Final Presentation

Project Moonshot

Bart Stokman, Rikkert ten Klooster, Bas Heuft, Killian Hesterman

Content

- Introduction to Moonshot
- Job description
- Task division
- LDAP server
- RADIUS server
- Pluggable Authentication Module
- Name Service Switch
- Secure Shell
- WebDAV

Introduction to Moonshot

- Single unifying technology
  - extending the benefits of federated identity
    - Cloud infrastructures
    - High Performance Computing
    - Grid infrastructures
  - other commonly deployed services
    - mail
    - file store
    - remote access
    - instant messaging

Goal of Moonshot

- Technology to enable the management of access to a broad range of services and applications
  - single technology
  - single infrastructure
  - single sign-on
    - internal services
    - external services.
- Enhance the user's experience
- Reduce costs for those organisations supporting users, and delivering services to them.

Job description

- Find out what Moonshot does and how it works
  - Possibilities of Moonshot
  - Which applications are supported
- Implement Moonshot Authentication & Authorization
  - OpenSSH
  - WebDav

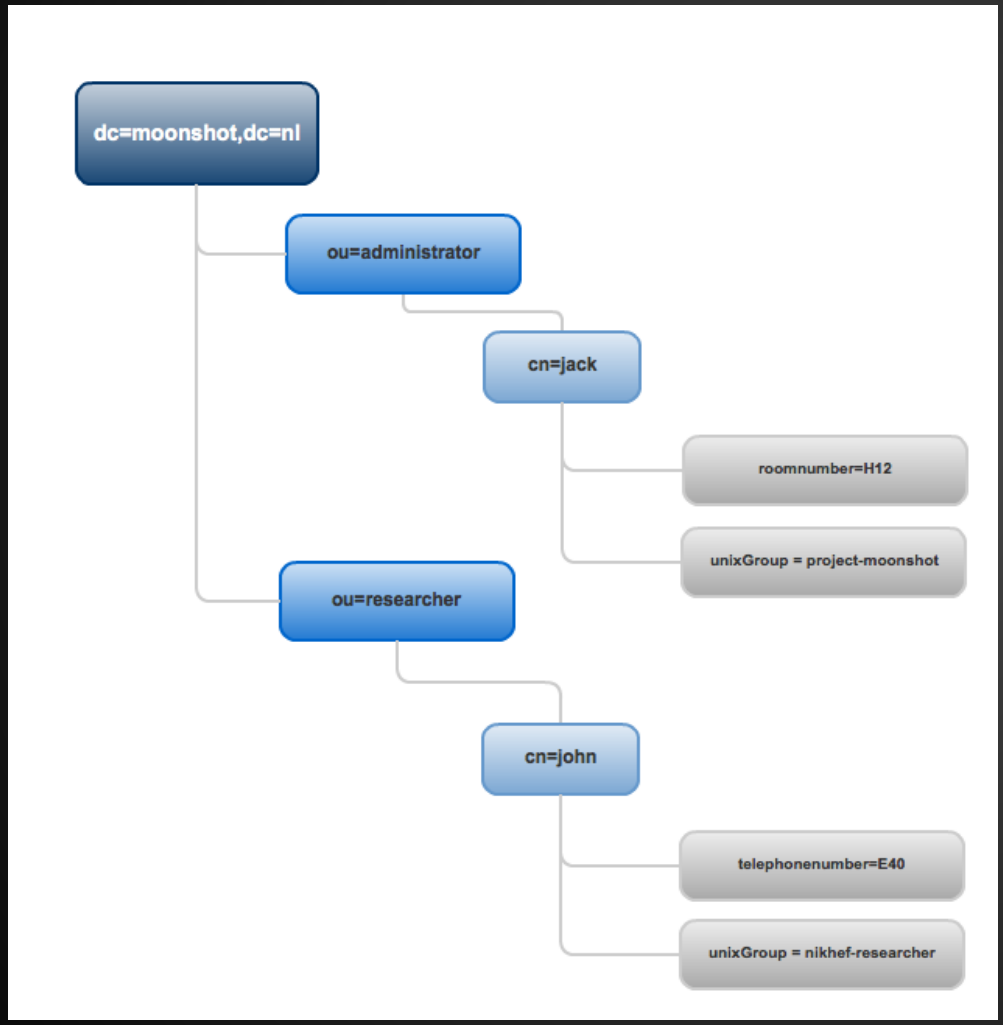
Task Division

- Infrastructure
  - OpenLDAP
  - FreeRADIUS
  - PAM
  - NSS
- Two applications
  - OpenSSH
    - Bas Heuft
    - Killian Hesterman
  - WebDav
    - Rikkert ten Klooster
    - Bart Stokman

