Æ-DIR

„Paranoid user management with OpenLDAP“

at Chemnitzer Linuxtage 2015

Who?

- Michael Ströder <michael@stroeder.com>
- Freelancer
- Focus on
  - Directory services (LDAP etc.), identity management
  - X.509-based PKI, encryption, digital signature
- Open source projects as developer
  - web2ldap
  - python-ldap

Why? (1)

- Why another LDAP user management?
- Infrastructure gets more complex
  => different security requirements
- Administrative roles are mixed/relaxed (DevOps)
  => fine-grained authorization also for system administration needed
- Audit trail
  - find out who did what
  - need persistent IDs for all entitites!
  - never ever re-use IDs!

Why? (2)

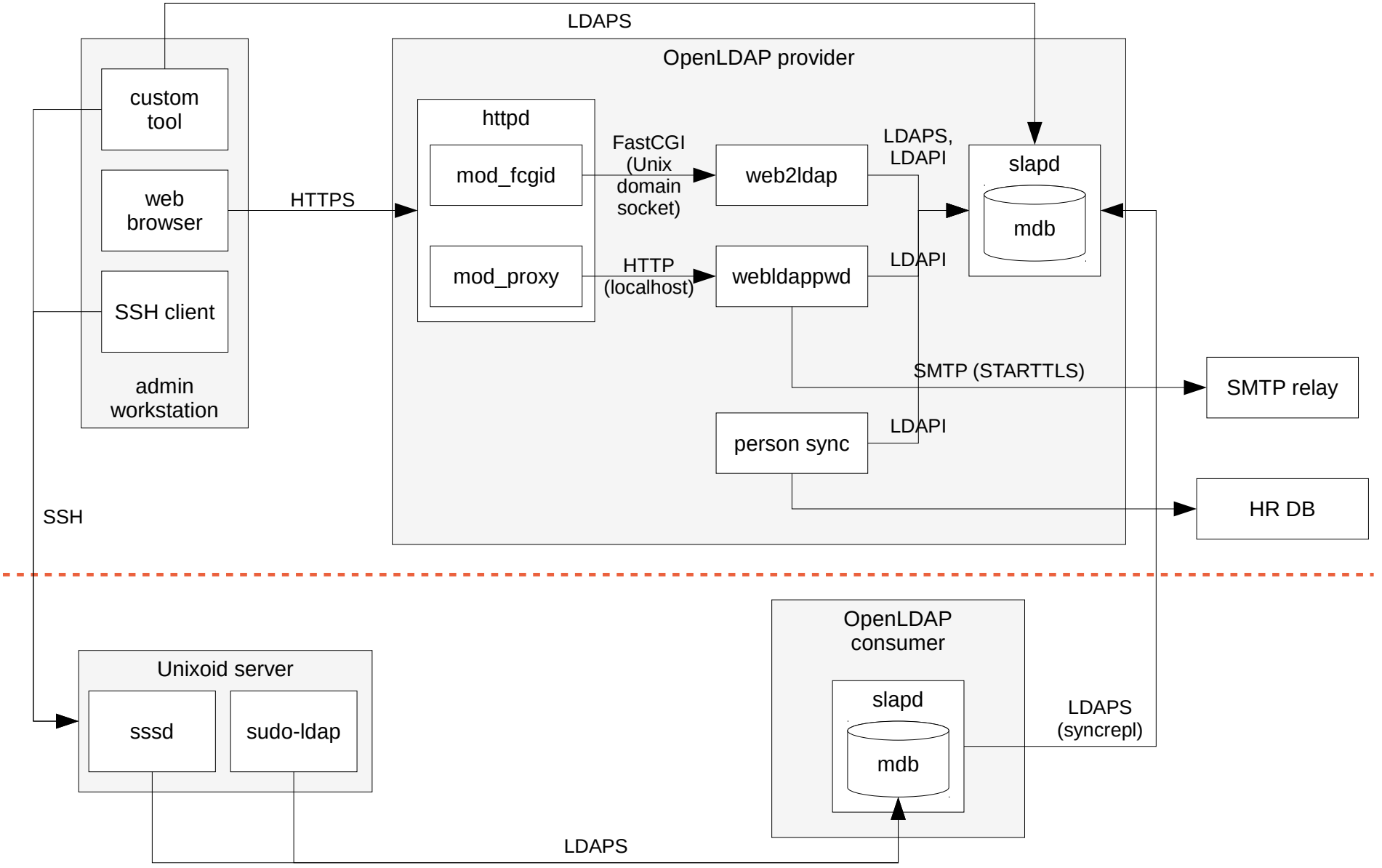
- Follow need-to-know principle!
- Systems must not know information only needed by other systems
- => Fine-grained authorization of servers/services to users/groups/sudoers etc.
- => Individal authentication of servers/services necessary for providing “views” by ACLs
- AFAIK no such LDAP-based solution available

