Kerberos, AFS and SSH for your Understanding

Published messages

Apr	2003	We have installed a ssh v3.6.1p1 with Kerberos5 support on Linux.
Apr	2003	We have installed a new authentication method (Kerberos5) on Linux.
May	2003	The complete switch to Kerberos 5.
Feb	2004	Mail service up again (only SSL or Kerberos Authentication)
Jun	2004	quota limit for AFS home directories now 500 Mbyte
Jul	2004	ssh will use protocol version 2 by default.
Aug	2004	New Linux AFS fileserver. Now we have 15 TByte AFS space.

Why Kerberos 5?

and more about Kerberos 5

Why is AFS our strategy?

and more about AFS

Why SSH Protocol 2?

and more about SSH

Why I put these things together ?

- our concept
- our basics for security
- our goal: Kerberos, AFS and SSH working together

Kerberos Authentication with shared secret

- to authentice users and services on an unsafe network
- The kerberos server is trusted by all entities on the network, users, hosts and services, so called principals.
- All principals share a secret password or key with the kerberos server.
- This enables principals to verify that the messages from the kerberos server are authentic.
- Each Kerberos installation defines an administrative realm of control.

Kerberos TGT, Ticket and Token

- a user first talks to the Authentication Service on the Key Distribution Center (KDC) to get a Ticket Granting Ticket (TGT).
- When the user wants to talk to a Kerberized service, he uses the TGT to talk to the Ticket Granting Service (on KDC)
- TGS verifies the user's identity using the TGT and issues a ticket for the desired service (ssh,imap,AFS,acrontab,...)
- the AFS token is a special Kerberos 4 service ticket

Kerberos Time and Clocks

- system clocks have to be synchronized
  timestamps play an important rule during authentication

- expiration of TGT and Tickets for security reasons
  lifetime 25 hours

Kerberos Krb5 is better than Krb4

- The key salt algorithm has been changed to use the entire principal name.
- The network protocol has been completely redone (based on ASN.1) (Abstract Syntax Notation)
- support for forwardable, renewable and postdatable tickets
- Kerberos tickets can now contain multiple IP addresses and addresses for different types of networking protocols.
- A generic crypto interface module is now used, so other encryption algorithms beside DES can be used.
- More flexible cross-realm authentication.
- Pre-authentication

  to overcome the ability to do an offline dictionary attack, one weakness of kerberos4

Kerberos kinit and klog

kinit -h	shows all options
kinit -R	renew TGT for further 25 hours, AFS token too max renew life time: 30 days Attention: renewed ticket no longer forwardable
kinit -l time	new TGT with specified lifetime, like 1h
kinit	initializes the credentials cache if you have had tickets or tokens of other realms or cells they are destroyed after kinit
klog	is obsolete, but can still be used to get an AFS token for another cell: klog user@cell

Kerberos klist and kdestroy

klist shows your tickets and tokens

Format of Kerberos 5 principals:
krbtgt/IFH.DE@IFH.DE
host/pippin.ifh.de@IFH.DE
Format of Kerberos 4 principals:
krbtgt.IFH.DE@IFH.DE
imap.manto@IFH.DE
Format of AFS Token:
afs@IFH.DE

kdestroy destroys tickets and tokens

AFS short history

AFS Andrew Filesystem is distributed network file system

In the 80s developed at CMU; research project with IBM.
Later spun off into Transarc labs, absorbed by IBM;
sold AFS as commercial product.
September 2000 ➡ announcement of OpenAFS
Since then several releases of OpenAFS.

Benefits of AFS

- Global filespace
- good security with strong authentication
- data replication (readonly)
- data relocation (readwrite, transparent to clients)
- online backups (clone volumes) enable users to recover from rm accidents without admin intervention

Why we don't like to have NFS

- very weak security
  everybody should know: no truly private information in NFS
  criminal behaviour can cause data loss
- data relocation is very expensive, needs maintenance time
  and a lot of communication with users
- a crash or shutdown of an NFS server can cause a lot of
  hanging clients maybe in the future: NFS4 better

AFS ACLs

AFS knows more than traditional UNIX filesystem permissions:

read, write, insert,delete

lookup, lock, administrate

but only per directory

(one disadvantage of AFS)

group and other UNIX bits are ignored, but

AFS "respects" the owner UNIX bits:

Only read a file if its UNIX 'r' owner mode bit is 'r'

Only write to a file if its UNIX 'r' and 'w' owner mode bits are not '-'

Only execute a file if its UNIX 'r' and 'x' owner mode bits are not '-'

The owner of the directory always has administrate rights.

Without further ACLs the owner bits are valid for ALL users.

AFS ACLs cont.

