Integrating RHEL and LDAP/AD (Users and Groups)

Patrick Mooney
General Mills

Agenda

- Past methods
- Current methods
- Solutions
- Additional considerations
- Wrap-up / Discussion

Static Config (nss_files)

Place entries directly in /etc/(passwd|group)

Could be automated with config management

- Pros:
  - Simple
  - Most reliable
  - Best support
- Cons:
  - Doesn't scale
  - Not even with cfg mgmt.

Generated DB (nss_db)

Create BDB file for distribution to clients

Query LDAP/AD during generation for dynamic content

- Pros:
  - Reliable
  - Fast
  - Dynamic (to a degree)
- Cons:
  - Custom code needed
  - Synced groups still require grooming
  - Intersection with files

Others (NIS, nss_ldap)

No experience, no detail

- Pros:
  - Dynamic
- Cons:
  - Caching issues
  - Schema requirements

SSSD

- Previously covered at RHUG
- Caches locally – no nscd required
- Multi-domain/multi-source support
- Easy setup on domain-joined servers
- Automatic uid/gid translation (with caveats…)

Why isn't SSSD a good solution?

- Non-Linux devices may require stricter schema
- LDAP access issues
- Distributed responsibility for *NIX machines
- Active Directory scope…

Typical example.com LDAP

```
dn: uid=alice,ou=it-staff,dc=example,dc=com
uid: alice
memberOf: LinuxAdmins
memberOf: IT-Staff
memberOf: Employees
homeDir: /home/alice
userShell: /bin/zsh
uidNumber: 101
gidNumber: 1000
```

Realistic LDAP

```
dn: uid=bob,ou=salaried,ou=local,
        ou=site,dc=example,dc=com
uid: bob
memberOf: LinuxAdmins
memberOf: Linux-distlist
… 80 groups cut …
memberOf: Domain Users
memberOf: employees-birthday-party-distlist
memberOf: app_license_some-product
objectSid: WW91IHdpbiBhIHByaXplCg==
```

Realistic LDAP at GMI

- 56000 users
  - All members of Domain Users
  - No acceptable OU boundaries to filter on
- 45000 groups
  - Many used for purposes other than org-structure (software licensing, mailing lists, etc)
  - Few have Linux friendly names

Work-arounds

- sssd.conf - ldap_group_search_filter
  - Authoring filter is difficult or impossible
  - Groups still present, all stay numeric
  - Domain Users is always present
- Sync to IDM or openLDAP
  - Creates second source of truth
  - Syncing can be complicated