LDAP

- What is LDAP
  - Lightweight Directory Access Protocol
- Wherefore is LDAP used?
  - Storing of user account data
- Why LDAP?
  - Lightweight
  - Support from RADIUS
  - Hierarchical structure
- LDAP & Moonshot
  - EduPerson schema
  - Unix group stored in attribute

LDAP hierarchy



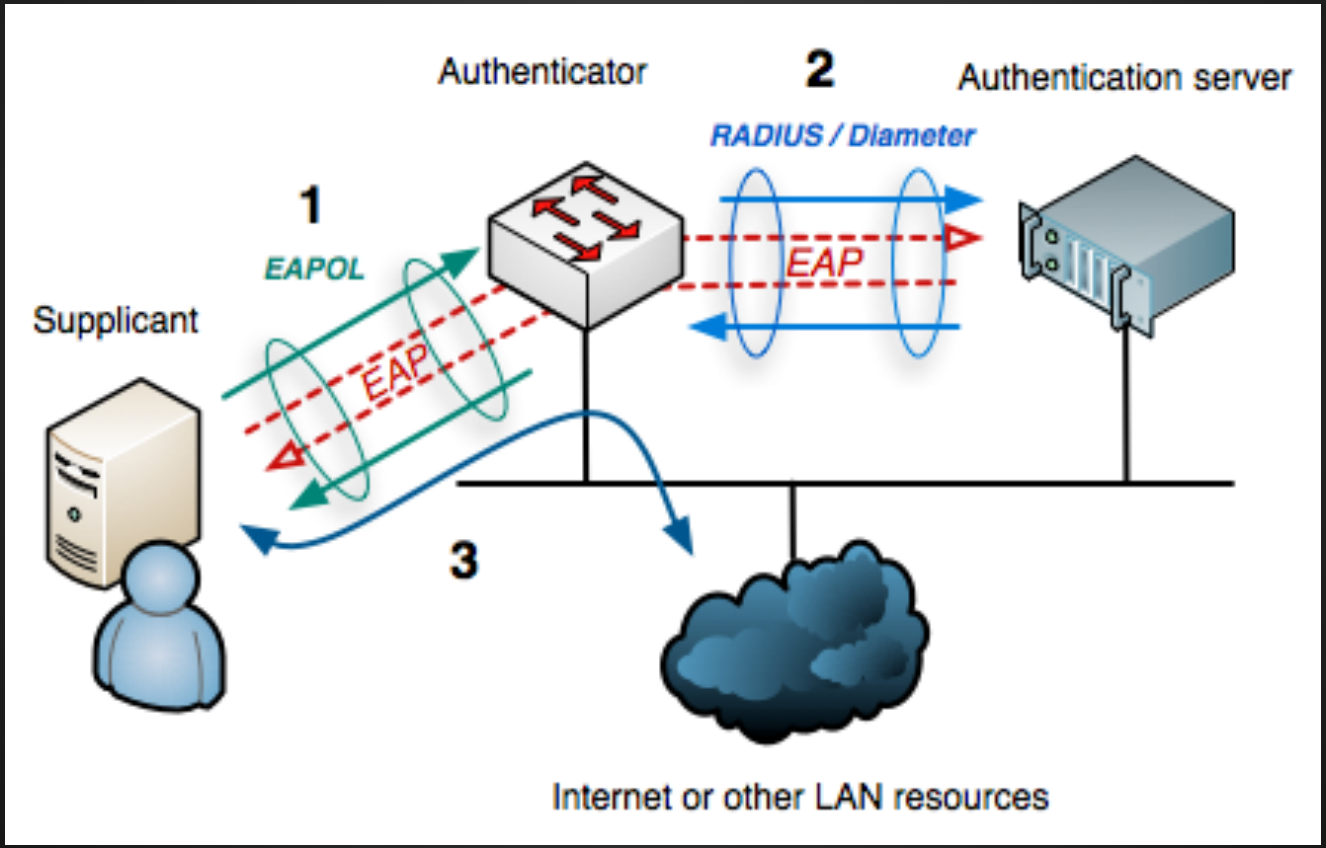
RADIUS server

- What is RADIUS?
- Wherefore is RADIUS used?
- FreeRADIUS configuration

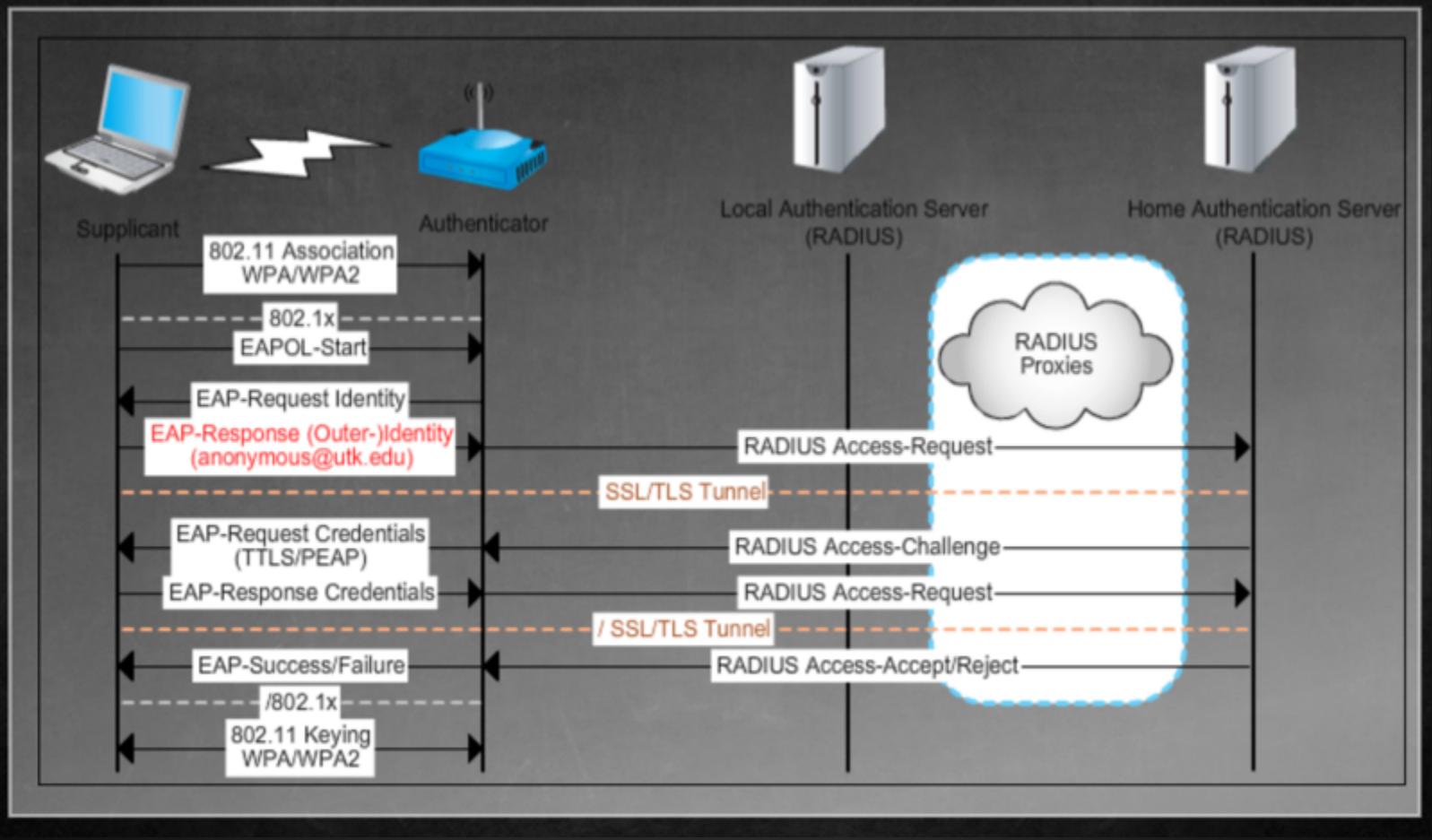
What is RADIUS?