ACLs are lists of pairs: (*who mode*)

who can be a user, a group or a group of IP addresses

mode is a list of bits like rwid

ACLs Examples

system:administrator rlidwka	(afs admin)
system:anyuser l	(really world accessable, lookup)
desy-hosts rl	(141.34.0.0,131.169.0.0)
webserver rl	
group:amanda rlidwka	(member of group amanda)

to list ACLs

fs listacls *path*

AFS ACLs cont.

to change ACLs

fs setacl *path who mode [who mode ... who mode]*

shortcuts:

read	rl
write	rlidwk
all	rlidwka
none	-

fs sa ~/project group:nuastro read

readable for group group:nuastro

you should not give others write access to your home directory tree

better done in group space

RSR statement:

http://www.desy.de/rsr/intern/rsr-stat-1999-06.html

AFS ACLs cont.

Inheritance of ACLs

subdirectories (not mountpoints) inherit the acls of its parents
at creation time

Changing ACLs in a directory tree

find *dir* -type d -noleaf -exec fs setacl { } group:nuastro rl \;
find *dir* -type d -noleaf -exec fs copyacl *dir1* { } \;

ACLs not file based, what to do?

Example: file .forward, .procmailrc

```
fs la ~/public
      desy-hosts rl
      wwinzig rlidwka
ls -l .forward :     .forward      ➞   public/.forward
ls -l .procmailrc:  .procmailrc   ➞   public/.procmail
```

AFS Token

to get an AFS token

- log in with a password
- log in via ssh from an other host of our Kerberos domain
  - on which you have a valid Kerberos5 TGT
  - (ticket forwarded,new token generated from ticket);
  - but you do NOT get an AFS token using ssh key authentication
- unlock the screen (xlock, xscreensaver)
- run kinit

to check AFS tokens

- run klist, tokens

AFS Cache

the client maintains a local cache for performance reasons

- persistent (still available after reboot)
- readwrite
- local changes to a file are flushed to the server after closing it
- after crash of PC or hard RESET the cache can be corrupt and you need a system admin
- in case of cache problems sometimes helps: fs checkvol

AFS Volumes and Mountpoints

- AFS space is handled in chunks called volumes
- each volume has an associated quota
- a volume need not be mounted
- a volume can have more than one mountpoint
- a volume can have readonly replicas
- a volume can have a backup volume (snapshot) generated last night

fs listquota *dir*

➞ you get the quota but also the volume name

lsmount

display the mount point(s) for a given *vol_name* or *path*
(generated from nightly cron jobs)

example: lsmount -t /afs/ifh.de/group/pitz

AFS Volumes and Mountpoints Cont.

Naming scheme

/afs/ifh.de/user
/afs/ifh.de/group
/afs/ifh.de/www
/afs/ifh.de/@sys/products

home directory

is one volume
one is the snapshot from last night : ~/.OldFiles
more are possible for scratch or www or user defined

AFS Backup

http://www-zeuthen.desy.de/computing/services/AFS/backup.html

- Nightly snapshot available, but not mounted;
  to mount the snapshot:
  - find the afs volume the lost data belongs to:
    fs listquota /afs/.ifh.de/group/mygroup/myproject
    volume name g.mygroup.vol10
  - check the creation time of the backup volume:
    vos exa g.mygroup.vol10
  - mount the volume:
    (you need admin rights (ACL:a))
    fs mkmount /afs/.ifh.de/user/o/otto/mygroup g.mygroup.vol10.backup
  - remove the mountpoint after you copied the lost files:
    fs rmmount /afs/.ifh.de/user/o/otto/mygroup

AFS Backup Cont.

- most volumes with a quota less than 2 Gbyte are in the backup (incremental) and you can recover with afs_recover
  - afs_recover -date *YYYYMMDD*
    you get a recovered volume mounted in ~/Recover
  - copy the wanted files
  - unmount and delete the recover volume with
    afs_remove

AFS Quota

each volume has a quota
list quota with
fs listquota *path*

home directory: max quota: 500 Mbyte
should stay below 90% not to run in trouble

Now we can handle quota for projects with afs_admin.
AFS group admins can manage the AFS group space by themselves.

Manipulating Protection Groups

AFS pricipals (hosts, user and groups) can be members of protection groups.
All unix groups have corresponding PTS groups with the same members.
All PTS groups are owned by the princical group.

- to list the member a group of users

  pts membership group:*group*

- create your own group with your friends and give them special rights

  pts creategroup wwinzig:myfriends
  pts adduser friend wwinzig:myfriends
  pts membership wwinzig:myfriends
  pts rename wwinzig:myfriends wwinzig:proj_a
  pts delete wwinzig:proj_a
  fs setacl ~/project_A wwinzig:proj_a rl

AFS sysname

binary type, a per-host property
DESY Linux 5: i586_linux24
Solaris 8: sun4x_58
may be a list as well (SLD3: i586_rhel30 i586_linux24 i386_linux24)

fs sysname reports the CPU/operating system type
livesys

a path component with @sys is automatically resolved to the binary type of the host you are working on.

