

Winbind Varlink Service

What is it and what is it for?

Samuel Cabrero <scabrero@samba.org>

About me

- Zentyal
 - First contact with the Samba team
 - Samba XP 2013, *Challenges and experiences of the Samba 4.0 integration in Zentyal server*
- Joined SUSE in 2017
 - SUSE Labs Samba team
- Samba Team member since 2019

Long history short

■ What `winbind varlink service` is it?

A `nss_winbind.so` replacement,
based on `systemd` recent features.

■ What is it for?

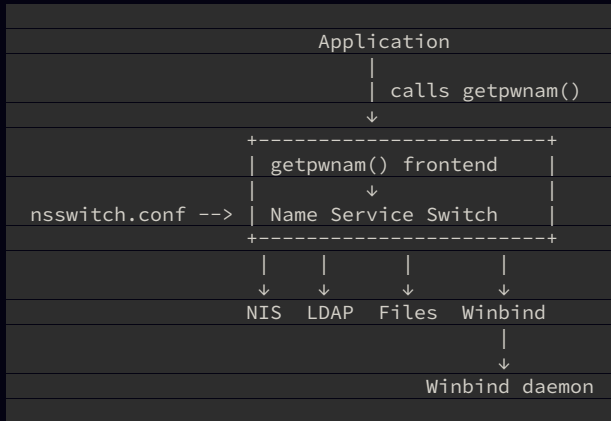
For the same as `nss_winbind.so`.

■ Then? What the motivation was?

Postponed.

Name Service Switch

- API that connects a computer with a variety of sources of common configuration databases and name resolution mechanisms
- Designed after a method used by Sun Microsystems in the C library of Solaris 2
- Implemented in the GNU C Library
- Defines
 - Interface for applications to query databases
 - Interface for modules to provide information



`/etc/nsswitch.conf:`

```
passwd:    files winbind
shadow:    files
group:     files winbind
...
```

NSS modules naming scheme

```
/* Query users */
NSS_STATUS _nss_winbind_getpwuid_r();
NSS_STATUS _nss_winbind_getpwnam_r();

/* Enumerate users */
NSS_STATUS _nss_winbind_setpwent();
NSS_STATUS _nss_winbind_endpwent();
NSS_STATUS _nss_winbind_getpwent_r();
```

```
/* Query groups */
NSS_STATUS _nss_winbind_getgrnam_r();
NSS_STATUS _nss_winbind_getgrgid_r();
NSS_STATUS _nss_winbind_initgroups_dyn();

/* Enumerate groups */
NSS_STATUS _nss_winbind_setgrent();
NSS_STATUS _nss_winbind_endgrent();
NSS_STATUS _nss_winbind_getgrent_r();
NSS_STATUS _nss_winbind_getgrlst_r();
```

```
struct passwd {
    char    *pw_name;        /* user name */
    char    *pw_passwd;     /* user password */
    uid_t   pw_uid;         /* user ID */
    gid_t   pw_gid;         /* group ID */
    char    *pw_gecos;      /* user information */
    char    *pw_dir;        /* home directory */
    char    *pw_shell;      /* shell program */
};
```

```
struct group {
    char    *gr_name;       /* group name */
    char    *gr_passwd;     /* group password */
    gid_t   gr_gid;        /* group ID */
    char    **gr_mem;       /* group members */
};
```


* `getgrlst_r()` is a `getgrent_r()` version without enumerating members for each group

Systemd User/Group record Lookup API

- Takes the role of NSS for `passwd` and `group`
 - Allows applications to query users and groups from local services
 - Allows local services to provide users and groups to local applications
- The user and group records are JSON strings
- Varlink IPC transport
- Simple API, only 3 methods

Introduced concepts

- Varlink
- Systemd record lookup API
- Systemd user and group records



Varlink

■ Introduction

- It is an interface description format and IPC protocol
- Based on JSON
- Connection oriented transport
- C library and tools: <https://github.com/varlink/libvarlink>

Varlink

Interface

- Defined in a plain text file
- Reverse-domain name
- Definition contains methods, types, and errors returned from method calls

```
interface io.systemd.UserDatabase
```

```
method GetUserRecord(uid: ?int, userName: ?string, service: string) ->  
    (record: object, incomplete: bool)
```

```
method GetGroupRecord(gid: ?int, groupName: ?string, service: string) ->  
    (record: object, incomplete: bool)
```

```
method GetMemberships(userName: ?string, groupName: ?string, service: string) ->  
    (userName: string, groupName: string)
```

```
error NoRecordFound()
```

```
error BadService()
```

```
error ServiceNotAvailable()
```



```
error ConflictingRecordFound()
error EnumerationNotSupported()
```

Varlink

Protocol

- Messages are encoded as JSON encoded and terminated with a single NUL byte.
- A service responds to requests in FIFO order, messages are never multiplexed
- Requests can be queued
- Multiple responses for a single request using `more` and `continues` flags.

