Æ-DIR

„Paranoid user management with OpenLDAP“

at LDAPcon 15

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)

- Infrastructure gets more complex
  - Many systems
  - Different security requirements
- Mixed/relaxed administrative roles (DevOps)
  - Admins for production environment
  - Developers
  - Management / Auditors
- Audit trail (who did what)
  - need persistent IDs for all entitites!
  - never ever re-use IDs!

Why? (2)

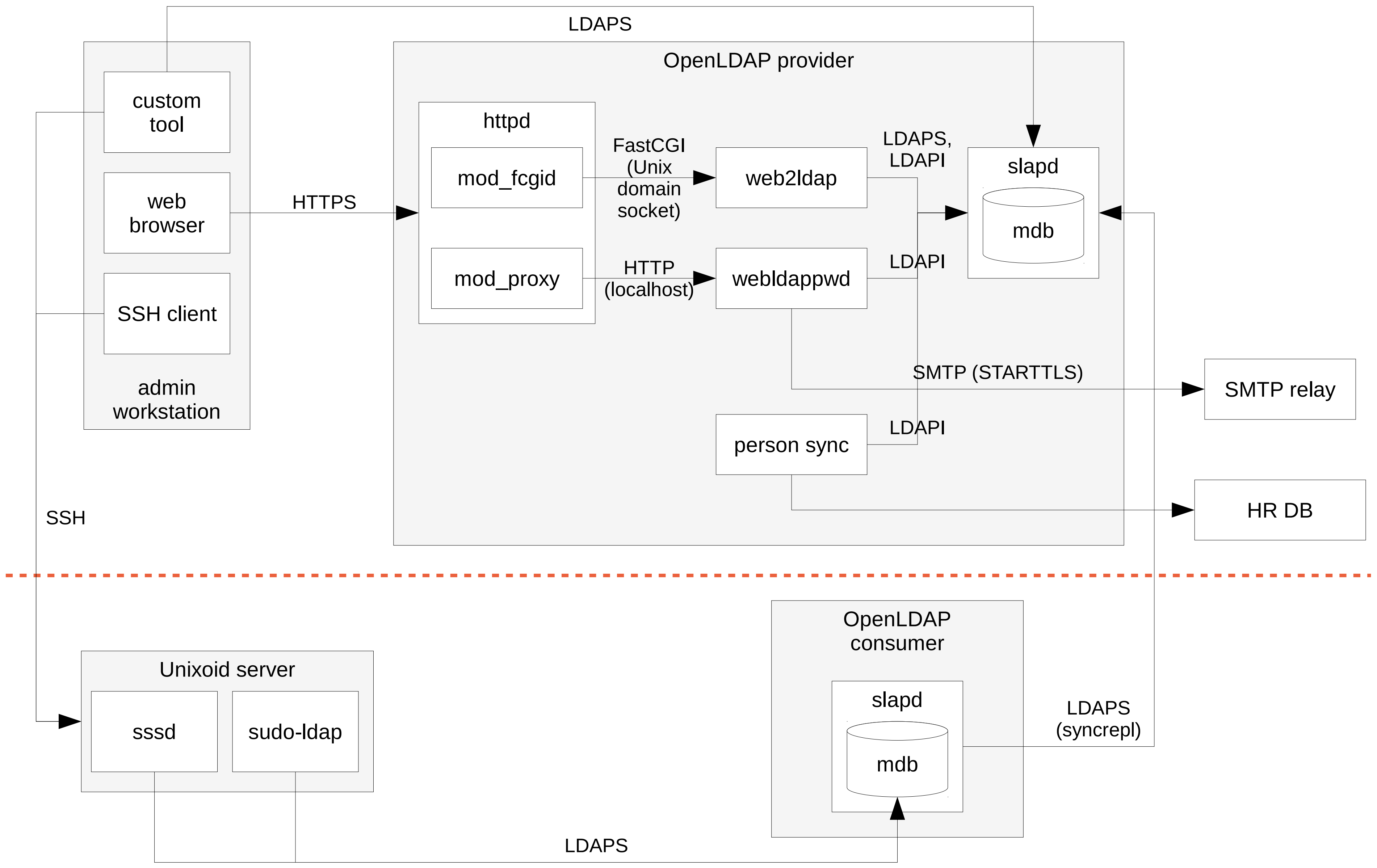
- Strictly follow need-to-know principle!
  => Fine-grained authorization of servers/services to users/groups/sudoers etc.
  => Individual authentication of servers/services
  => Provide “views” by ACLs