Example PITZ group using that: ls -l /afs/ifh.de/group/pitz/doocs/

used to make binaries and libraries on all platforms with the same path available
can only be used in AFS space

AFS additional

AFS and Batchsystem SGEEE (Sun Grid Engine Enterprise Edition)

for job submission you need a valid kerberos 5 ticket

AFS and Cron

special acrontab

http://www-zeuthen.desy.de/computing/services/AFS/acrontab.html

Further Notes:

no man pages, but

fs help; fs listacl -help

vos help; vos exa -help

pts help; pts exa -help

SSH program and protocol

The Secure Shell protocol provides four basic security benefits:

- user Authentication
- host Authentication
- data Encryption
- data Integrity

Implementations of Secure Shell offer the following capabilities:

- a secure command-shell
- secure file transfer
- secure X11 forwarding
- remote access to a variety of TCP/IP applications via a secure tunnel

SSH protocol 1 and 2

SSH Version 1, developed in 1995, is being phased out to replace the non-secure UNIX "r-commands" (rlogin, rsh, and rcp).
In favor of SSH Version 2, 2001 standardized by the IETF's (Internet Engineering Task Force) Secure Shell Working Group, SSH1 is deprecated.

SSH2 has taken its place, but why ?

There are proven cryptographic weaknesses in the protocol:

ssh Insertion Attack (1998)

ssh Compensation Attack (2001/2002)

A lot of security problems in SSH1 during the last years.

SSH SSH2 is better than SSH1

SSH2 is a complete rewrite of the protocol

- separate transport, authentication and connection protocol
- implemantation of Diffie-Hellmann key exchange method -
  the star of SSH2

  Diffie-Hellman is a key agreement protocol, and was developed by Diffie and Hellman in 1976. The entire purpose of Diffie-Hellman is to allow two entities to exchange a secret over a public medium without having anything shared beforehand.
- strong cryptographic integrity check
- modular cryptographic and compression algorithms
- no longer server keys needed
- host based authentication not dependend on the network address

SSH OpenSSH

OpenSSH supports both SSH1 and SSH2

Why haven't we changed to SSH2 earlier?

- no Kerberos ticket forwarding;
  what means no single sign on.
- no Krb4 support
- no AFS support

Now things have changed and we started switching to protocol 2.

SSH Examples for ssh and scp

```
ssh     myname@desthost       [command]
scp     [host1:]file1         [host2:]file2
scp  -r     local/work        athena.mit.edu:/path/to/remote
scp     athena.mit.edu:/path/from/remote/'*.c'          local
scp     myname@athema.mit.edu:/path/from/remote   local
```

xssh desthost

(script adapted by DESY Zeuthen)

to start an ssh in a separate window

SSH public key authentication

- to generate authentication keys for ssh protocol 2 run
  ssh-keygen -t dsa

- add the public dsa key to $HOME/.ssh/authorized_keys

ATTENTION:
With public key authentication you do NOT get an AFS token!
If you use X11 forwarding you will run into timeouts during login into DESY hosts and have no secure X11 forwarding.

You can use this for passwordless login or copying data into your notebook or home PC.