  => Æ - **A**uthorized **E**ntities

Components: Overview

- OpenLDAP
- web2ldap with templates & plugins
- Simple web application for password self-service
- Special admin tools (mostly command-line)
  - bulk initialization of servers
  - reporting
- LDAPS / StartTLS everywhere - no exception!
- *sssd* and *sudo-ldap* currently used as client components, other software possible

Components: Architecture



Components: OpenLDAP (1)

- OpenLDAP 2.4.39+ with back-mdb
- No *rootpw*!
- *authz-regexp* for SASL/EXTERNAL (clients certs and *LDAPI)*
- Heavy use of regex- & set-based ACLs and constraints (slow)
- Overlays used:
  - accesslog, lastbind
  - constraint, refint, unique, memberof
  - ppolicy, rwm, noopsrch

Components: OpenLDAP (2)

- Two-tier replication
- SASL/EXTERNAL with TLS server certificates
- Providers with multi-master replication (MMR):
  - for data maintenance
  - access only for human admins
  - no access for servers and services
- Read-only consumers
  - Provide user, group and sudoers entries to servers and services
  - no write access/chaining => password change not possible from servers

Components: Provider tools

- Various tools running on provider:
  - HR data synchronisation job
  - Password self-service web application
  - Group update job
- LDAPI with SASL/EXTERNAL and *authz-regexp* mapping separate user accounts to LDAP authz-DNs
  => no clear-text passwords needed in configuration!

Components: web2ldap

- web2ldap 1.2.x with customization
- LDIF and HTML templates
- Plugin classes
  - display values with additional information
  - normalize and validate values
  - select lists (mostly 1:1 relationship to URI constraints)
  - Generating *uid*, *uidNumber* and *gidNumber*
- Supplemental schema for DIT structure rules and name forms (because not supported in OpenLDAP 2.4.x)
- Personal authorization of user
  => no privilege escalation possible