- AFAIK no such LDAP-based solution available
  => Æ-DIR - Authorized Entities Directory

Components: Overview

- OpenLDAP
- web2ldap with HTML/LDIF 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*!
- Avoid system passwords: *authz-regexp* for SASL/EXTERNAL (clients certs and *LDAPI*)
- Heavy use of regex- & set-based ACLs/constraints
- Overlays used:
  - accesslog, lastbind
  - constraint, refint, unique, memberof
  - ppolicy, rwm, noopsrch

Components: OpenLDAP (2)

- Two-tier replication
- Providers with multi-master replication (MMR):
  - for data maintenance
  - access only for human admins
  - no access for servers and services
- <u>Read-only</u> consumers
  - Provide user, group and sudoers entries to servers and services
  - no write access/chaining
    => passwd/PAM not possible from normal servers

Components: Provider tools

- Various tools locally running on provider:
  - HR data synchronisation job
  - Password self-service web application
  - Group update job
- LDAPI with SASL/EXTERNAL
- *authz-regexp* maps local POSIX user accounts to LDAP authz-DNs
  => no clear-text passwords 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*
- Authorization only in slapd
  => no privilege escalation
- Supplemental schema for DIT structure rules and name forms (not directly in OpenLDAP 2.4.x)

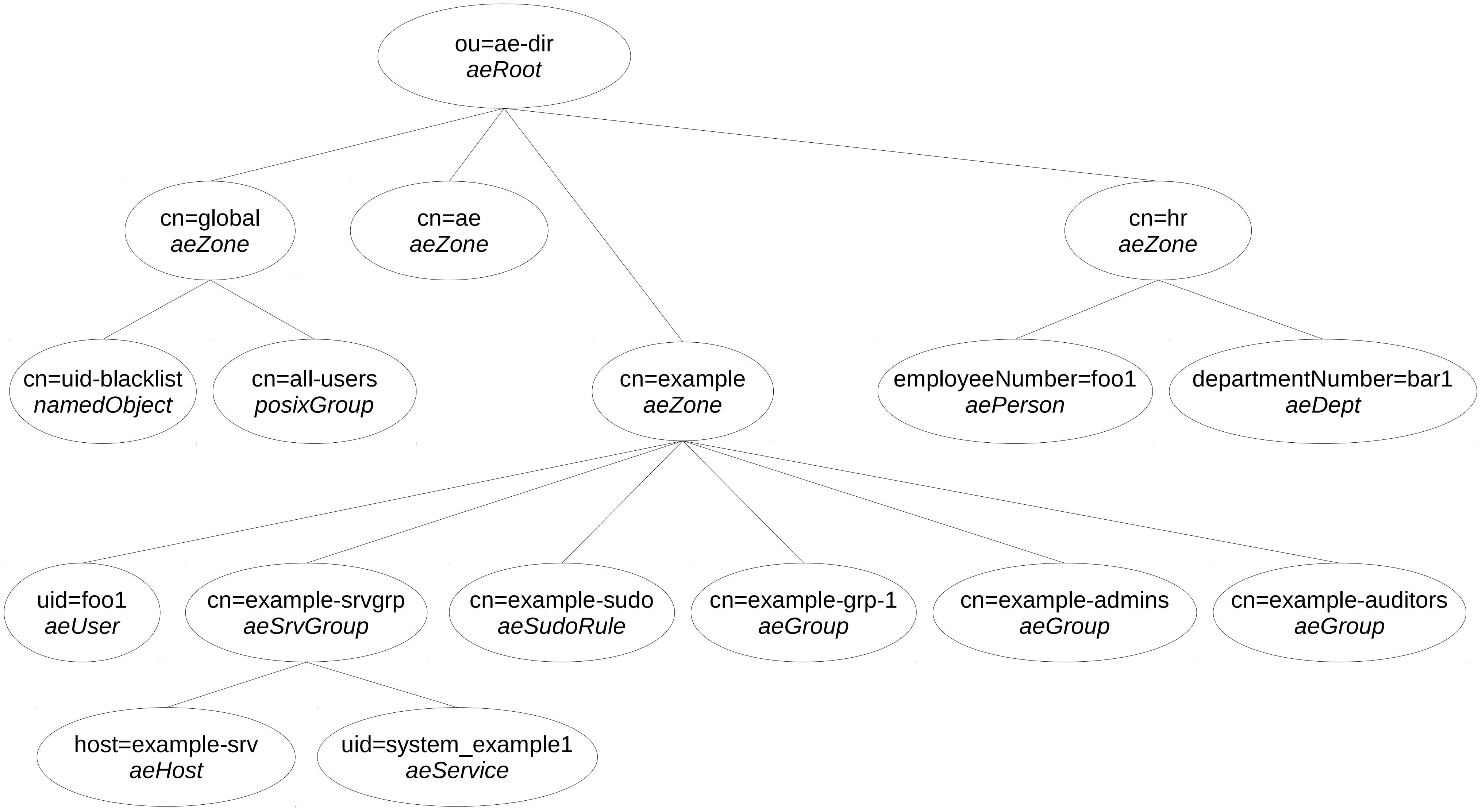
Roles

- No anonymous/guest access!
- ***Æ* admins** may *manage* everything within ou=ae-dir and can *read* cn=monitor and cn=config
- ***Æ* auditors** may *read* everything within ou=ae-dir
- **Zone admins** may *write* anything within a zone
- **Zone auditors** may *read* anything within a zone
- **Setup admins** may *write* aeHost/aeService
- **Users** may *read* own entries, other members of own groups, change own password

Schema: Requirements

- Compability to
  - NIS-LDAP (RFC 2307 and RFC2307bis)
  - sudo-ldap schema
- Support all common PAM/NSS clients, no strong need to have own PAM/NSS client
- => Æ-DIR's classes are sub-classes of standards
- Constraints to avoid input errors
- Common management meta data

