The Three Headed Dog Ate My SSH Keys!

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

Stephen Goetze
Kirk Wolf
Dovetailed Technologies, LLC

#SHAREorg

Trademarks and Attributions

- 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
- Cerberus image: By Tretinville [CC BY-SA 3.0 (http://creativecommons.org/licenses/by-sa/3.0)], via Wikimedia Commons

Agenda

- Discuss SSH key management
- Summarize Kerberos and its benefits
- Present a Use Case combining SSH and Kerberos:

  Single sign on with z/OS, Linux and Windows

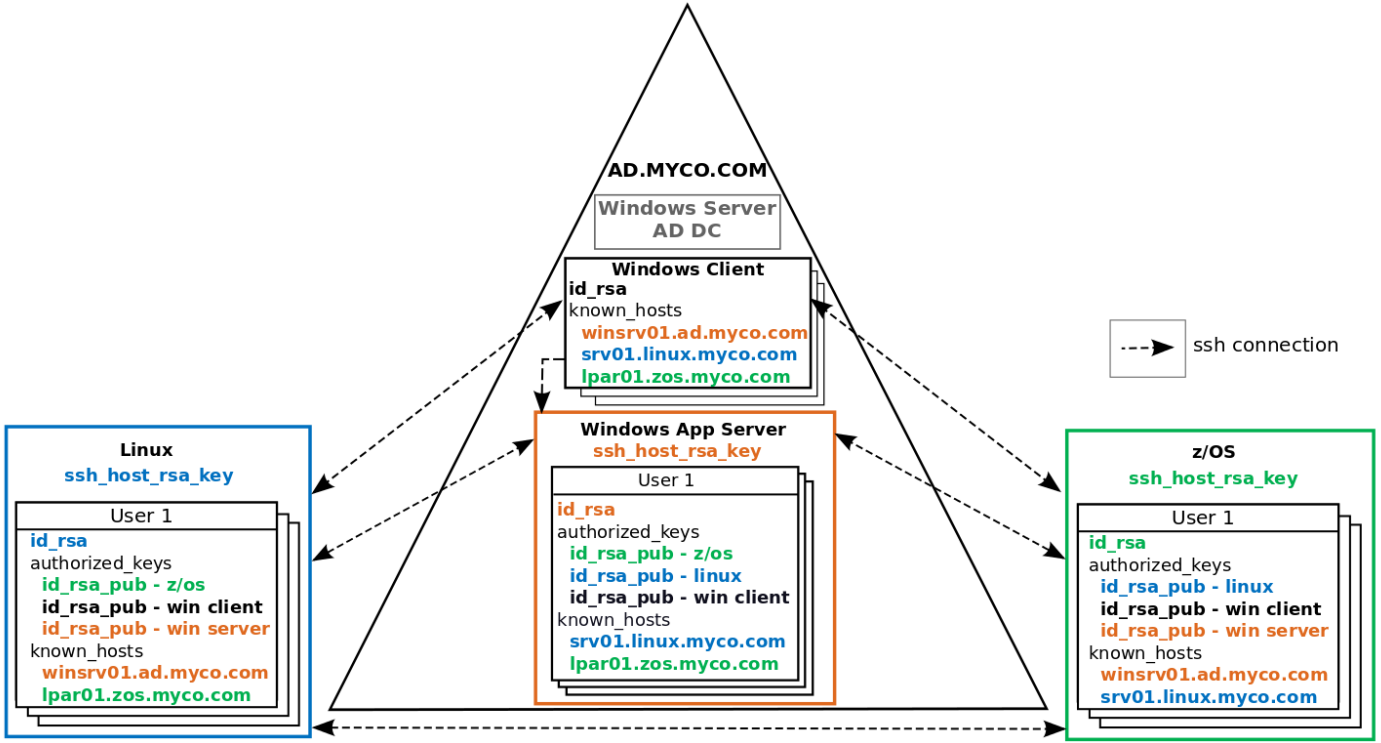
SSH Requires Mutual Authentication

- During Key Exchange
  - Server offers client its public key for **host authentication**
- During User Authentication
  - A **user** key pair can be used to authenticate the user to the server
- Authentication of both the client and server is referred to as *mutual* authentication

SSH Key Management

- SSH keys can be effective and more secure than passwords, and work well in small groups.
- Corporate environments can pose real challenges
- For example:
  - Users don't always keep their private keys secure
  - *Actual* verification of host keys is difficult
  - Standard SSH keys have no expiration
  - Issues grow with the number of nodes

