Building a KDC

Kerberos Implementations

- RedHat 5 comes with MIT Kerberos 1.6
- Ubuntu 10.04 LTS comes with MIT Kerberos 1.8.1
- Admin through CLI, but from any remote machine ("kadmin" protocol runs over TCP)
  - e.g. add / remove / modify principals
- Heimdal
    - developed outside the US when exporting crypto from the US was illegal
- AD

KDC Security

- A compromise of KDC destroys the whole realm
- Run it on a "dedicated" server
  - Can probably live with LDAP too (non-root)
- Firewall off everything except Kerberos ports
    - kerberos: udp 88
    - kadmin: tcp 749
    - kpasswd: udp 464
- Almost no-one should have local shell logins; use kadmin (remotely) to administer database

KDC Security (cont)

- Enable "pre-authentication" for all users
  - In initial TGT exchange, user required to send hash of their password (already default in Ubuntu)
  - Makes it harder to do dictionary attacks
- Disable Kerberos 4 (removed in MIT 1.7)
- allow_weak_crypto=false (default from MIT 1.7)
    - may need to keep arcfour (RC4) cipher if interoperating with older Windows servers
- Keep service keytabs and host keytabs separate and protected via filesystem permissions!

Realms

```
primary/instance@REALM
```

Realms

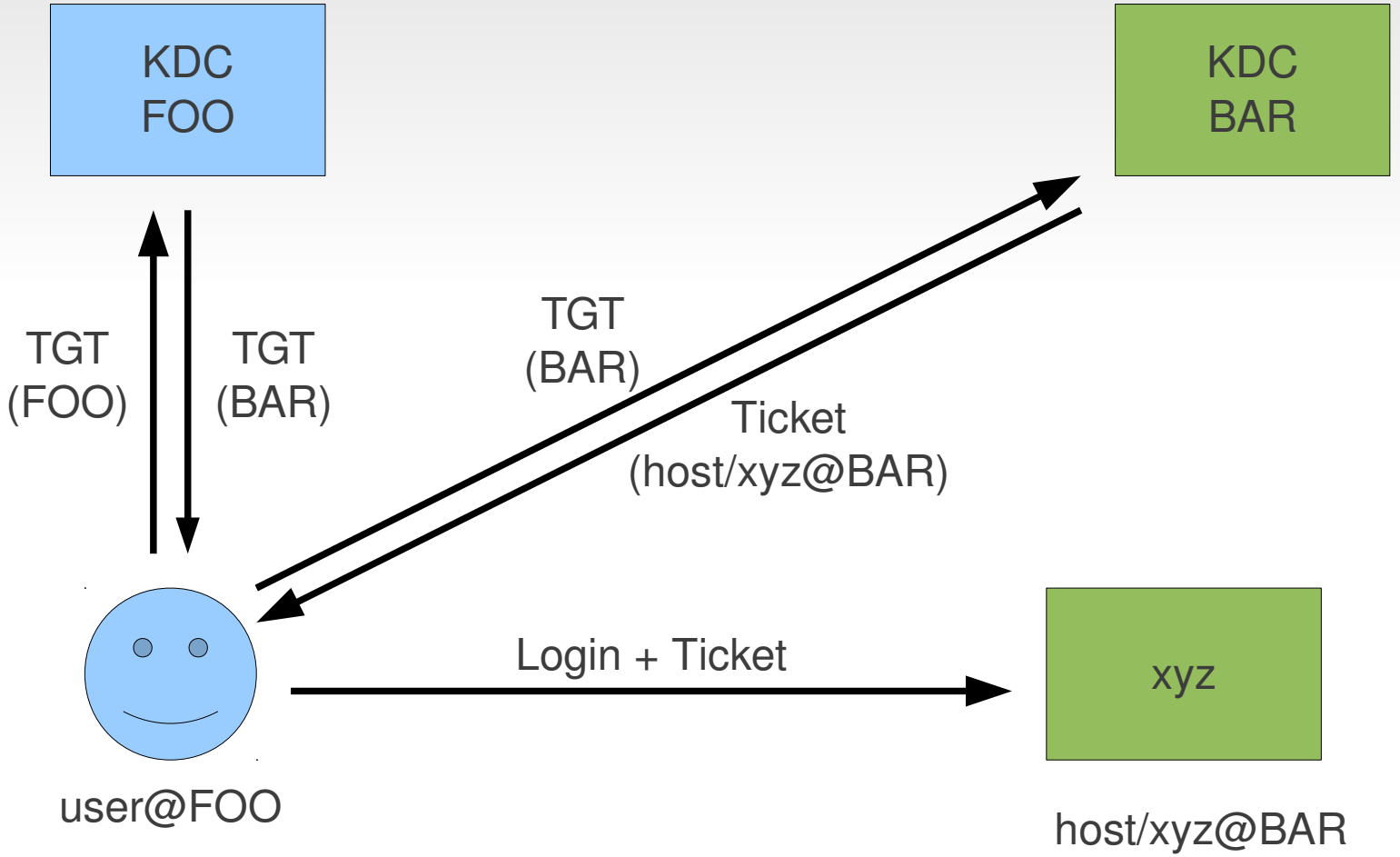
- A realm is a collection of principals which trust the same KDC
- Conventionally in UPPER.CASE to distinguish from DNS domains
- A principal in one realm can get a ticket to prove their identity to a principal in another realm: this is "cross-realm authentication"
- To do this, the KDC in the first realm must share a key with the KDC in the second realm (*)

