryption Options. Additionally, SSH Client software must be installed. The following clients were tested:

- The standard distribution of the PuTTY 0.64 client supports GSSAPI Authentication, but not Key Exchange. An updated open source version of PuTTY with GSSAPI Key Exchange support is available from https://marcussundberg.com/putty/.
- The VanDyke Software SecureFX 7.3.4 client supports GSSAPI Authentication and Key Exchange.

4.4 Service Principal Configuration

A service principal name (SPN) uniquely defines an instance of a service within a network. Each SSH Server running on z/OS, Linux and Windows is assigned an SPN. The SPN name has the following format:

```
host/fully.qualified.hostname@REALM
```

Each server has a different way to configure a service principal name.

4.4.1 z/OS OpenSSH Server Service Principal

The z/OS SSHD service principal is defined in the `ZOS.MYCO.COM` realm. The following commands define the SPN `host/lpar01.zos.myco.com@ZOS.MYCO.COM`:

```
ALTUSER SSHDAEM DFLTGRP(SYS1) PASSWORD(password) NOEXPIRED
     KERB(KERBNAME('host/lpar01.zos.myco.com'))
ALTUSER SSHDAEM NOPASSWORD
```

Note: The service principal must be defined with a non-expiring password which is why the `NOEXPIRED` option is specified. Additionally, in order to generate a Kerberos key, the password must change. Once the key has been generated, the SSHDAEM user no longer needs a password so the `ALTUSER` command is executed again with `NOPASSWORD`.

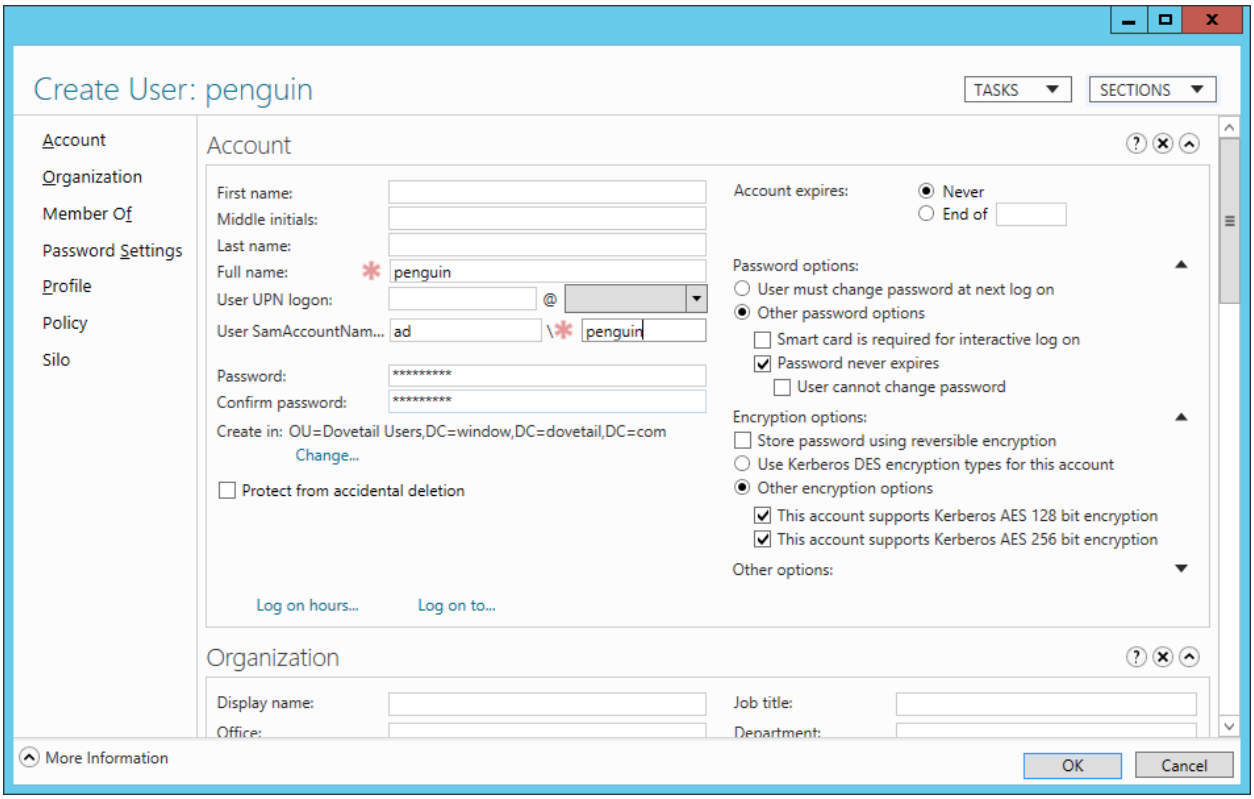
4.4.2 Linux OpenSSH Server Service Principal