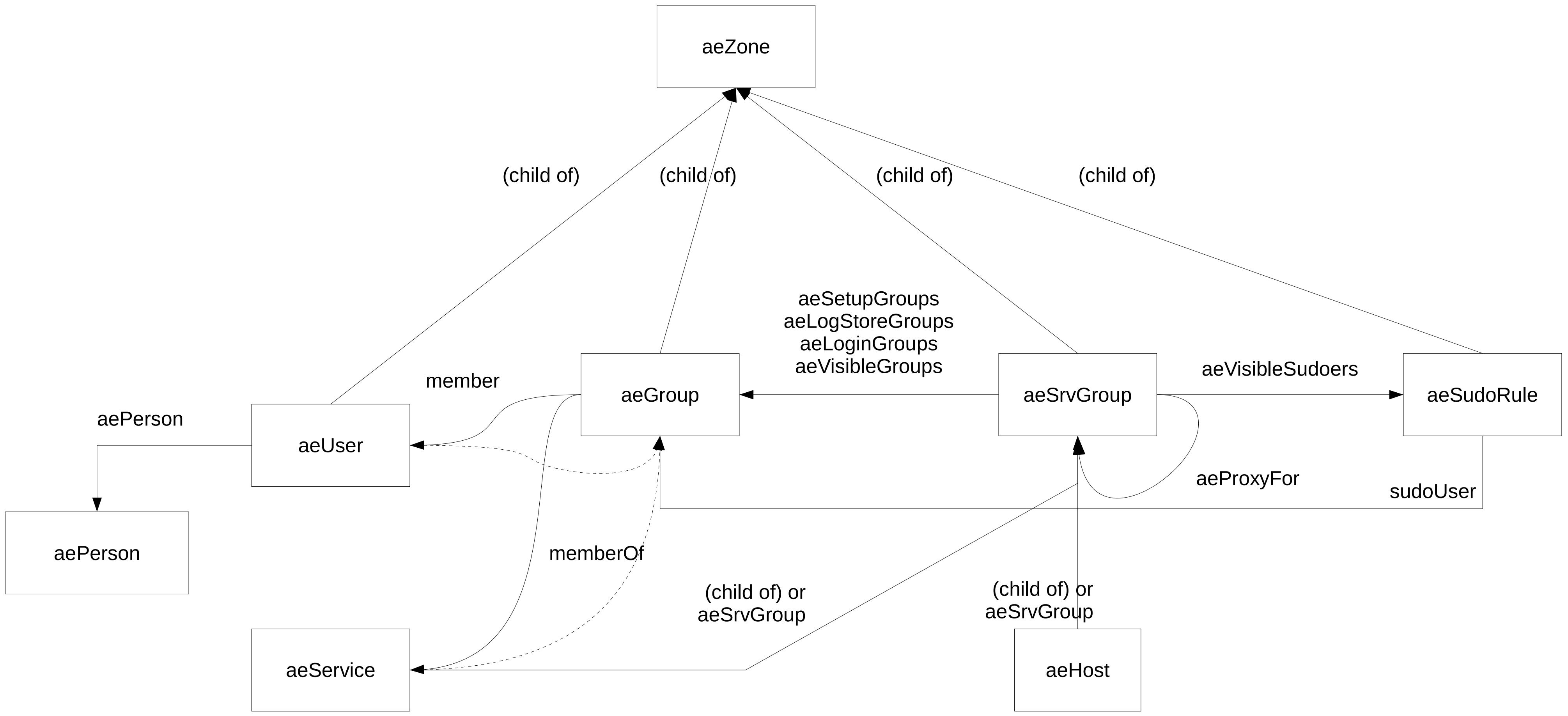
Directory Information Tree (DIT)



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
    - *aeZone* → *ae**
    - *aeSrvGroup* → *aeHost* / *aeService*
  - *sudoUser* → *aeGroup* backw. compatible by prefix name
- Cross-zone references 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 OpenLDAP-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 uniqueness 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 <u>one</u> 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! YMMV...
- 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* (see draft-findlay-ldap-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