- Authentication Server
  - Authentication
  - Authorization
  - Accounting
- Realms / Federation
- Security
  - 802.1x
  - EAP-TTLS

802.1x

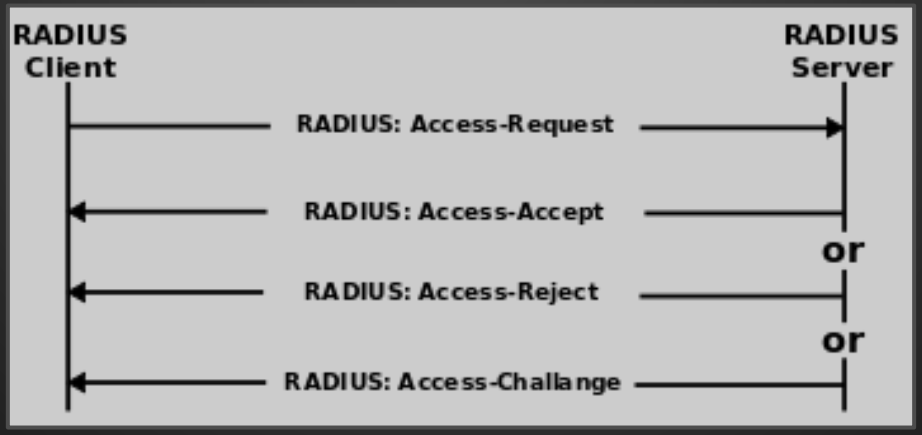

Authenticator
2
Authentication server
RADIUS / Diameter
1
EAPOL
EAP
EAP
Supplicant
3
Internet or other LAN resources

Eduroam setting



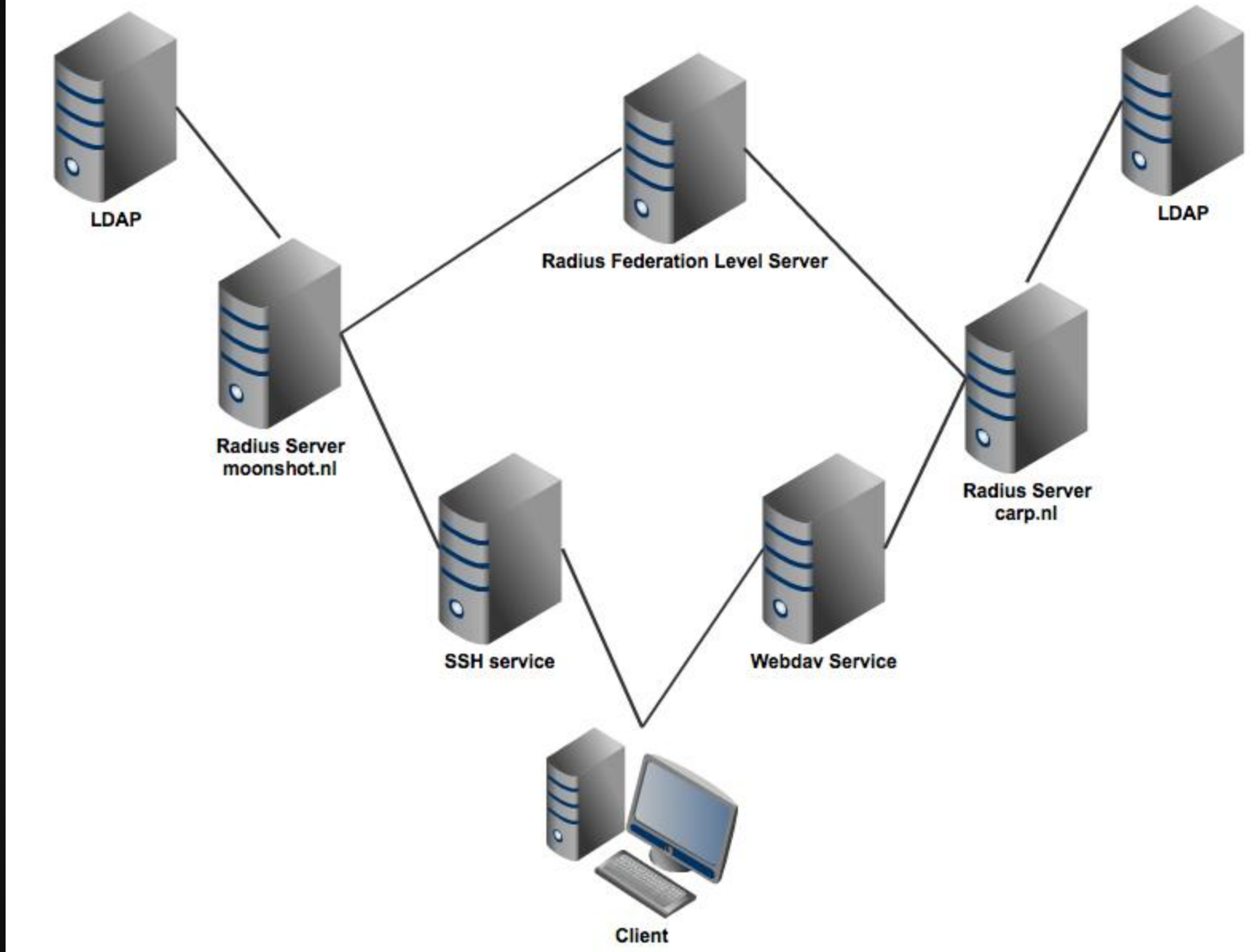
Wherefore is RADIUS used?