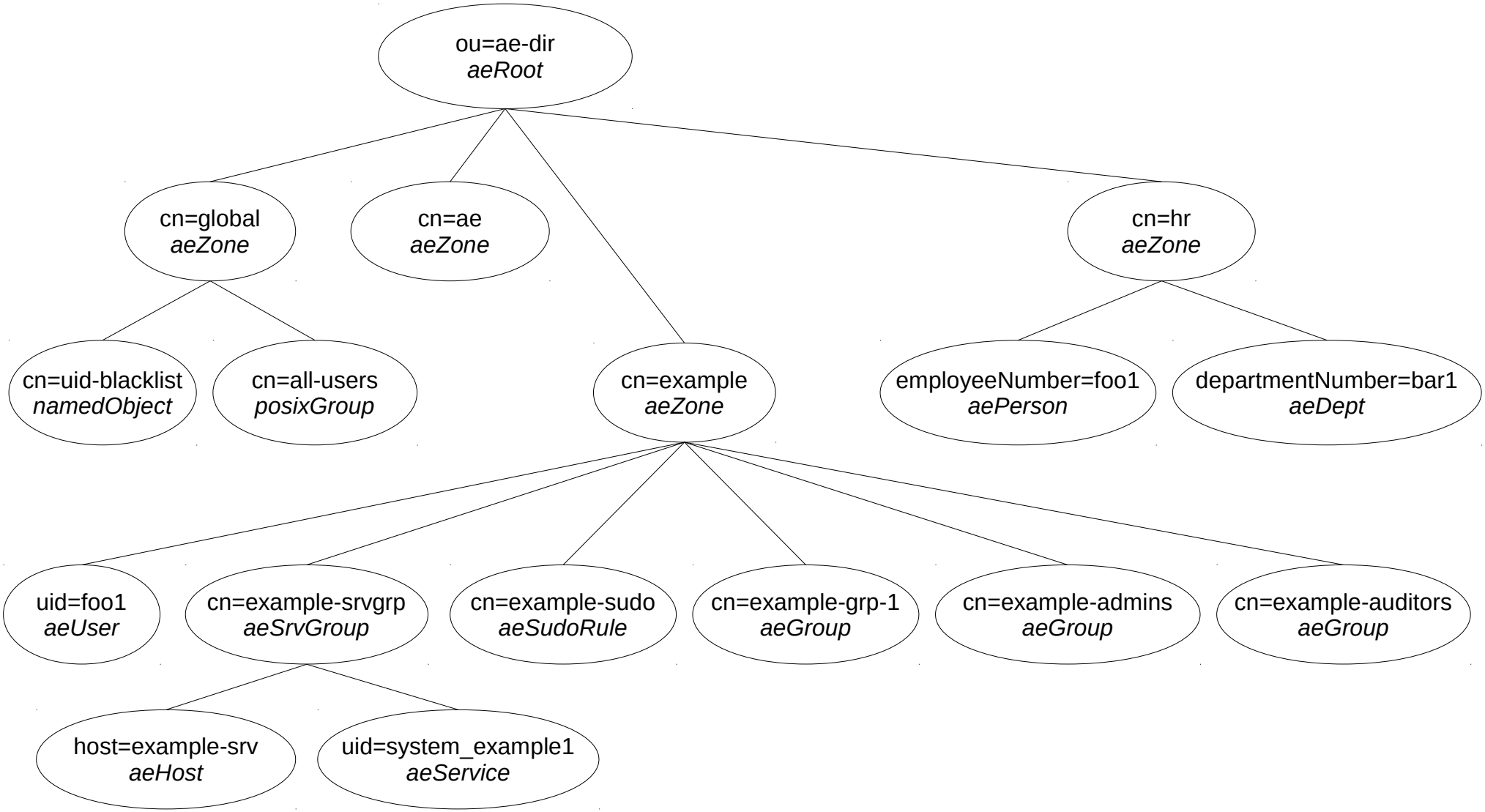
Roles

- Æ admins
  - Can *manage* everything within ou=ae-dir
  - Can *read* cn=monitor and cn=config
- Æ auditors: Can *read* everything within ou=ae-dir
- Zone admins: Can *write* anything within a zone
- Zone auditors: Can *read* anything within a zone
- Setup admins: Can *write* aeHost/aeService
- Users: Can *read* own entries, other members of own groups, change own password, etc.
- Anonymous/guest access is disallowed!

Schema: Basics

- Strong requirement:
  Compability to existing NIS-LDAP and sudoers schema
- Common meta data in abstract class *aeObject*
- For all structural object classes:
  - DIT content rules
  - DIT structure rules & name forms