SSH Key Proliferation



SSH Key Management Solutions

- IBM RACF digital certificates can add expiry to SSH keys
  - z/OS only
- Third party SSH key management products
  - Expensive, complex
- OpenSSH certificates
  - Not widely used
- X509 (PKI) certificates
  - Not part of RFC standard
  - Spotty support; none in OpenSSH
- How about…

Kerberos (Cerberus)

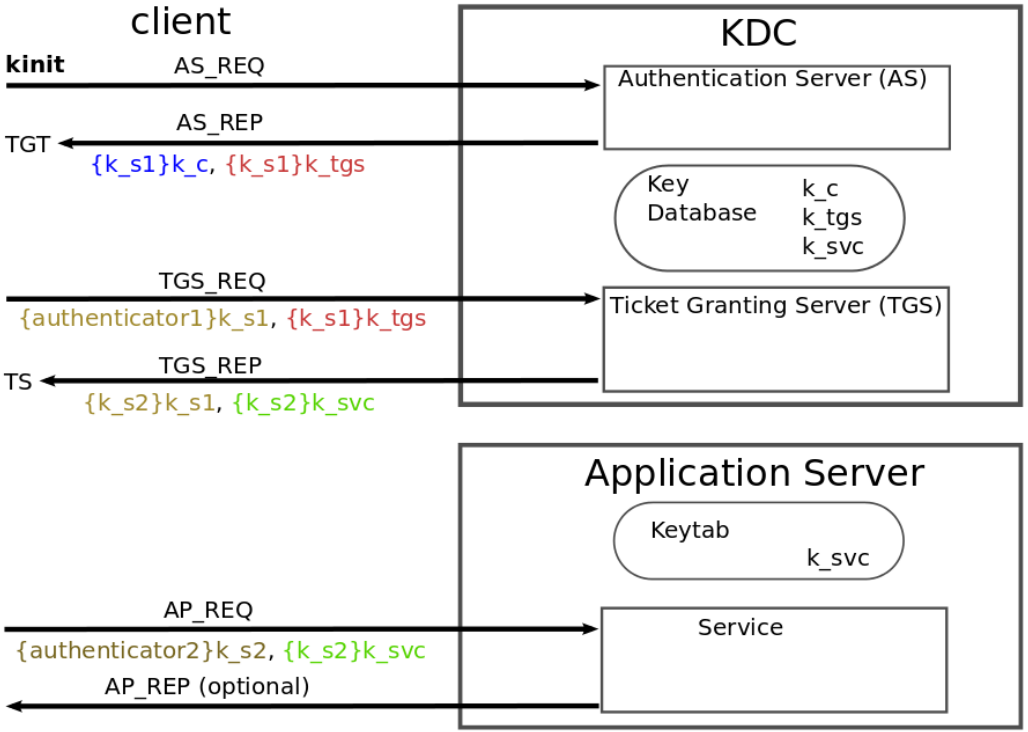
Kerberos

- MIT: “Kerberos is an authentication protocol for trusted hosts on untrusted networks”
- Three heads: Authentication, Authorization and Auditing
- *Principals* (users and services) exist in a *Realm*
  - *Cross Realm* authentication allows multiple Realms to interoperate
- Primary Features:
  - **Secure**: Passwords never transmitted in clear text
  - **Mutual Authentication**: Both client and server authenticate
  - **SSO**: One logon permits access to multiple resources
  - **Trusted Third Party**: Centralized server(s) manage secure keys
- Kerberos shares the primary SSH authentication goals and adds SSO and Key management

Kerberos Keys

- SSH has *user*, *host*, and *session* keys
- Kerberos has *user*, *service (host)*, and *session* keys
  - User and service keys are “long-term” keys, managed and stored by the Key Distribution Center (KDC)
  - The user can create a copy of his key from a password
  - The service keeps a copy of its key in a local, secure *keytab* file
- Communication between the user and service is encrypted with the session key created by the KDC

Kerberos Authentication (Conceptual)