- Connecting realms
- Checking credentials in LDAP
- Passing through LDAP attributes objects



FreeRADIUS configuration

- LDAP authentication
- Attribute mapping
  - checkItem Cleartext-Password userPassword
  - replyItem Unix-User unixUser
- Add Clients
  - client localradtest{ ipaddr = 10.198.5.37 secret = testing123 reguire_message_authenticator = no nastype = other }
- Add Realms
  - *realm carp.nl{ type = radius authhost = LOCAL accthost = LOCAL secret = testing123 nostrip }*


LDAP
Radius Federation Level Server
LDAP
Radius Server
moonshot.nl
Radius Server
carp.nl
SSH service
Webdav Service
Client

Pluggable Authentication Module (PAM)

- What is PAM?
- Why Moonshot with PAM?
- Possibilities Moonshot with PAM
- Control options for PAM
- PAM stacks
- Security Considerations

What is PAM?

- Authentication, establish the user who they claim to be
- Account, provide account verification types of service
- Password, update authentication mechanisms
- Session, provides both opening and closing hook for modules to affect the service

Why Moonshot with PAM?

- Provides an authentication scheme that can be used by many applications
- Allows system administrators and developers to control over authentication
- Allows developers to develop applications without creating their own authentication mechanism
- Well documented

Possibilities Moonshot with PAM

- Single Sign-on for users with ‘username’@’ domain’ for multiple services (SSH, WebDAV, NFS).
- Authorization based on the user’s LDAP-attributes.
  - Map users to (temporary) pool-accounts.
  - Map users to (temporary) Unix-groups.
  - Manage data storage per user.
  - Manage connection speed per user.
  - etc.

Control options for PAM

- Requisite, upon failure the authentication process will be terminated
- Required, the module is required for the process to succeed
- Sufficient, the module is sufficient for the process to succeed
- Optional, if the module fails PAM will go further
- Include, the included PAM stack will be executed

Default PAM stack OpenSSH daemon CentOS 6.2

```
#%PAM-1.0
auth       required     pam_sepermit.so
auth       include      password-auth
account    required     pam_nologin.so
account    include      password-auth
password   include      password-auth

# pam_selinux.so close should be the first session rule
session    required     pam_selinux.so close
session    required     pam_loginuid.so

# pam_selinux.so open should only be followed by sessions to be executed in the user context
session    required     pam_selinux.so open env_params
session    optional     pam_keyinit.so force revoke
session    include      password-auth
```

Our PAM Stack

```
auth        required    pam_radius_auth.so debug try_first_pass
account     required    pam_radius_group.so
account     required    pam_radius_auth.so
# delete group-mapping done by pam_radius_group.so
session     required    pam_radius_removegroup.so
session     required    pam_radius_auth.so debug
# delete group-mapping done by pam_radius_group.so
session     required    pam_loginuid.so
# pam_selinux.so open should only be followed by sessions to be executed in the user context
session     required    pam_selinux.so open env_params
session     optional    pam_keyinit.so force revoke
```

Security considerations

- Credentials as plain text to the RADIUS
  - Man-in-the-middle attack
- System administrator
  - Federation trust

- Possible security measurements
  - SSL
  - IPSEC connection

Name Service Switch (NSS)

- What is NSS?
- Why use NSS in Moonshot?

What is NSS?

- NSS stands for Name Service Switch
- NSS provides sources for common configuration databases and name resolution
- Our configuration in /etc/nsswitch.conf
  - passwd: files moonshot
  - shadow: files
  - group: files

Why use NSS in Moonshot?