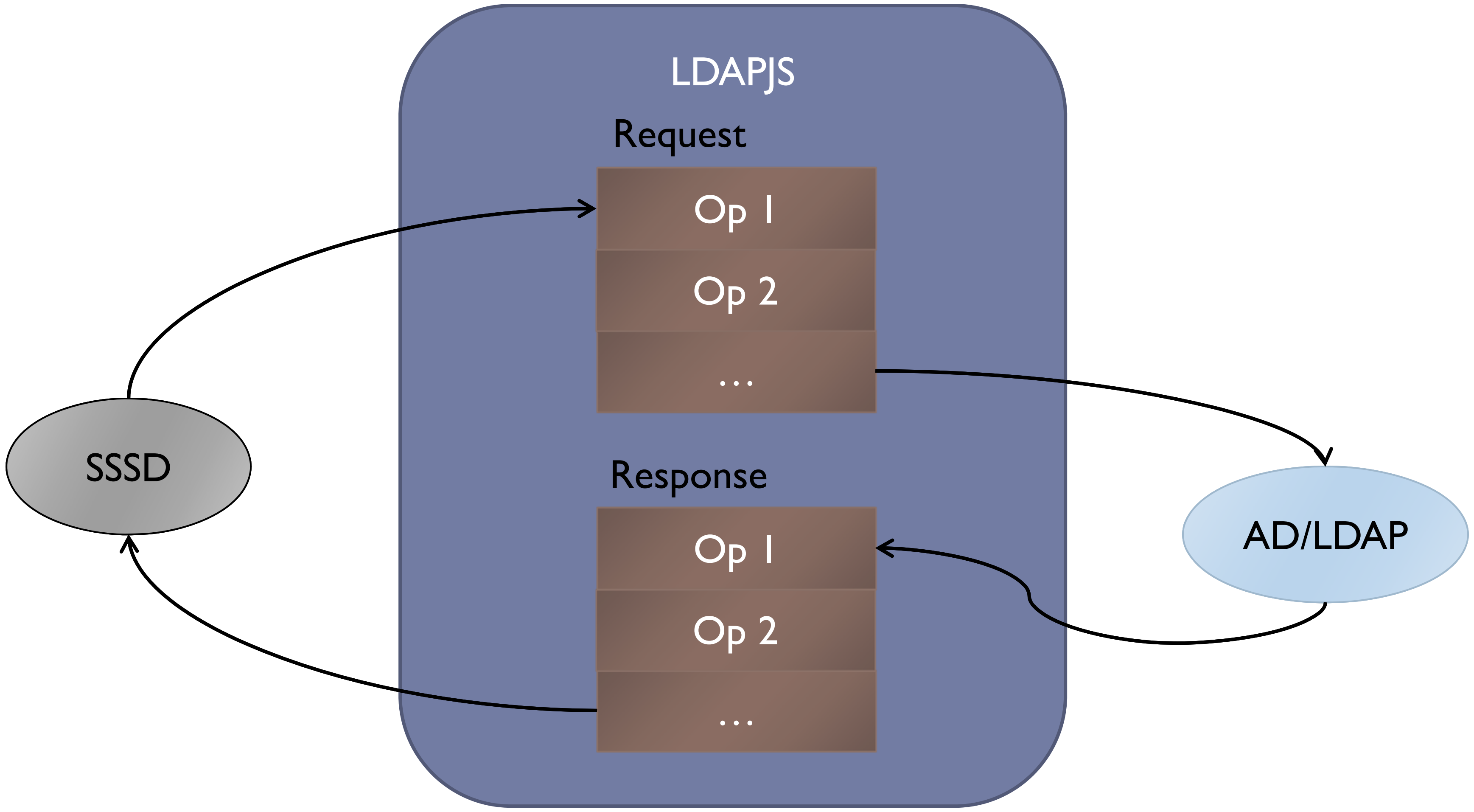
Filter on-the-fly

- Must Haves:
  - Live data
  - Tailored to the specific organization
  - Easy for admin/operations staff to maintain
- Avoid:
  - Data stored outside of single source of truth
  - Complicated caching or syncing

Proxy it (and mangle on the fly)

- Built on ldapjs
  - Runs on node.js (available in EPEL6)
  - Few alternatives for LDAP server APIs
    - openLDAP overlays – written in C
    - OpenDJ – written in Java
- Framework not a product
  - Abstracts/simplifies the LDAP plumbing
  - Requires application-specific setup

Architecture



Module Possibilities

- ObjectSID id-mapping (same as SSSD)
- Filter groups based on "complex" logic
  - Keep names Linux/UNIX safe
  - Prevent from appearing in memberOf/member
- Set shell/homedir based on group membership
- Translate schema on the fly (AD to rfc2307)

Example

```
var filterChain = new lmp.mangle.Chain()
.chain(new lmp.mangle.Simple(function (out) {
  out['cn'] = out['cn'].replace('Bob','Robert');
}))
.chain(new lmp.mangle.Simple(function (output) {
  var match = 'cn=restrict,ou=group,dc=test,dc=com';
  var dn = ldap.parseDN(match);
  output['memberOf'].forEach(function (group) {
    if (dn.equals(group)) {
      output['userShell'] = '/bin/lameshell';
    }
  });
}));
```

Example (continued...)

```
var client = ldap.createClient({
  url: "ldap://server.test.com:3268",
  bindCredentials: "myPassword",
  bindDN: "cn=myUser,ou=Users,dc=test,dc=com",
});
var log = bunyan.createLogger({name: 'Example'});
var proxy = new lmp.SearchProxy(client, filterChain,
log);
var server = ldap.createServer();
```

Example (continued…)

```
/* Allow anyone to bind */
server.bind('cn=root', function(req, res, next) {
  res.end();
  return next();
});
server.search(
  'DC=test,DC=com', proxy,  proxy.execute
);
server.listen(1389, '0.0.0.0', function () {
  console.log('LDAP server up at: ' +  server.url);
});
```