Request:

```
{
  "method":
    "io.systemd.UserDatabase.GetUserRecord",
  "parameters": {
    "service": "org.samba.winbind"
  },
  "more": true
}
```

Responses:

```
{
  "parameters": {
    "incomplete": false,
    "record": {
      "gid": 100513,
      "homeDirectory": "/home/AFOREST/sshsvc",
      "service": "org.samba.winbind",
      "shell": "/bin/bash",
      "uid": 101103,
      "userName": "AFOREST\\sshsvc"
    }
  }
}
```

```
}  
},  
"continues": true  
}
```

Varlink

The Varlink's `service` interface

- Every varlink service offers this interface
 - Describes all interfaces the service provides
 - Provides information about the service implementation itself
- Interface name is `org.varlink.service`

```
interface org.varlink.service  
  
method GetInfo() -> (  
    vendor: string,  
    product: string,  
    version: string,  
    url: string,  
    interfaces: []string  
)  
  
method GetInterfaceDescription(interface: string) -> (description: string)
```

```
error InterfaceNotFound(interface: string)
error MethodNotFound(method: string)
error MethodNotImplemented(method: string)
error InvalidParameter(parameter: string)
error PermissionDenied()
error ExpectedMore()
```

Varlink

Why varlink?

1. Can be used during early boot and late shutdown
2. Protocol is JSON based, perfect to transport JSON data
3. Allows streaming, used to enumerate users/groups

Systemd User/Group record Lookup API usage

Services

- Each subsystem that needs to define users and groups on the local system is supposed to
 1. Implement the `io.systemd.UserDatabase` interface
 2. Offer its interfaces on a Varlink AF_UNIX/SOCK_STREAM file system socket bound into the `/run/systemd/userdb/` directory

Clients

- When a client wants to lookup a record, it contacts all sockets in the directory in parallel enqueueing the same query.
- First positive response is returned to applicaiton, or if all fail, last seen error.

Well-known services

- `io.systemd.Multiplexer`
 - Multiplexes client queries to all other running services

- Simplifies client development
- Not available during earliest boot and final shutdown phases
- `io.systemd.NameServiceSwitch`
 - Makes the classic NSS user/group records available as JSON User/Group records
- `io.systemd.Home`
- `io.systemd.DynamicUser`
- `io.systemd.Machine`

Systemd User/Group record Lookup API - Retrocompatibility

Compatibility with NSS

- When using systemd API, lookups into NSS databases handled by `io.systemd.NameServiceSwitch` service
- When using NSS API, `nss_systemd.so` will synthesize NSS structures from JSON records
- man `systemd-userdbd.service`

Remarks

- Unlike in NSS there is no service order
 - First service to answer wins
 - The API does not define any mechanism to deal with collisions
 - Delegates responsibility to system administrator
- The API does not provide caching

Records (I)

User records

- KV format, JSON encoded
- Longer and more diverse than NSS `struct passwd`
- Seven sections
 - regular
 - privileged
 - perMachine
 - binding
 - status
 - signature
 - secret
- Intended to be extensible

```
{
  "autoLogin" : true,
  "disposition" : "regular",
  "enforcePasswordPolicy" : false,
  "lastChangeUsec" : 1565950024279735,
  "userName" : "grobie",
  "memberOf" : [ "wheel" ],
  "privileged" : {
    "hashedPassword" : [
      "$6$WHBKvAFFT9jKPA4k..."
    ]
  },
  "signature" : [
    {
```

- Windows credential information?
- Parental control?
- Examples using extended information:
 - `pam_systemd`
 - `systemd-logind`
 - `systemd-homed`

```

        "data" : "LU/HeVrPZSzi3MJ0PVHwD5m/xf5...",
        "key" : "-----BEGIN PUBLIC KEY-----\n..."
      }
    ],
    "binding" : { "15e19cf24e004b949dda..." : { ... } },
    "status" : { "15e19cf24e004b949ddaa..." : { ... } }
  }

```

[1] https://systemd.io/USER_RECORD

Records (II)

- Group Records are to `struct group` what User Records are to `struct passwd`
- They also consist of seven sections
- Similar properties and they carry some identical (or at least very similar) fields.

```

{
  "groupName" : "grobie",
  "binding" : {
    "6b18704270e94aa896b003b4340978f1" : {
      "gid" : 60232
    }
  },
  "disposition" : "regular",
  "status" : {
    "6b18704270e94aa896b003b4340978f1" : {
      "service" : "io.systemd.Home"
    }
  }
}

```

```
}
```

[1] https://systemd.io/GROUP_RECORD

Methods (I)

GetUserRecord()