- NSS can be used to map user accounts to ‘pool-accounts’.
- User@domain can be mapped to pool001
- "@" are not allowed Unix-usernames

Secure Shell (SSH)

- OpenSSH & Moonshot
- Specific configurations for OpenSSH
- Incorrect password problem

OpenSSH & Moonshot

- Open-source
- Default on a lot of operating systems
- Successfully tested with GSSAPI and moonshot enabled OpenSSH by Janet
- Same benefits with PAM and NSS

Specific configurations for OpenSSH

- sshd_config
  - PasswordAuthentication yes
  - ChallengeResponseAuthentication yes
  - UsePam yes

Problem password incorrect (1/3)

- rad_recv: Access-Request packet from host 10.198.5.35 port 32768, id=70, length=120
  - User-Name = "rikkert@moonshot.nl"
  - User-Password = "\010\n\r\177INCORRECT

Problem password incorrect (2/3)

- [ldap] returns ok
- [pap] login attempt with password "?INCORRECT″
- [pap] Using clear text password "suiker"
- [pap] Passwords don't match

auth.h (line 49-75):

```
struct Authctxt {
        int             valid;          /* user exists and is allowed to login */
        struct passwd   *pw;            /* set if 'valid' */
}
```

auth1.c (line 401-407):

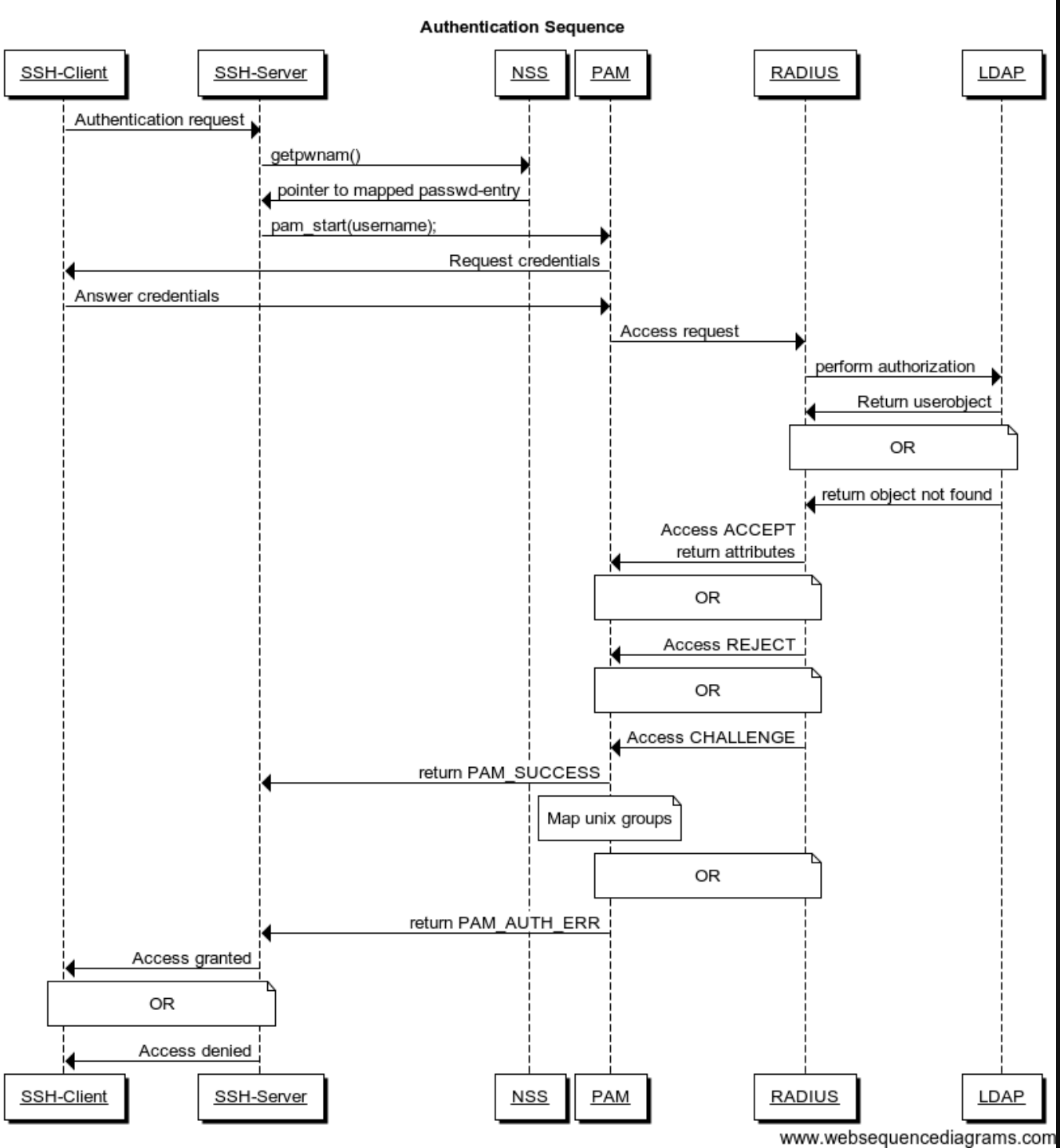
```
/* Verify that the user is a valid user. */
        if ((authctxt->pw = PRIVSEP(getpwnamallow(user))) != NULL)
                authctxt->valid = 1;
        else {
                debug("do_authentication: invalid user %s", user);
                authctxt->pw = fakepw();
        }
```