(*) Multi-hop through chain of KDCs also possible

Cross-realm authentication

principal	key
user@FOO	XXXXX
krbtgt/BAR@FOO	ZZZZZ

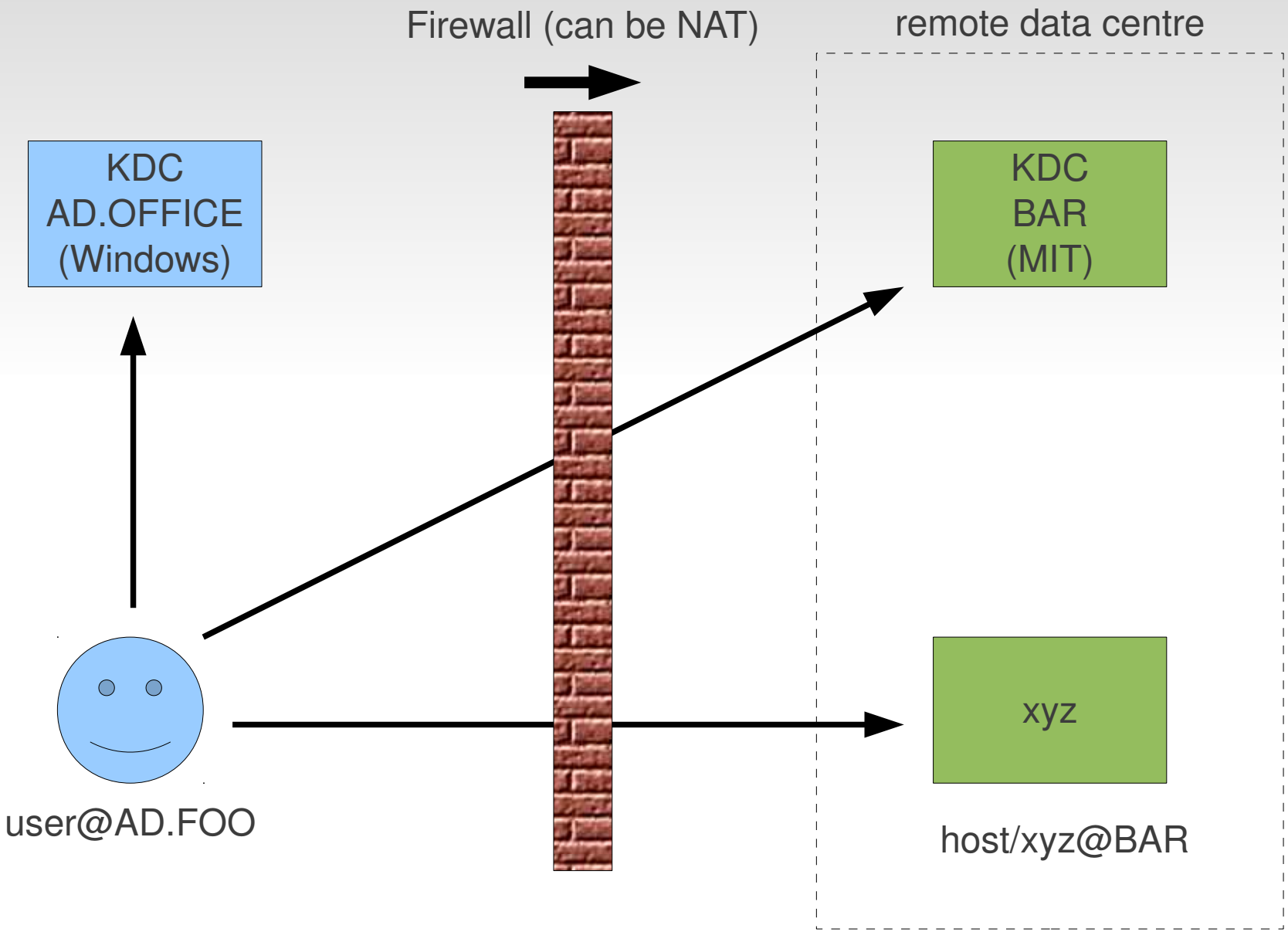
principal	key
host/xyz@BAR	YYYYY
krbtgt/BAR@FOO	ZZZZZ



