Identity Mapping for Windows + UNIX Environments

Jordan Brown
Oracle

Problem

- No matter how you approach the data – whether from a local application, NFS, SMB, or another mechanism – you should be treated the same.
  - Files that you create should have the same ownership.
  - Your access rights should be the same.

Background – UNIX UID/GID

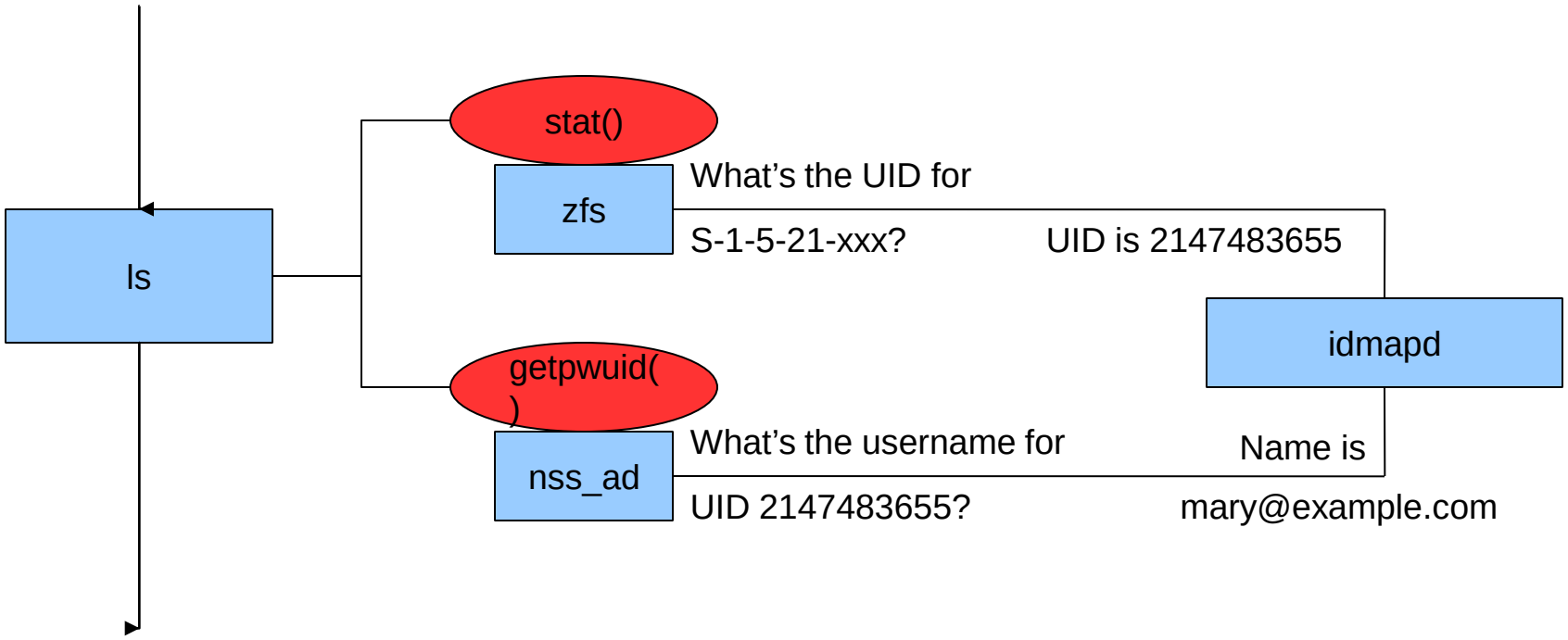
- UNIX internally tracks file ownership and access controls using integer "user IDs" and "group IDs".
- There is no provision for federation – central allocation is required.
- Users and groups use separate numbering spaces and are distinguished by context.

Background – Windows SID

- Windows tracks file ownership and access controls using variable length "security IDs".
- The probability of collision is very low.
- Users and groups use the same SID space and are distinguished by directory metadata.