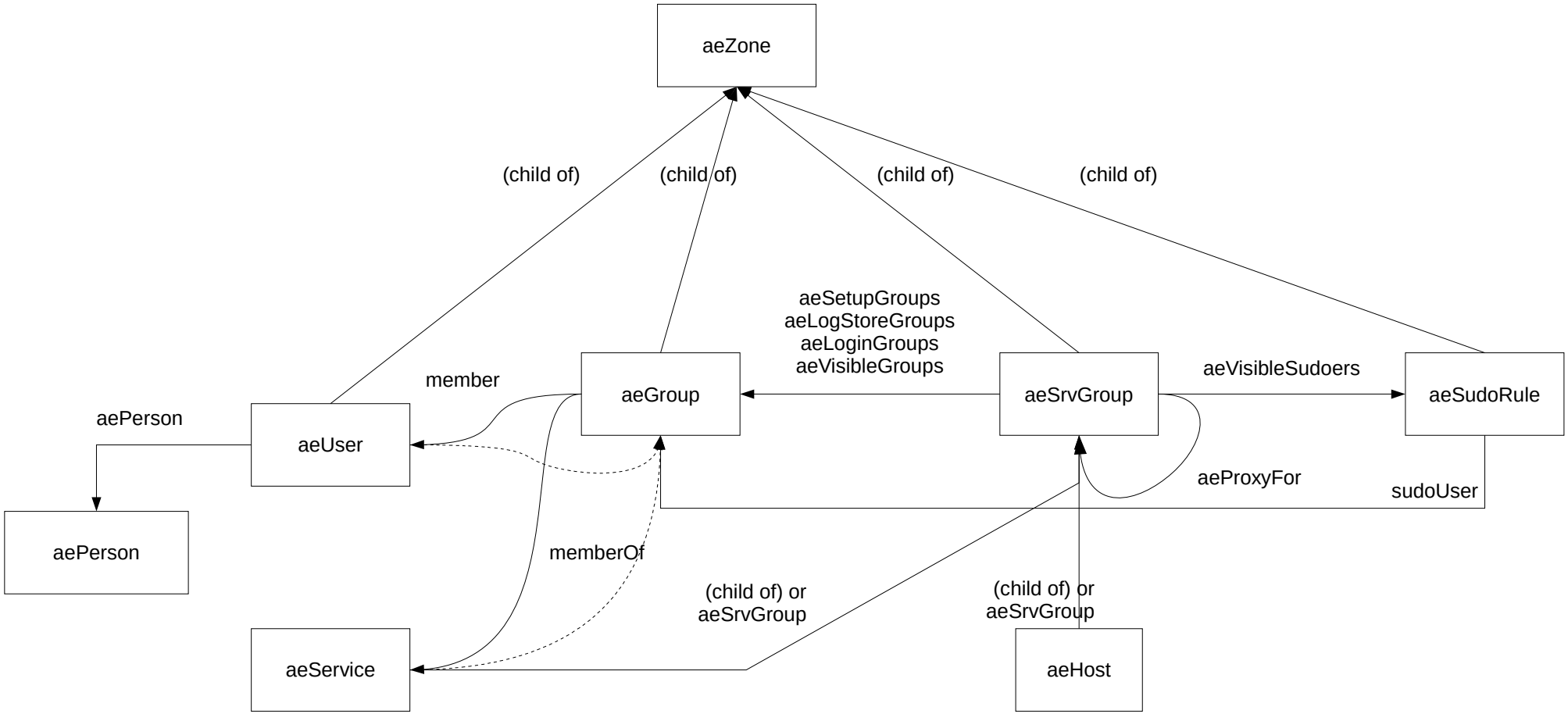
Directory Information Tree (DIT)



Reference Attributes

- The entity relationship is evaluated by ACLs to determine access rights of bound entity
- References between entries:
  - most times by DN
  - sometimes by tree structure (zones and server/service groups)
  - in one case by prefix name (*sudoUser* → *aeGroup*)
- Cross-zone references most times allowed (except *aeProxyFor*)

Entity Relationships



Schema: aeObject

- Abstract object class for meta data used as common base class for all structural object classes:
  - *aeStatus*
    active (0), deactivated (1), archived (2), requested (3)
  - *description*
    Descriptive text for entries is helpful afterwards!
  - *aeNotAfter* and ae*NotBefore*
    Used to limit usage period (not usable in ACLs though)
  - *aeTicketId*
    Sure you have a tracker application, don't you?

Schema: aeZone

- Simple container for delegated administration
- Characteristic attribute for RDN: *cn*
- Default role groups in zone *foo*: *foo-admins* (zone admins) and *foo-auditors* (zone auditors)
- Special zones:
  - cn=people: for *aePerson* entries (HR data)
  - cn=global: UID blacklist, global primary *posixGroup*, global sudoers default, etc.
  - cn=ae: For maintaining Æ directory itself, e.g. role groups for Æ *admins* and Æ *auditors*

Schema: aePerson

- *aePerson* entries should be synchronized from HR
- Based on *inetOrgPerson* and *msPerson*
- Typically one person entry per *active* employee, but be prepared for strange data coming from HR!
- Attribute *mail* is mandatory for password self-service in this customer deployment
- Possible characteristic attributes for RDN: *employeeNumber* or *uniqueIdentifier*
- Attribute *uid* disallowed to avoid clash with user entries!

Schema: aeUser (1)

- Characteristic attribute for RDN: *uid*
- One or more *aeUser* entries reference a single *aePerson* entry => n:1 mapping
- Immutable attributes, never change/re-use values:
  - *aePerson*
  - *uid*
  - *uidNumber*
- Primary group in *gidNumber* is constrained to one possible value in existing *posixGroup* entry!
- Never use a local group IDs in *gidNumber*!