auth_pam.c(r. 234, 834-839):

```
static char badpw[] = "\b\n\r\177INCORRECT";
if (sshpam_authctxt->valid &&
        (sshpam_authctxt->pw->pw_uid != 0 ||
        options.permit_root_login == PERMIT_YES))
                buffer_put_cstring(&buffer, *resp);
        else
                buffer_put_cstring(&buffer, badpw);
```

Solution password incorrect

Update OpenSSH_5.3p1 to OpenSSH_5.8p2


Authentication Sequence
SSH-Client
SSH-Server
NSS
PAM
RADIUS
LDAP
Authentication request
getpwnam()
pointer to mapped passwd-entry
pam_start(username);
Request credentials
Answer credentials
Access request
perform authorization
Return userobject
OR
return object not found
Access ACCEPT
return attributes
OR
Access REJECT
OR
Access CHALLENGE
return PAM_SUCCESS
Map unix groups
OR
return PAM_AUTH_ERR
Access granted
OR
Access denied
SSH-Client
SSH-Server
NSS
PAM
RADIUS
LDAP
www.websequencediagrams.com

What is webdav?

- File collaboration over HTTP
- Client-Server
- Clients for:
  - Windows
  - Linux
  - OS X

Clients support mounting drives

- Maintenance of properties, NameSpace management, Collections, Overwrite protection

How did we authenticate?

- authnz_external_module
  - External authentication over pwauth
- pwauth
  - Authentication over PAM support
- authz_unixgroup_module
  - Authorization based unix groups