Mapping Overview - ls

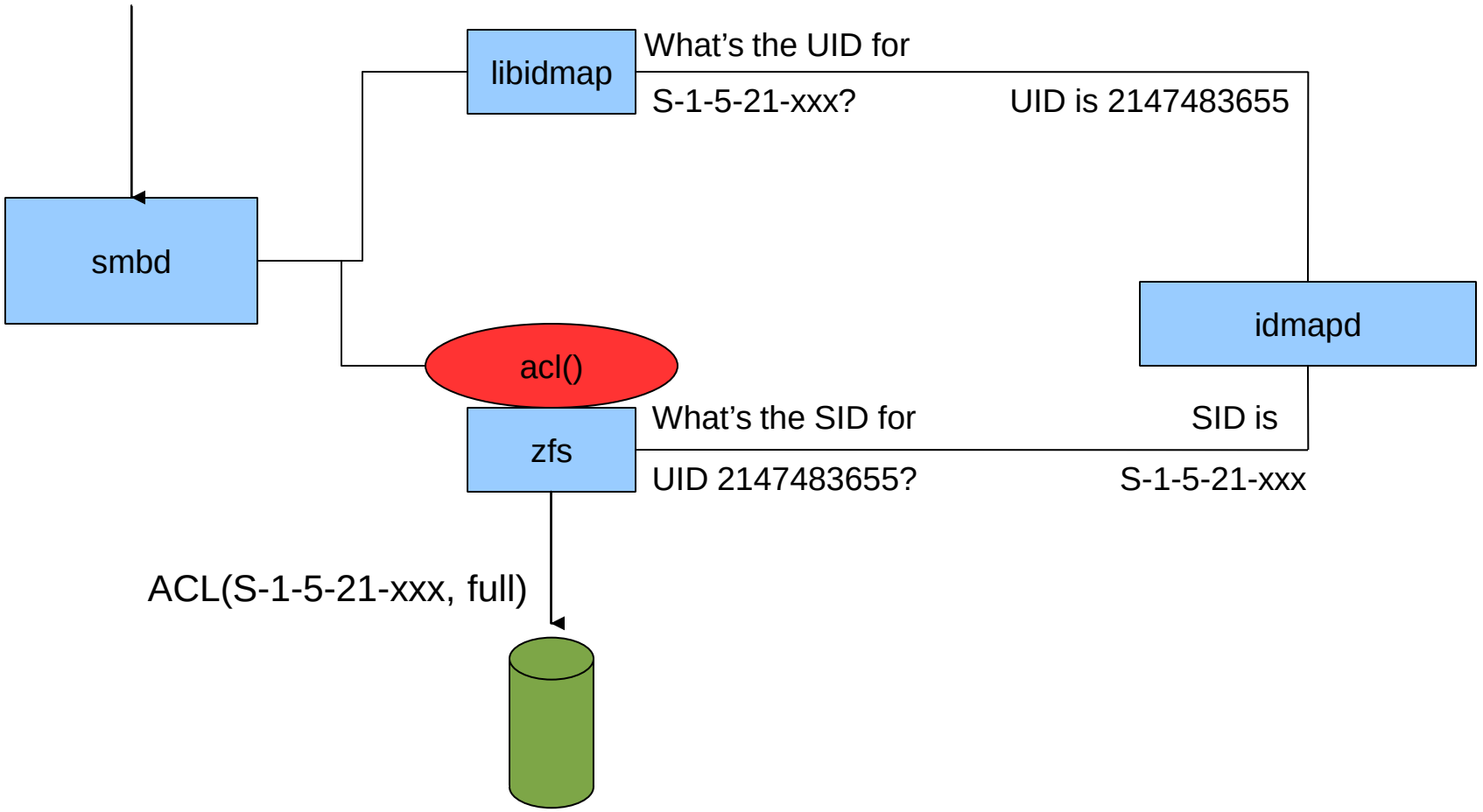

```
$ ls -l mary-file
```




```
-rw-r--r-- 1 mary@example.com staff 1169 2010-09-01 10:56 mary-file
```

Mapping Overview - SMB

SMB: set ACL(somefile, mary@example.com=full)



Multiple Mapping Mechanisms

- Ephemeral mapping
- Local UID/GID mapping
- Rule-based mapping
- Directory-based mapping

Ephemeral Mapping

- Ephemeral IDs support Windows users with no associated UNIX user.
- UID and GID ranges between 2^{31} and 2^{32} are reserved for ephemeral use.
- The SID, not the ephemeral ID, is stored in the file system.

```
# idmap show -c mary@example.com
winuser:mary@example.com -> uid:2147483655
```

Local UID/GID Mapping

- Unix UID/GID without Windows equivalents are mapped to local-machine-based SIDs.
- Very similar to the treatment of Windows local users.
- SMB server can look up names for these SIDs, just as for Windows.