Because the Linux server is being added to the Windows realm, creating a service principal for a Linux server requires steps to be executed on the Windows Server as well as the Linux server.

4.4.2.1 Create an Active Directory User for the Linux Service Principal

On the Windows Server, create an Active Directory user for the host (for example: penguin. Create the user using the Active Directory Administrative Center user interface (Server Manager ->tools-> Active Directory Administrative Center). The screenshot below shows the minimum information required:

- Full name
- User SamAccountName
- Password and confirmed password
- Password never expires
- Encryption options: select both AES options *(Note: this setting is required for Kerberos to work with z/OS.)*



4.4.2.2 Configure the Service Principal Name and Generate a keytab

After the user has been created, execute the `ktpass` command to map the service principal to the user and to generate a keytab file.

Execute the following on the Windows domain controller from the Windows PowerShell.

```
ktpass princ host/srv01.linux.myco.com@AD.MYCO.COM mapuser ad\penguin
   -crypto all -pass password -ptype KRB5_NT_PRINCIPAL out penguin.keytab
```

The command output will look like the following. Note specifically, the message in bold below.

```
PS C:\Users\Administrator> ktpass princ host/srv01.linux.myco.com@AD.MYCO.COM mapuser
ad\penguin -crypto all -pass xxxxxxx -ptype KRB5_NT_PRINCIPAL out penguin.keytab
Targeting domain controller: server.ad.myco.com
Using legacy password setting method
Successfully mapped host/srv01.linux.myco.com to penguin.
Key created.
Key created.
Key created.
Key created.
Key created.
Output keytab to penguin.keytab:
Keytab version: 0x502
keysize 72 host/srv01.linux.myco.com@AD.MYCO.COM ptype 1 (KRB5_NT_PRINCIPAL) vno 3 etype
0x1 (DES-CBC-CRC) keylength 8 (0xec409bbf1ad3e670)
keysize 72 host/srv01.linux.myco.com@AD.MYCO.COM ptype 1 (KRB5_NT_PRINCIPAL) vno 3 etype
0x3 (DES-CBC-MD5) keylength 8 (0xec409bbf1ad3e670)
keysize 80 host/srv01.linux.myco.com@AD.MYCO.COM ptype 1 (KRB5_NT_PRINCIPAL) vno 3 etype
0x17 (RC4-HMAC) keylength 16 (0x48a0917d773a94229c6b3231429349ab)
```

```
keysize 96 host/srv01.linux.myco.com@AD.MYCO.COM ptype 1 (KRB5_NT_PRINCIPAL) vno 3 etype
0x12 (AES256-SHA1) keylength 32
(0xbb3709357b595cb15c6975d8964b74303b1f93237effe99ed7619cb4058b9fcb)
keysize 80 host/srv01.linux.myco.com@AD.MYCO.COM ptype 1 (KRB5_NT_PRINCIPAL) vno 3 etype
0x11 (AES128-SHA1) keylength 16 (0xd309d8c89f8e218a3dd7705787856611)
PS C:\Users\Administrator>
```

To verify that the `ktpass` command worked correctly, select the new user in the Active Directory Administrative Center and click on properties. The account should have been updated with the User UPN logon field set to `host/srv01.linux.myco.com@AD.MYCO.COM`.

4.4.2.3 Merge the generated keytab file with the Linux krb5.keytab

Transfer the keytab file to the Linux server as a binary file. On Linux, use the `ktutil` command to merge the service principal into `/etc/krb5.keytab`.