- Looks up or enumerates users

Request

- If only `uid` given --> Lookup user by UID
- If only `userName` given --> Lookup user by name
- Both given --> Lookup user matching both
- None given --> Enumerate users (optional, can return `EnumerationNotSupported`)
 - Call needs `more` flag

```
method GetUserRecord(  
    uid : ?int,  
    userName : ?string,  
    service : string  
) -> (  
    record : object,  
    incomplete : bool  
)
```


- Each reply carries a record.
- The service parameter mandatory, in this case `org.samba.winbind`

Response

- User record returned in the `record` field
- `incomplete` indicates if parts of the record have been removed, like the `privileged` field

Methods (II)

GetGroupRecord()

- Looks up or enumerates groups
- Works as GetUserRecord but for groups

```
method GetGroupRecord(  
    gid : ?int,  
    groupName : ?string,  
    service : string  
) -> (  
    record : object,
```

```
)  
incomplete : bool
```

Methods (III)

GetMemberships()

- Inquire about group memberships
- Unlike `GetUserRecord` and `GetGroupRecord`, lists of memberships returned by different services are always combined
- **It is the authoritative source about memberships**
 - It is not enough checking the `memberOf` field from a user record
 - Or the `members` field of a group

Request

```
method GetMemberships(  
    )
```

- If only `userName` given --> List matching user memberships
- If only `groupName` given --> List matching group members
- None given --> List all known memberships of any user and any group
- Both given --> Check the membership
- Needs `more` flag unless both given

```

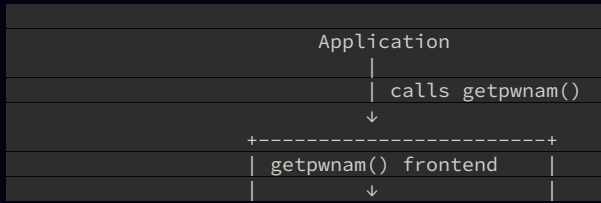
    userName : ?string,
    groupName : ?string,
    service : string
) -> (
    userName : string,
    groupName : string
)

```

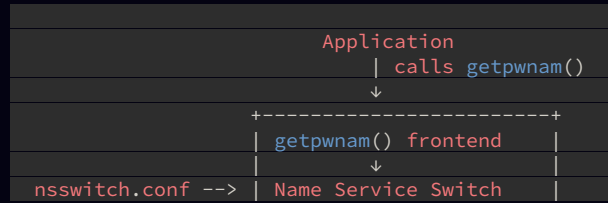


Replaced parts

Application using NSS API:



Application using NSS API:



Proof of concept

- Host and some images already have `nss_systemd` installed and enabled
- If winbind provides a varlink service...
 - Just need to bind-mount `/run/systemd/userdb/` in the host and containers

[1] <https://www.suse.com/c/the-first-prototype-of-adaptable-linux-platform-is-live>

Implementation details

- `io.systemd.UserDatabase` interface implemented in winbind
 - Supports enumeration
 - Implemented in
 - `source3/winbindd/winbindd_varlink.h`
 - `source3/winbindd/winbindd_varlink.c`
 - `source3/winbindd/winbindd_varlink_getuserrecord.c`
 - `source3/winbindd/winbindd_varlink_getgrouprecord.c`
 - `source3/winbindd/winbindd_varlink_getmemberships.c`

- Build with `--with-systemd-userdb`
- Activate with `winbind varlink service = yes`
- Records only contain `struct passwd` and `struct group` equivalent fields
- Using <https://github.com/varlink/libvarlink>
 - PR #58 `varlink help` not working with camel-case interface names
 - PR #59 `VarlinkStream` not dispatching out data when `write()` returns `EAGAIN`
 - PR #60 Build fixes with `-Werror=cast-qual` and `-Werror=implicit-fallthrough`
 - Bundled into `third_party/varlink`, `--bundled-libraries=varlink`
- Systemd had problems handling `DOMAIN\user` format
 - PR #26386 `userdb: Skip unsafe characters check parsing memberships`
- **Draft MR** https://gitlab.com/samba-team/samba/-/merge_requests/2928

DEMO

```
services:  
  # runs a samba ADDC  
dc:  
  # runs winbindd, varlink service enabled  
varlink:
```

```
volumes:
  - vl-local-samba-wb-socket:/usr/local/samba/var/run/winbindd
  - vl-systemd-userdb:/var/run/systemd/userdb

# runs smbd
smb:
  volumes:
    - vl-local-samba-wb-socket:/usr/local/samba/var/run/winbindd
    - vl-systemd-userdb:/var/run/systemd/userdb

# runs sshd
ssh:
  volumes:
    - vl-systemd-userdb:/var/run/systemd/userdb
```

Questions/Discussion

From https://systemd.io/USER_GROUP_API

Other projects are invited to implement these services too. For example, it would make sense for LDAP/ActiveDirectory projects to implement these interfaces, which would provide them a way to do per-user resource management enforced by systemd and defined directly in LDAP directories.