Rule-based Mapping

- Locally-specified mapping rules: "Windows user **`joe.user@example.com`** is UNIX user **`joe`**".

```
# idmap add joe.user@example.com unixuser:joe
add     winname:joe.user@example.com    unixuser:joe

# idmap show -c joe.user@example.com unixuser
winname:joe.user@example.com -> unixuser:joe

# idmap show -c unixuser:joe
unixuser:joe -> winuser:joe.user@example.com
```

Rule-based Mapping

- Wild-card mapping: "All Windows users **@example.com** are the same as the same-named UNIX user".

```
# idmap list
add     winname:joe.user@example.com        unixuser:joe

# idmap add *@example.com unixuser:*
add     winname:*@example.com    unixuser:*

# idmap show -c sam@example.com
winuser:sam@example.com -> unixuser:sam

# idmap show -c unixuser:sam
unixuser:sam -> winuser:sam@example.com
```

Rule-based Mapping

- One-way mapping: "When Windows says fred@**example.com**, treat it as UNIX user **joe**".

```
# idmap list
add     winname:*@example.com    unixuser:*
add     winname:joe.user@example.com        unixuser:joe

# idmap add -d fred@example.com unixuser:joe
add -d  winname:fred@example.com        unixuser:joe

# idmap show -c fred@example.com
winuser:fred@example.com -> unixuser:joe

# idmap show -c unixuser:joe
unixuser:joe -> winuser:joe.user@example.com
```

Directory-based Mapping

- Information associating Windows users with UNIX users is stored in a directory service with other information about the user/group.
- Good for enterprise environments because the mappings are shared across servers.

Directory-based Mapping

- Active Directory mode:
  - IDMU. Standard Microsoft AD add-in; AD entry is augmented with UNIX information.
  - Name-based. AD entry is augmented with a custom attribute containing UNIX name.
- Good when storage admin is a Windows admin.

Directory-based Mapping

- "Native LDAP" mode:
  - UNIX directory entry is augmented with a custom attribute containing Windows name.
- Good when storage admin is a UNIX admin.

Diagonal Mapping

- Problem
  - UNIX files can be owned only by users, not groups.
  - Windows files can be owned by groups.
- Solution
  - Mapping is context sensitive.
  - Rules separately specify "straight" mappings (user-to-user, group-to-group) and "diagonal" mappings (user-to-group, group-to-user).

```
# idmap show -c 'Domain Admins@example.com' uid
wingroup:Domain Admins@hn-ads.com -> uid:2147483654

# idmap show -c 'Domain Admins@example.com' gid
wingroup:Domain Admins@hn-ads.com -> gid:2147483652
```

Tracing

- Problem
  - Mapping can be complex to configure.
  - Several network services are involved.
  - Why didn't it work? Why did it come up with **that** answer?
- Solutions
  - UIs to exercise mappings
  - Detailed logging of mapping steps, either optionally as part of explicit test mapping or by configuration for all mappings (very verbose!).

Tracing


```
#  idmap show -cV sam@example.com
winuser:sam@example.com -> unixuser:sam
Trace:
        winname sam@example.com -> unknown - Start mapping
        winname sam@example.com -> unixname - Not a well-known account
        winname sam@example.com -> unixname - Not a local SID
        winname sam@example.com -> unixname - Not found in mapping cache
        winname sam@example.com -> unixname - Not found in name cache
        winuser sam@example.com S-1-5-21-2142033078-2761654387-2715285478-1115
            -> unixname - Found with LSA
        winuser sam@example.com S-1-5-21-2142033078-2761654387-2715285478-1115
            -> unixuser - Matching rule: *@example.com -> *
        winuser sam@example.com S-1-5-21-2142033078-2761654387-2715285478-1115
            -> unixuser sam 32142 - UNIX name found
        winuser sam@example.com S-1-5-21-2142033078-2761654387-2715285478-1115
            -> unixuser sam 32142 - Rule-based mapping
        winuser sam@example.com S-1-5-21-2142033078-2761654387-2715285478-1115
            -> unixuser sam 32142 - Done
```

Thhhat's All, Folks!

Q&A