Notes on cross-realm auth

- All communication is from client to KDCs and from client to service
- Hence it works if the remote KDC and server are outside of NAT

Cross-realm with firewalls



Mixing office AD with MIT?

- Keep your Windows domain (AD.OFFICE)
- Remote data centre has its own realm, running MIT Kerberos under Unix
- Hosts share keys with their local DC's realm (and hence are independent of AD)
- Set up cross-realm authentication
- Your password is only stored in the office AD, and not in the data centre!

Notes

- Windows users who have logged into AD don't even need to kinit! (Kerberized putty - untested)
- Unix users:
  - kinit user [or kinit user@AD.OFFICE]
  - or use pam_krb5 on your workstation to get tickets when you login
- Data centre still needs its own LDAP server
- ssh fallback to password auth will only work for users which exist in the DC's KDC
  - Equally, you can have datacentre-only users not in AD

Realm mapping

- We want user foo@AD.OFFICE to login as system user "foo" on machines in some different realm
- Bulk rule to avoid creating .k5login entries for every user
- Configured in /etc/krb5.conf on each server

```
[realms]
 WS.NSRC.ORG = {
  auth_to_local = RULE:[1:$1@$0](^.*@AD\.OFFICE$)s/@AD.OFFICE$//
  auth_to_local = DEFAULT
 }
```

Exercise

- Put your machine in its own realm, e.g. pc1.ws.nsrc.org is in REALM1.WS.NSRC.ORG
- Build your own KDC
- kinit to your own KDC
- (Spare time exercises for cross-realm auth)

LDAP

LDAP

- "Lightweight Directory Access Protocol"
- General rule: any protocol with "lightweight" or "simple" in its name, isn't :-(
- The protocol and the data model are standardised
- Use for passwd/group information is informal, but widely implemented
  - RFC2307 - "experimental"
  - RFC2307bis - still only a draft (expired)

LDAP Data model

- Records are called entries, containing attributes
- Entries identified by unique Distinguished Name
- Permitted attributes from objectClass and schema

```
dn: uid=ldapuser,ou=People,dc=ws,dc=nsrc,dc=org
objectClass: account
objectClass: posixAccount
cn: ldapuser
uid: ldapuser
uidNumber: 10004
gidNumber: 100
homeDirectory: /home/ldapuser
loginShell: /bin/bash
```

LDAP protocol

- A packed binary (ASN.1) protocol over TCP
- Supports TLS and SASL
- Small set of operations: bind (authenticate), search, add, modify, delete, compare, modifyDN
- Command-line tools e.g. ldapsearch, ldapadd, ldapmodify ... entries in text format (LDIF)
- Search options
    - baseDN, scope, filter, attrs

OpenLDAP + Kerberos

- A bit tricky but doable
- Beware some of the HOWTOs on the net
  - server config is now stored within LDAP itself
  - "man slapd-config" for full info
  - old HOWTOs may show you slapd.conf instead
  - older Ubuntu created a default database
- The handouts show how we built the class server
- Care mapping Kerberos ticket realm to auth DN (documentation is wrong: ITS#6757)

Exercise

- Build a Kerberized LDAP server

Managing users?

- Adding/removing users via ldapadd/ldapdelete is a bit painful
- Write your own scripts, or use someone else's
  - e.g. "ldapscripts" package - needs patching for GSSAPI
- Lots of GUI projects out there. If you find a good one, let us know
- Simple user management interface is probably the main advantage of Active Directory if you have it

You may also come across...

- Using LDAP for Authentication (pam_ldap)
  - User sends password to server; server checks it using an LDAP bind operation
  - Passwords are exposed repeatedly
  - Needs TLS and certificates for security
  - Critical security of ACLs, e.g. allow users to change their own password but not see other people's!
- Using LDAP as a Kerberos data store
  - Might simplify replication, but IMO the overall complexity is unlikely to be worth it

Other projects

- FreeIPA under development, worth a look
- Integrates Fedora OS, Kerberos MIT, Red Hat/Fedora Directory server, DNS server, Certificate Authority, ...
- Easier to manage than just the individual components? You decide.