Kerberos Feature Summary

- **Secure:** Passwords never transmitted in clear text
  - Client can create its key from a password
  - Service has its key in a keytab file
- **Mutual Authentication**: Both client and server authenticate
  - KDC creates session key signed by each party's long term key
  - Shared access to this key proves mutual authentication
- **SSO**: One logon permits access to multiple resources
  - Kinit is run once to get a Ticket Granting Ticket (TGT)
  - The TGT is used to get Service Tickets (TS) without requiring a password
- **Trusted Third Party**: Centralized server(s) manage secure keys
  - User and Service long term keys held by the KDC
  - Centralized handling of Expiration and Revocation

Previous chart is conceptual; see http://www.kerberos.org/software/tutorial.html and https://www.ietf.org/rfc/rfc4120.txt for complete details

OpenSSH Kerberos Integration

- OpenSSH has two integration paths:
- (Old) Kerberos options
  - These are server side (sshd) only options that will validate a user password through the server's Kerberos KDC
  - Rarely used, as the benefits are minor; no z/OS support
- GSSAPI options
  - Generic Security Service Application API
  - Enable both client (ssh) and server (sshd) integration with Kerberos

OpenSSH GSSAPI Options

- To enable base Kerberos integration, **both** the client and server must specify
  - **GSSAPIAuthentication**
  - Enables Kerberos *user* authentication
- To enable both *user* and *host* (mutual) authentication, both the client and server must specify
  - **GSSAPIKeyExchange**
  - This option is not available on all OpenSSH implementations, but is prevalent on Linux, and now available on z/OS

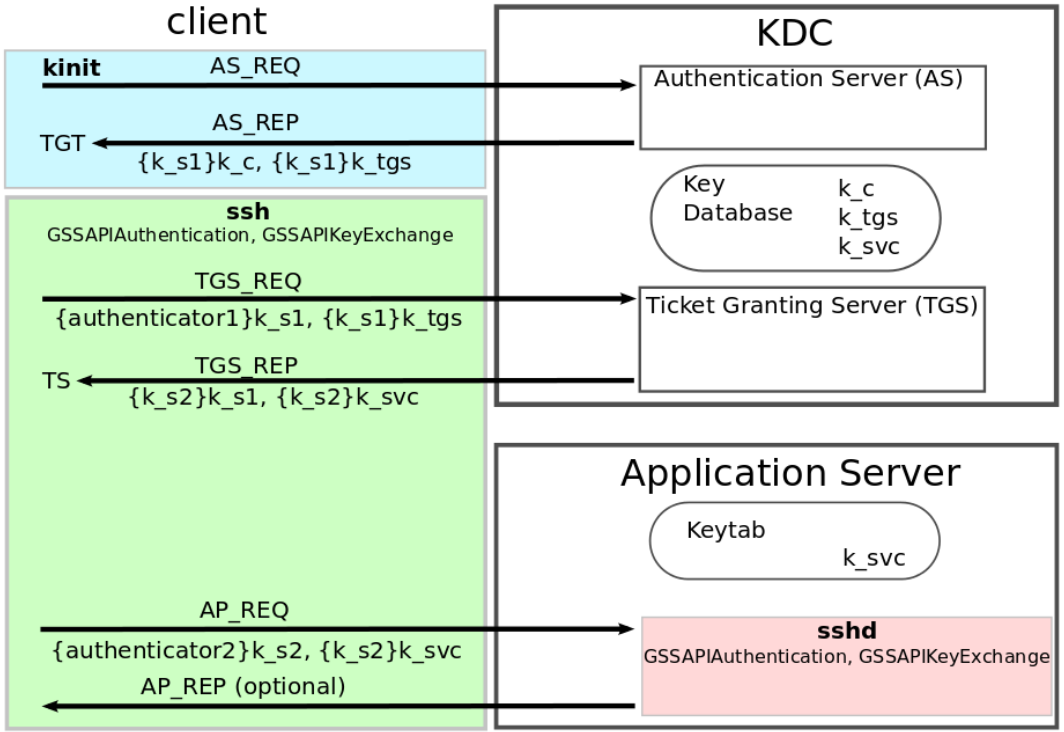
OpenSSH GSSAPI Server Options

- **GSSAPICleanupCredentials**
  - Specifies whether to automatically destroy user's credentials cache (the TGT) on logout
- **GSSAPIStrictAcceptorCheck**
  - Determines whether to be strict about the identity of the GSSAPI acceptor a client authenticates against. This facility is provided to assist with operation on multi homed machines
- **GSSAPIStoreCredentialsOnRekey**
  - Controls whether the user's GSSAPI credentials should be updated following a successful connection rekeying