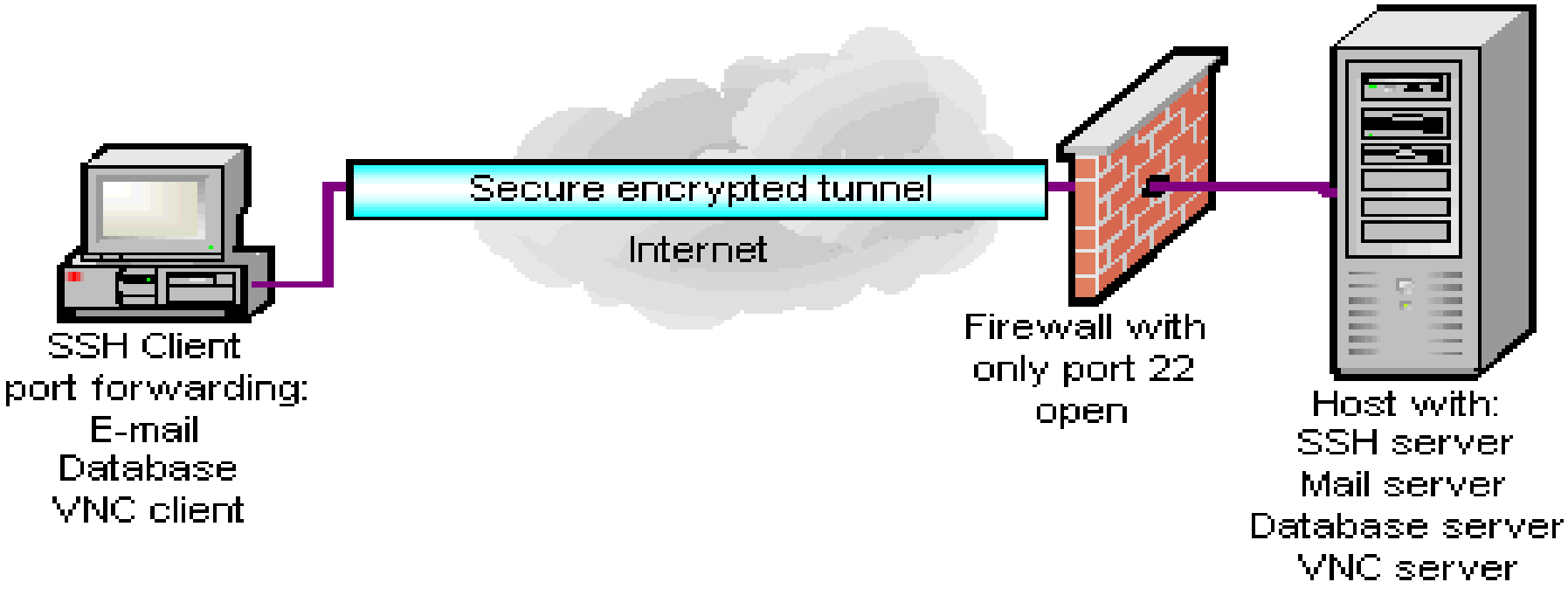
mindterm - Java applet implementing SSH and a terminal emulator

https://bastion.desy.de

If you accept the certificate of the DESY-CA and use the signed applet you can connect directly to systems at DESY Zeuthen.
Overtype the initial string bastion.desy.de with e.g. pub.ifh.de.

Secure Port Forwarding (Tunneling)

Port forwarding allows data from normally unsecured TCP/IP applications to be secured



Secure Port Forwarding (Tunneling) Cont.

ssh has built-in support for intelligent X11 forwarding

If you are running some X clients on the remote system the ssh server is forwarding them to the ssh client which is forwarding them to the X server through the encrypted tunnel.

X11 forwarding switched on by default at DESY.

X11client (remote) → ssh server (remote)

→ ssh client (local) → Xserver (local)

Secure port forwarding with "ssh -L"

You can forward arbitrary connections through your ssh tunnel using the -L option.

This makes your ssh client listen on a given port and forward traffic received there through the tunnel;

it instructs the remote sshd to send the traffic to a given IP address and port.

ssh -L *port*:localhost:*destport desthost* ...

ssh -L *port:desthost:destport remotehost* ...

SSH Secure Port Forwarding (Tunneling) Cont.

Example 1: direct connection to your desktop from outside

```
ssh -l myuserid -L 7777:mydesk.ifh.de:22 pub.ifh.de cat -
```

In a separate window you can use now ssh or scp to connect directly to your internal host through the tunnel. For example:

```
ssh -p 7777 localhost uname -a
scp -p -P 7777 localhost:data/file1.txt .
```

Example 2: access to DESY internal webpages

```
ssh -L 7777:webserver:80 pub.ifh.de cat -
```

In another window you can use the tunnel

```
ssh -p 7777 localhost
```

and start mozilla with http://localhost:7777/

Example 1: VNC (Virtual Network Computing) with ssh tunnel

- start on the destination host vncserver
  vncserver
- replace in ~/.vnc/xstartup twm by startkde or fvmw2 or whatever you want
- build an ssh tunnel to the destination host with
  port = 5900 + Display
  ssh -L *port*:localhost:*port* *dest*
- now you can start vncviewer with the given display on your local host
  vncviewer localhost:*display*
- please run vncserver -kill *display* when done
- AFS/Krb credentials are those of the vncserver process

SSH Further applications

- rsync –rsh="ssh"
- protecting mysql sessions
- securing cvs by pserver port forwarding

SSH Troubleshooting

? hanging clients?

~? list of supported escape sequences

~. terminate connection

~^Z suspend ssh

~# list forwarded connections and more ...

? is the kerberos ticket expired?

? are you running into quota? Then .xauth can´t be written.

? warnings about wrong ssh keys? danger: man-in-the-middle-attack.
X11 forwarding switched off. If destination host key changed
you have to remove the old key from your ~/.ssh/known_hosts.

? no idea? debugging with ssh -vvv, send the output to uco@ifh.de

Thank you for your attention and long patience

Further questions ?