For example:

```
sudo ktutil
ktutil: rkt penguin.keytab
ktutil:  list
slot KVNO Principal
---- ---- ---------------------------------------
1    8 host/srv01.linux.myco.com@AD.MYCO.COM
2    8 host/srv01.linux.myco.com@AD.MYCO.COM
3    8 host/srv01.linux.myco.com@AD.MYCO.COM
4    8 host/srv01.linux.myco.com@AD.MYCO.COM
5    8 host/srv01.linux.myco.com@AD.MYCO.COM
ktutil: wkt /etc/krb5.keytab
ktutil: q
```

Note: The `penquin.keytab` file should be treated securely. Once the file has been merged into the Linux `keytab.krb5` file, copies on the Windows and Linux server should be removed in order to prevent unauthorized access.

4.4.3 Windows SSH Server Service Principal

On the Windows Server, the service principal `host/winsrv01.ad.myco.com@AD.MYCO.COM` already exists. No additional configuration is required, because Kerberos is tightly integrated with Windows.

4.5 Verify the SSH Connections

Before running the following connection tests, restart the started task for the z/OS KDC to make sure that all configuration changes are in effect.

4.5.1 z/OS Client SSH Connection to Linux

To connect to the Linux server from z/OS, first execute the `kinit` command with KERBNAME userid. If the `-s` option is used on the `kinit` command, the command completes without prompting for a password. This command will obtain a ticket granting ticket for the default realm, `ZOS.MYCO.COM`.

On the SSH command, specify the `GSSAPIAuthentication=yes` and `GSSAPIKeyExchange=yes` options if not already enabled in your `ssh_config`. Since this is a cross-realm connection, the credential presented to the Linux server is `aeneas@ZOS.MYCO.COM`. The `auth_to_local` configuration in the Linux `krb5.conf` will convert the principal to `aeneas@AD.MYCO.COM` before authenticating. If `aeneas` is not defined on the linux server, `aeneas` can be mapped to a linux user with a `.k5login` file which resides in the linux user's home directory. This file contains a list of Kerberose principals that are allowed access to the linux user's account. In this scenario, the file would contain: `aeneas@ZOS.MYCO.COM`.

```
kinit -s
ssh -oGSSAPIAuthentication=yes -oGSSAPIKeyExchange=yes aeneas@srv01.linux.myco.com
```

Note: The `kdestroy` command is used to clear cached credentials.

4.5.2 z/OS Client SSH Connection to Windows

The commands to connect to the Windows SSH Server are the same as shown above for the Linux SSH server connection.

```
kinit -s
ssh -oGSSAPIAuthentication=yes -oGSSAPIKeyExchange=yes aeneas@winsrv01.ad.myco.com
```

4.5.3 Linux Client SSH Connection to z/OS

To connect to the z/OS server from Linux, first execute the `kinit` command. You will be prompted for your Windows password. This command will obtain a ticket granting ticket for the default realm, `AD.MYCO.COM`. On the SSH command, specify the `GSSAPIAuthentication=yes` and `GSSAPIKeyExchange=yes` options if not already enabled in your `ssh_config`. Since this is a cross-realm connection, the credential presented to the z/OS server is `aeneas@AD.MYCO.COM`. The following KERBLINK configuration that was setup earlier on z/OS will convert the principal to `aeneas@ZOS.MYCO.COM` and allow login for the z/OS user id `HERCULES`:

```
RDEFINE KERBLINK /.../AD.MYCO.COM/aeneas APPLDATA('hercules')
```

```
kinit aeneas
ssh -oGSSAPIAuthentication=yes -oGSSAPIKeyExchange=yes hercules@lpar01.zos.myco.com
```

Note: See 3.4.7 for information on optionally eliminating the need to enter the Windows password by using a user keytab.

4.5.4 Linux Client SSH Connection to Windows

To connect to the Windows server from Linux, first execute the `kinit` command. You will be prompted for your Windows password. This command will obtain a ticket granting ticket for the default realm, `AD.MYCO.COM`. On the SSH command, specify the `GSSAPIAuthentication=yes` and `GSSAPIKeyExchange=yes` options if not already enabled in your `ssh_config`. This connection is not a cross-realm connection because both the Linux server and the Windows server are in the same realm.

```
kinit aeneas
ssh -oGSSAPIAuthentication=yes -oGSSAPIKeyExchange=yes aeneas@winsrv01.ad.myco.com
```

Note: See 3.4.7 for information on optionally eliminating the need to enter the Windows password by using a user keytab.