OpenSSH GSSAPI Client Options

- **GSSAPIClientIdentity, GSSAPIServerIdentity**
  - Explicitly specifies the client/server identity
- **GSSAPIDelegateCredentials**
  - Forward credentials (the TGT) to the server (Note: on z/OS, the default kinit returns a non-forwardable TGT. Use “kinit –f”)
- **GSSAPIRenewalForcesRekey**
  - If set to “yes” then renewal of the client’s GSSAPI credentials will force the rekeying of the ssh connection
- **GSSAPITrustDns**
  - Set to “yes” to indicate that the DNS is trusted to securely canonicalize the name of the host being connected to

Kerberized SSH



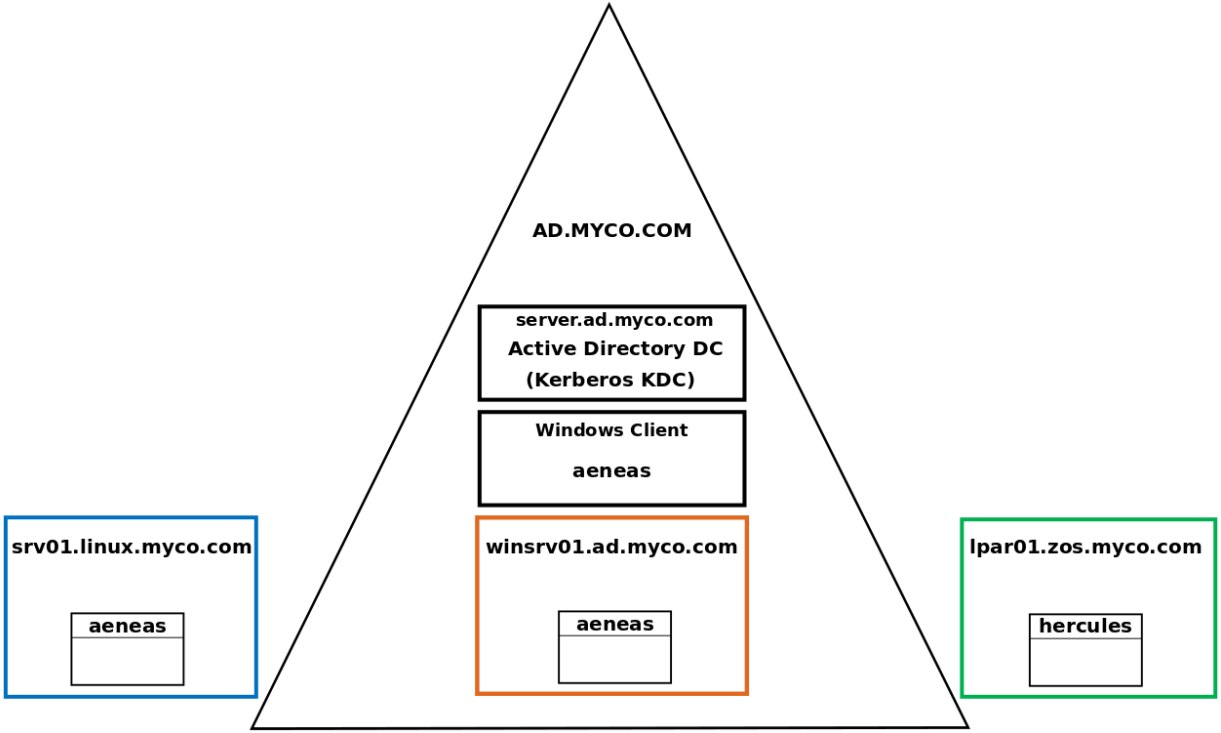
Reality Check

- If Kerberized SSH is so great, why isn't it everyone doing it?
- Historically, lack of widespread adoption due to:
  - Insufficient/incompatible crypto options
  - No z/OS solution
  - Lack of ubiquitous Windows SSH options
  - Complicated setup and administration

Making Keberos Easier to Adopt

- Crypto options have been upgraded across all platforms
  - aes256-cts-hmac-sha1-96
- IBM z/OS V2R2 OpenSSH – PTF UA79909 (Dec 2015)
  - Fully integrates z/OS OpenSSH with IBM's Network Authentication Service (Kerberos)
- Microsoft (Yes, Microsoft)
  - PowerShell will support OpenSSH client and server
  - Complexity is reduced by Kerberos integration in Active Directory / Domain Controller
    - Domain Controller *is Kerberos*