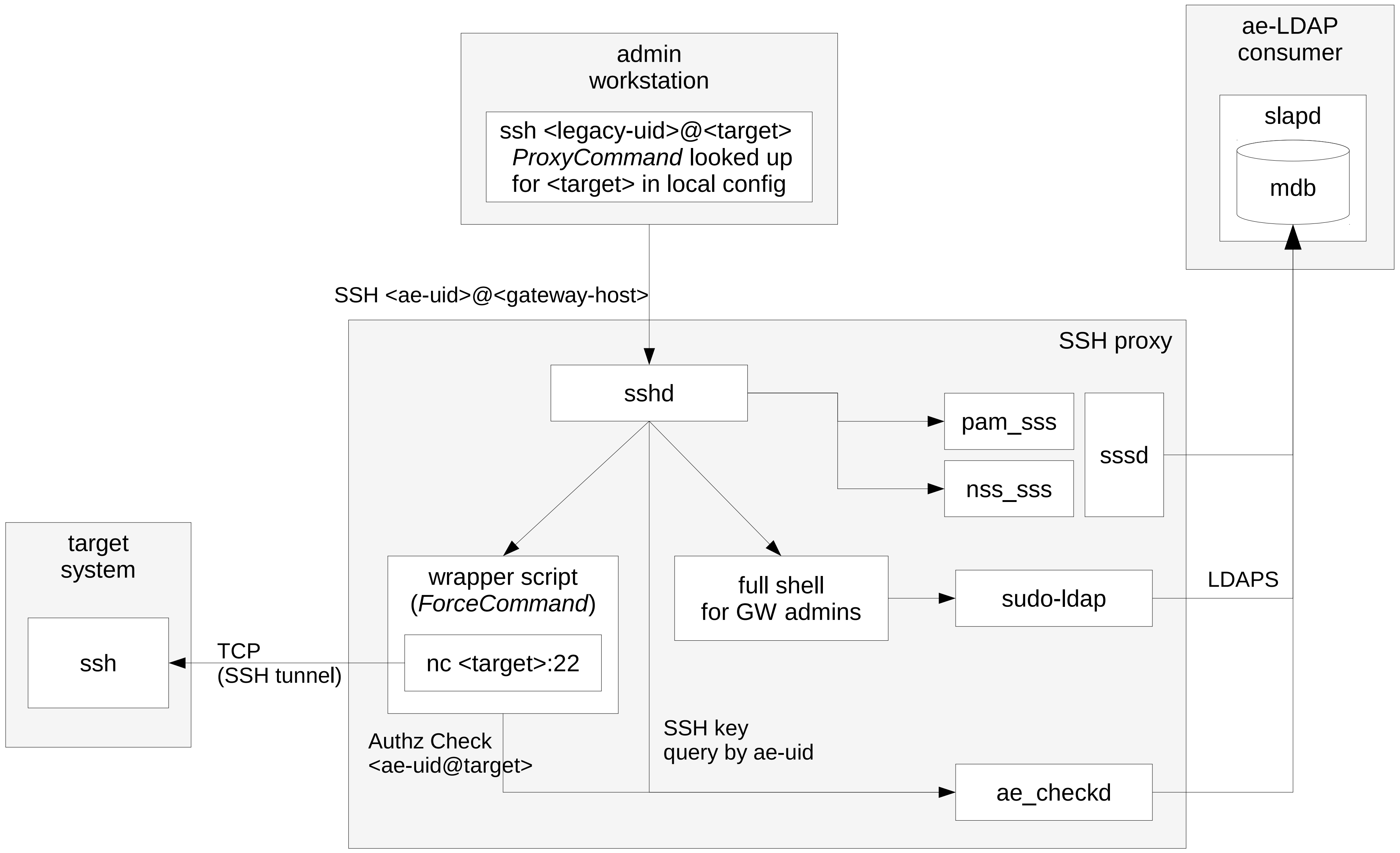
Schema: aeHost

- 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 with authorization



Conclusion (1)

- ACLs in OpenLDAP server are additional boundary against privilege escalation in frontends
- Still local components enforce access rights (e.g. OS enforces file ownership/permissions
- Privilege separation with separate credentials is a good thing
- Depends on how people are willing to use the mechanisms provided
  => awareness/teaching needed

Conclusion (2)

- You eventually need a fallback login if all fails, the exact procedures might differ
- (Set-based) ACLs are
  - quite complex
  - a performance hog (currently just more hardware)
- Change management:
  - It's hard to not open security holes afterwards
  - Upcoming ideas should always have a real use-case and fit into role model!
  - 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-sock*
- Replace set-based ACLs by custom *dynacl* module:
  - hopefully faster
  - evaluate *aeNotAfter* and *aeNotBefore*
  - skilled C programmers needed

Ideas: More integration

- 2-factor authc without separate infrastructure: Shared secrets, counters etc. in user entries (mainly done now)
- Machine deployment and network access control: Existing DHCP/DNS/RADIUS/PXE/TFTP schemas are a real mess
- MIT Kerberos (multiple realms?)
- Samba (multiple domains?)
- Config management: Tie *Puppet* node declaration or *ansible* playbook to *aeSrvGroup/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* for diversity:
  Are ACIs powerful enough?

Question & Answers