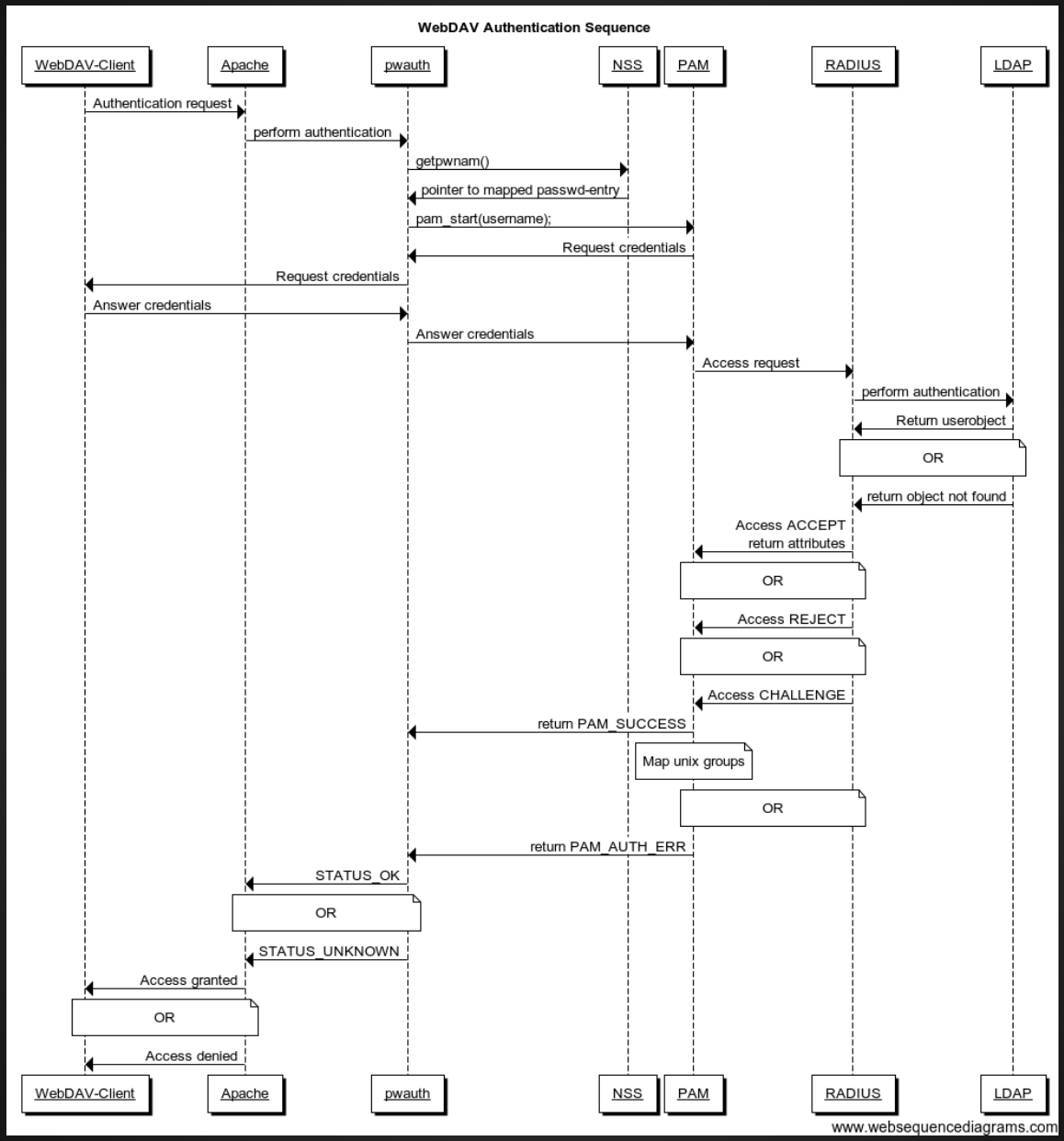
Authentication

PAM stack
#%PAM-1.0.1

auth	required	pam_radius_auth.so debug try_first_pass
account	required	pam_radius_group.so
account	required	pam_radius_auth.so
session	required	pam_radius_removegroup.so

Example Config file:

```
require valid-user
<Limit DELETE>
        require user admin
</Limit>
```


WebDAV Authentication Sequence
WebDAV-Client
Apache
pwauth
NSS
PAM
RADIUS
LDAP
Authentication request
perform authentication
getpwnam()
pointer to mapped passwd-entry
pam_start(username);
Request credentials
Request credentials
Answer credentials
Answer credentials
Access request
perform authentication
Return userobject
OR
return object not found
Access ACCEPT
return attributes
OR
Access REJECT
OR
Access CHALLENGE
return PAM_SUCCESS
Map unix groups
OR
return PAM_AUTH_ERR
STATUS_OK
OR
STATUS_UNKNOWN
Access granted
OR
Access denied
WebDAV-Client
Apache
pwauth
NSS
PAM
RADIUS
LDAP
www.websequencediagrams.com

Authorization

redirect to user folder
RewriteRule ^$ /%{REMOTE_USER}/ [R=301,L]

Create links to folders to which the user is authorized.

Authorization based on unix user groups:
.htaccess of a folder
require group project-moonshot

Problems

Quota Problem

- Apache patch (Fixed)

Cadaver rewrite mod

- Use of other clients (No fix for Cadaver)

(13) Permission denied

- Disable httpd_t in SElinux (Fixed)

Future research

- GSS
  - Broader support of SSH servers / clients
  - Ability to plug in authorization modules
  - Broaden the support of services with end to end security
- Webdav redirect support
  - Mainly client support
- PAM module expansion
  - Multiple attributes
  - Enhanced security
- XACML support
  - Define authentication / authorization policies

Summary

- Authorization made possible with PAM.
- Not ready for federated environments due to security issues

Demonstration

- WebDAV
- SSH

John
unix-group project-moonshot
password: moonshot

Jack
unix-group: nikhef-researcher
password: moonshot