Schema: aeUser (2)

- *uid* is not derived from person's name!
- Associated DIT content rule allows AUX classes:
  - *posixAccount* (RFC2037)
  - *ldapPublicKey* for SSH authorized keys
  - *msPwdResetObject* for password reset self-service
  - (to be extended..Kerberos etc.)

Schema: aeService

- Tool user, service user, machine user, whatever you call it…
- Characteristic attribute for RDN: *uid*
- Associated DIT content rule allows AUX classes:
  - *posixAccount* (RFC2037)
  - *ldapPublicKey* for SSH authorized keys
- Two different use-cases:
  - Member of user group (*aeGroup*) similar like *aeUser*
  - Member of service group(s) (*aeSrvGroup*):
    Retrieves user and group entries, but no login

Schema: aeGroup

- Characteristic attribute for RDN: *cn*
- Derived from:
  - *groupOfEntries*
    Attribute *member* used optionally, empty group possible
  - *posixGroup (classic RFC 2307)*
    allows to satisfy also legacy clients
  - *groupOfURLs*
    For provisiong groups based on LDAP searches defined in attribute *memberURL* (use with care!)
- Overlay *slapo-memberOf* sets back-link to groups in attribute *memberOf* of member entries

Schema: aeSrvGroup

- Server/service group
- Characteristic attribute for RDN: *cn*
- References to *aeGroup* entries for several rights and visibility
  - aeSetupGroups → Role “Setup admin”
  - aeLogStoreGroups
  - aeLoginGroups → access to *sshPublicKey*
  - aeVisibleGroups (e.g. NFS user groups)
- *aeVisibleSudoers* references visible *aeSudoRule* entries