Use Case Environment



Implementing Kerberized SSH

- **Goal:** Enable OpenSSH in a Single Sign-On Corporate Environment with z/OS, Windows and Linux
- **Start with:** an existing Windows 2012 Server (Domain Controller)
  - Windows user **aeneas** (already Kerberized!)
  - Realm: AD.MYCO.COM
  - Key Distribution Center (KDC): server.ad.myco.com
- **Next:** Kerberize Windows SSH

Reference: http://dovetail.com/docs/ssh/kerberos_sso.pdf

Kerberize Windows SSH

- **Install:** A Windows SSH server product
  - VanDyke Software VShell 4.1 Server with SSH2->Key Exchange enabled
  - PowerShell OpenSSH (when available)
- **Add:** Windows SSH clients
  - PuTTY 0.64 with KeyExchange: https://marcussundberg.com/putty/
  - VanDyke Software SecureFX 7.3.4
  - PowerShell OpenSSH (when available)
- **Next:** Kerberize Linux OpenSSH

Kerberize Linux OpenSSH

- **Windows:** define service principal: host/srv01.linux.myco.com@AD.MYCO.COM
  - Create a new Windows user: **penguin -** Must explicitly select Kerberos AES encryption options
  - Create penguin.keytab using **ktpass** command and transfer to Linux server

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

- **Linux:**
  - Install Kerberos Client and update /etc/krb5.conf with the default realm name and Windows KDC
  - Merge penguin.keytab with /etc/krb5.keytab using **ktutil**

Update /etc/ssh/**ssh_config**

GSSAPIAuthentication yes
GSSAPIKeyExchange yes
GSSAPIDelegateCredentials yes

Update /etc/ssh/**sshd_config**

GSSAPIAuthentication yes
GSSAPIKeyExchange yes
GSSAPICleanupCredentials yes

- **Next:** Kerberize z/OS OpenSSH

Kerberize z/OS OpenSSH

- **Windows:** define service principal: host/lpar01.zos.myco.com@AD.MYCO.COM
  - Create a new Windows user: **zos -** Must explicitly select Kerberos AES encryption options
  - Create zos.keytab using **ktpass** command and transfer (in binary) to z/OS

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

- **z/OS:**
  - Create /etc/skrb/krb5.conf defining the default realm name and Windows KDC
  - Merge zos.keytab with /etc/skrb/krb5.keytab using **keytab**

Update /etc/ssh/**ssh_config**

GSSAPIAuthentication yes
GSSAPIKeyExchange yes
GSSAPIDelegateCredentials yes

Update /etc/ssh/**sshd_config**

GSSAPIAuthentication yes
GSSAPIKeyExchange yes
GSSAPICleanupCredentials yes

  - Map incoming principal to SAF user:

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

- **Next: Review and Test**

Linux and z/OS krb5.conf

```
[libdefaults]

      default_realm = AD.MYCO.COM
      default_tkt_enctypes = aes256-cts-hmac-sha1-96
      default_tgs_enctypes = aes256-cts-hmac-sha1-96
      permitted_enctypes = aes256-cts-hmac-sha1-96
      # Specify the following on z/OS only
      # kdc_use_tcp = 1

[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
```

Test SSH connections

```
// Starting from a Windows client
// Kerberos TGT obtained automatically during Windows logon
(PuTTY to srv01.linux.myco.com)
Login as: aeneus

// Linux
// TGT has been forwarded, so no kinit needed
aeneas@srv01:$~ ssh hercules@lpar01.zos.myco.com

// z/OS
/home/hercules> ssh aeneas@winsrv01.ad.myco.com

// Windows Server
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.
C:\Users\aeneas
```