4.5.5 Windows Client SSH Connection to Linux

When connecting from a Windows client, there is no step to obtain a ticket granting ticket because this was done during the Windows login process. The `klist` command can be used to review the list of currently cached tickets.

When connecting with PuTTY 0.64, the GSSAPI authentication option is enabled under Connection -> SSH -> Auth -> GSSAPI. Selecting the checkbox labeled Attempt GSSAPI authentication (SSH-2 only) is all that is required to connect with Kerberos. Because PuTTY, does not support GSSAPI Key Exchange, Windows workstations must continue to use host keys. Additionally, the Linux and z/OS users continue to have a `known_hosts` file containing the Window's SSH Server host key.

If using the updated open source version of PuTTY with GSSAPI Key Exchange support available from https://marcussundberg.com/putty/, move the 'GSSAPI group exchange' algorithm to the top in the selection policy list on the Connection->SSH->Kex configuration page in addition to the GSSAPI configuration required for the standard version of PuTTY.

When connecting with VanDyke Software SecureFX 7.3.4, check the following settings in addition to specifying the hostname, port and username:

- In the Authentication box, select GSSAPI.
- In the Key exchange box, select Kerberos (Group Exchange). Note that this option must be the first enabled option in the list.

4.5.6 Windows Client SSH Connection to z/OS

The SSH client settings for connecting to z/OS are the same as listed above for a connection to a Linux SSH server. However, since the z/OS connection is a cross-realm connection, the credential presented to the z/OS server is `aeneas@AD.MYCO.COM`. The following KERBLINK configuration that was setup earlier on z/OS will convert the principal to `hercules@ZOS.MYCO.COM` and allow login for the z/OS user id `HERCULES`:

```
RDEFINE KERBLINK /.../AD.MYCO.COM/aeneas APPLDATA('hercules')
```

5 Troubleshooting

The following are some items that may be helpful when troubleshooting connection problems:

- A wire trace such as WireShark and/or a z/OS Packet Trace can be used to verify whether
  - network traffic is being dropped by firewalls or other reasons
  - servers are reporting the correct reverse DNS name
  - correct service principal names are being sent in KRB packets

- The following option can be used to override the hostname used to build the host service principal name. This can be especially useful in a test environment.

```
ssh -oGSSAPIAuthentication=yes -oGSSAPIServerIdentity=winsrv01.ad.myco.com
   aeneas@winsrv01.ad.myco.com
```

- Review firewall configurations to ensure that the following ports are open:

```
kerberos       88/udp
kerberos       88/tcp
kpasswd       464/udp
kpasswd       464/tcp
kerberos-adm 749/tcp
krb5_prop     754/tcp
```

- Kerberos tracing can be enabled for a Linux SSHD server by setting the environment variable `KRB5_TRACE` and running SSHD on a test port as root. For example:

```
KRB5_TRACE=/tmp/sshd_kerb.log /usr/sbin/sshd -p 4022 -eD
  -oLogLevel=DEBUG3 -oGSSAPIAuthentication=yes -oGSSAPIKeyExchange=yes
```

- Kerberos tracing can be enabled for a z/OS SSH client and server by setting the following environment variables. For Example:

```
export _EUV_SVC_MSG_LEVEL=VERBOSE
export _EUV_SVC_MSG_LOGGING=STDERR_LOGGING
export _EUV_SVC_DBG_MSG_LOGGING=1
export _EUV_SVC_DBG=*.9
export _EUV_SVC_DBG_FILENAME=/tmp/ kerb.log
```

  For Example, run a test SSH server on a test port:

```
/usr/sbin/sshd -eD -oLogLevel=DEBUG3 -p 4022  -oGSSAPIKeyExchange=yes -
oGSSAPIAuthentication=yes
```

6 Trademarks and Copyrights

RACF®, z/OS® and IBM® are trademarks of IBM Corporation.

Active Directory®, Microsoft ®, Windows ® are trademarks of Microsoft Corporation.

PuTTY is copyright 1997-2015 Simon Tatham.

VShell® and SecureFX® are trademarks or registered trademarks of VanDyke Software, Inc

Wireshark® is a trademark of SolarWinds Worldwide, LLC.