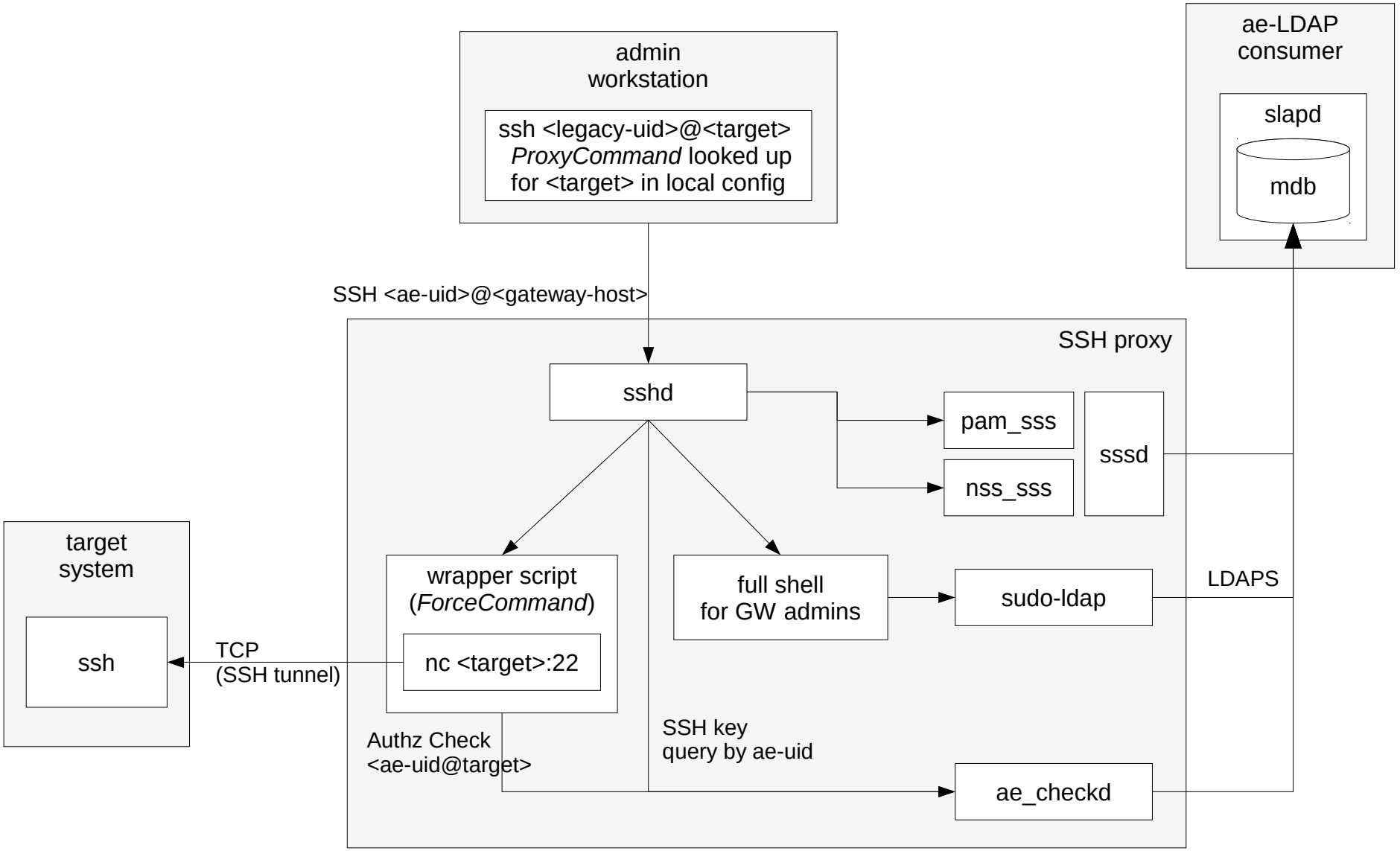
Schema: aeHost

- Each server has to authenticate to get authorized
- Characteristic attribute for RDN: *host*
- Membership in server group by
  - being subordinate entry of *aeSrvGroup* entry
  - reference attribute *aeSrvGroup*

Schema: aeSudoRule

- For SUDO rules instead of /etc/sudoers
- Derived from *sudoRole* object class (*sudo-ldap* schema)
- Restrictions added
  - *sudoUser* only reference user groups!
  - *sudoHost* disabled because OpenLDAP-ACLs will do it
- *sudo-ldap* always querys for each command
- *sssd* 1.9.x+ can also cache sudoers entries
- maybe sync rules into /etc/sudoers.d/ locally

SSH relay


admin
workstation
ssh <legacy-uid>@<target>
ProxyCommand looked up
for <target> in local config
ae-LDAP
consumer
slapd
mdb
SSH <ae-uid>@<gateway-host>
SSH proxy
sshd
pam_sss
nss_sss
sssd
target
system
ssh
TCP
(SSH tunnel)
wrapper script
(*ForceCommand*)
nc <target>:22
full shell
for GW admins
sudo-ldap
LDAPS
Authz Check
<ae-uid@target>
SSH key
query by ae-uid
ae_checkd

Conclusion

- ACLs in OpenLDAP backend is additional boundary against privilege escalation in frontends
- (Set-based) ACLs are
  - complex => regression testing script
  - a performance hog => currently more hardware used
- You eventually need a fallback login if all fails
- It's hard to not open security holes afterwards
- Upcoming ideas should always have a real use-case and fit into role model!
- ACL changes require regression testing!

Ideas: Performance tuning

- Æ aware client configuration tools e.g. tuning sssd.conf by using specific filters
- Rewriting filters for different identities (authz-DNs) based on OpenLDAP's *slapo-rwm*
- Replace set-based ACLs by custom *dynacl* module:
  - hopefully faster
  - evaluate *aeNotAfter* and *aeNotBefore*
  - skilled C programmers needed

Ideas: More integration

- DHCP/DNS/PXE/TFTP machine setup integration:
  Find out more about existing schema mess before
- MIT Kerberos (multiple realms)
- OTPwith shared secrets directly in user entries
- Samba (multiple domains)
- automount maps
- Configuration management:
  Tie *Puppet* node declaration or *ansible* playbook to *aeSrgvGroup* and/or *aeHost*

To do: Even more

- Æ schema spec as Internet draft (experimental)
- Implement ae_*demon*
  - Lean and nearly-zero-conf NSS/PAM demon
  - knows DIT and schema => optimized searches
  - boot-strap support
  - SASL/EXTERNAL with TLS clients certs (e.g. puppet certs)
- Implement *ae-dir-ui*
- Implementation with *OpenDJ*:
  Are ACIs powerful enough?