OpenSSH GSSAPI Client Output

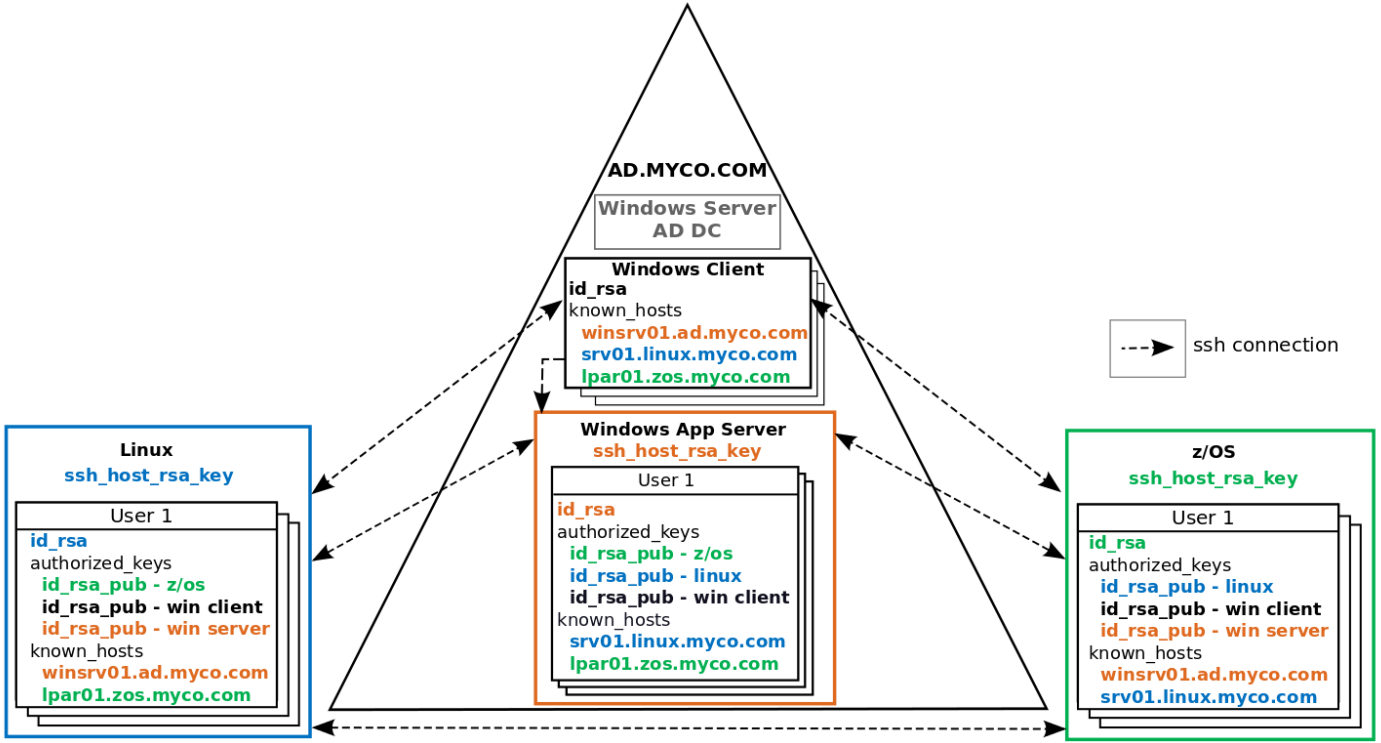
```
aeneas@srv01:$~ ssh -vv hercules@lpar01.zos.myco.com

(ssh)
debug1: Offering GSSAPI proposal: gss-gex-sha1-toWM5Slw5Ew8Mqkay+al2g==,…
debug1: SSH2_MSG_KEXINIT received
debug2: kex_parse_kexinit: gss-gex-sha1-toWM5Slw5Ew8Mqkay+al2g==,…
debug2: kex_parse_kexinit: ssh-rsa,ssh-dss,null
debug1: Doing group exchange
debug1: Calling gss_init_sec_context
debug1: Received GSSAPI_COMPLETE
debug1: Delegating credentials
debug1: Authentications that can continue: publickey,gssapi-keyex,gssapi-with-mic,password
debug1: Next authentication method: gssapi-keyex
debug2: we sent a gssapi-keyex packet, wait for reply
debug1: Authentication succeeded (gssapi-keyex).
```