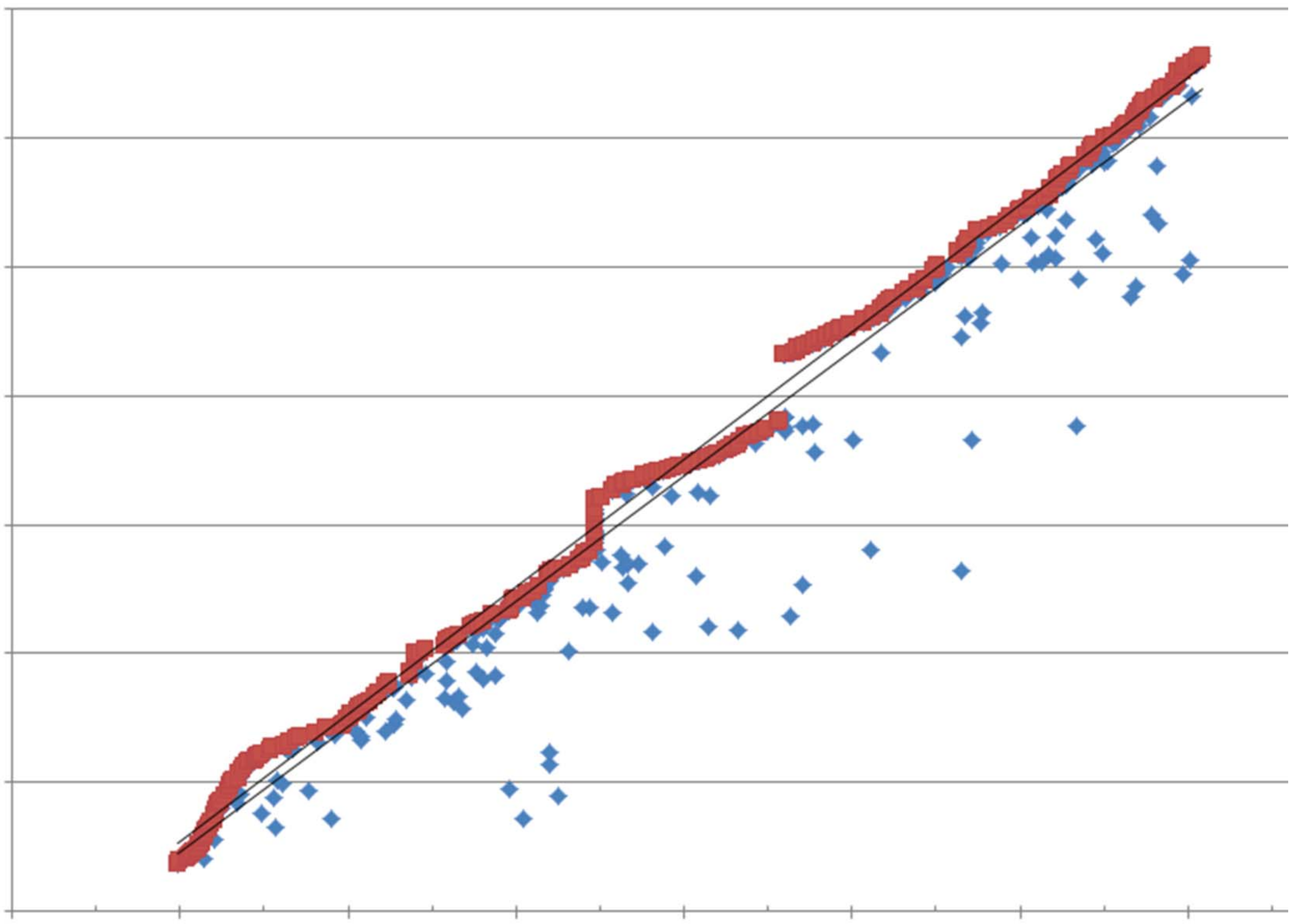
Current Status

- The Bad
  - Code is still pre-beta
  - Collection of modules is small
  - Have not performed exhaustive performance testing
- The Good
  - Using for an address book pilot
  - Planning to use for SSSD soon

Picking UID/GID ranges for ID mapping

- Relative-ID portion of objectSID determines offset
- Too low: overflowing objects will be invisible
- Too high: impedes multi-domain usage
- Get the data…

ObjectSID growth over time



Questions/Feedback

Resources

- ldapjs: github - mcavage/node-ldapjs
- node.js: nodejs.org
- proxy: github - pfmooney/node-ldapjs-mangle-proxy

patrick.mooney@genmills.com