OpenSSH GSSAPI Server Output

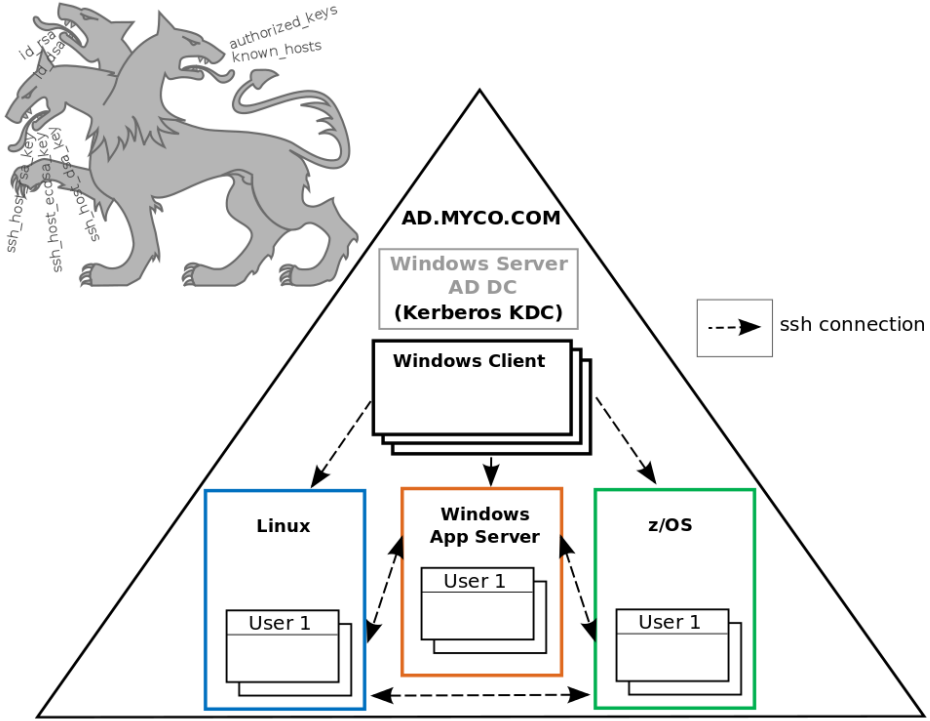
(sshd)

debug2: kexgss_server: Identifying gss-gex-sha1-toWM5Slw5Ew8Mqkay+al2g== [preauth]
debug2: kexgss_server: Acquiring credentials [preauth]
debug1: Doing group exchange [preauth]
debug1: Received some client credentials
debug1: userauth-request for user hercules service ssh-connection method gssapi-keyex [preauth]
debug2: input_userauth_request: try method gssapi-keyex [preauth]
Authorized to hercules, krb5 principal “aeneas@AD.MYCO.COM”

From SSH Keys…



… To Kerberized SSH!



Are We Done Yet?

- For many environments, a single Windows Realm solution is fine
  - GSSAPIAuthentication=yes -> Eliminates need for ssh **user** keys
  - GSSAPIKeyExchange=yes -> Eliminates need for ssh **host** keys
  - GSSAPIDelegateCredentials=yes -> SSO interactive ssh sessions all across the enterprise
  - Windows password is required for initial kinit

- What about z/OS batch jobs where interactive passwords can't be entered?
  - Kerberos **user** keytab files can be used to eliminate interactive password entry
    - Automatically expire when Windows password changes
  - Under z/OS, SAF principals can *transparently* authenticate against the local SKRBKDC using **kinit –s** (no password required)
    - Eliminates the need to use Windows passwords on z/OS (via kinit)
    - Requires a local z/OS Realm
    - Cross Realm Authentication with the Windows DC
    - See http://dovetail.com/docs/ssh/kerberos_sso.pdf for complete configuration details

References

- Kerberos Information
  - https://www.ietf.org/rfc/rfc4120.txt
  - http://www.kerberos.org/software/tutorial.html
  - Integrated Security Services Network Authentication Service Administration (SC23-6786-01)
- Dovetailed Technologies Resources (dovetail.com)
  - IBM Ported Tools OpenSSH / z/OS V2R2 OpenSSH - Quick Install Guide
    http://dovetail.com/docs/pt-quick-inst/index.html
  - Using OpenSSH in a Single Sign-on Corporate Environment with z/OS, Windows, and Linux
    http://dovetail.com/docs/ssh/kereros_sso.pdf
  - Dovetail webinar recordings:
    - IBM Ported Tools OpenSSH – Key Authentication
    - IBM Ported Tools OpenSSH – Using Key Rings
- IBM z/OS V2R2 OpenSSH: User's Guide (Order number: SC27-6806-01)
- PowerShell OpenSSH Information
  - http://blogs.msdn.com/b/powershell/archive/2015/10/19/openssh-for-windows-update.aspx
  - https://github.com/PowerShell/Win32-OpenSSH