Cybersecurity

Subject: POSIX ACL

Here, we are going to play with access control mechanisms of the POSIX (Portable Operating System Interface) standard, prevalent in the whole family of Unix-like operating systems.

1. POSIX Access Control Lists

POSIX ACL (*Access Control Lists*) was introduced to extend the standard Unix permission mechanism that controls access to a file (or directory) for the owner (u), owning group (g), and other users (o), hereafter called *basic permissions*. The extension allows defining authorization entries (ACE, *Access Control Entry*) for any individual user and/or group. However, POSIX ACLs are still limited to `read`, `write` and `execute` permissions.

1.1 POSIX authorization

Permissions traditionally used in Unix/Linux systems are limited to the following:

- `read`—reading a file / listing directory contents
- `write`—writing to a file / directory *i*-node modification
- `execute`—executing a program / in-directory operations

1.2 Access control algorithm

The POSIX standard offers the ability to mask permissions using the *mask* field. The *effective* access permissions are the bit product of the assigned permissions (ACE) and the mask. The access control algorithm performs the following steps (until the first match):

1. If the user owns a file—use the owner's ACE permissions (user::),
2. If the user is explicitly listed on the ACL—use the user's *effective* permissions,
3. If one of the user's groups is the owning group, or any group explicitly listed on the ACL—apply that group's *effective* permissions; if no matched group have sufficient effective rights—access is denied,
4. Eventually, the ACE of other users (other::) determine the access rights.

To preserve compatibility, the user::, group:: and other:: rights from the ACL are mapped to the *basic permissions* u, g and o, that can be displayed with the traditional ls command. However, with POSIX ACL at work, there is a slight discrepancy from the old-fashioned ls behavior: when the mask is set, the rights for the owning group (g) displayed with ls correspond to the mask value, not the group:: entry.

1.3 POSIX ACL rights management

Two commands operate on ACLs, one—getfacl—is for reading current permissions for files and directories, the other—setfacl—for changing them.

```
% ls -l
 -rw-r--r-- 1 james users 1000 2022-10-01 09:00 file.txt

% getfacl file.txt
# file: file.txt
# owner: james
# group: users
user::rw-
group::r--
other::r--
```

Adding a new ACE:

```
% setfacl -m user:john:rwx plik.txt
% getfacl file.txt --omit-header
user::rw-
user: john:rwx
group::r--
mask::rwx
other::r--
```

Changing permissions (in fact, not different from adding new ACE):

```
% setfacl -m u:john:r file.txt
% getfacl file.txt --omit-header
user::rw-
user: john:r
group::r--
mask::r--
other::r--
```

Removing permissions (a particular ACE):

```
% setfacl -x u:john file.txt
% getfacl file.txt --omit-header
user::rw-
group::r--
mask::r--
other::r--
```

It's worth noting, that a simple pipe can be used to copy an entire ACL from one file to another:

```
% getfacl file1.txt | setfacl -M- file2.txt
```

1.3.1 Default permissions

Default permissions are applied only to directories and allow to automatically grant extended permissions to newly created objects in a directory that has default permissions set up. The `x` permission is treated in a special way and, although it may be set in the default permissions, it is not effectively granted to newly created regular files.

Adding default permissions:

```
% setfacl -d -m group:students:wx directory
% getfacl directory --omit-header
user::rwx
group::r-x
other::r-x
default:user::rwx
default:group::r-x
default:group:students:-wx
default:mask::rwx
default:other::r-x
```

Modification and removal of default permissions is done in a similar way.

Copying all extended permissions AVEs to default permissions ACEs can be performed as follows:

```
% getfacl --access directory | setfacl -d -M- directory
```

1.3.2 Mask

ACL rights explicitly granted to users (except the owner) and groups are constrained by the mask (the mask specifies the maximum effective permissions). You can set the mask value using the following command:

```
% setfacl -m mask::rwx file.txt
```

For example, blocking the write permission using a mask is achieved with the command:

```
% setfacl -m m::rx file.txt
```

Using the `setfacl` command (also `chmod` on an object with extended permissions) will, by default, automatically alter the mask value, when manipulating permissions for explicitly listed users, owning group, and explicitly listed groups. To avoid automatic mask recalculation, `-n` option of `setfacl` can be used.

1.3.3 ACL removal

It is also possible to completely remove extended ACL permissions (`-b` option) or default permissions (`-k`).

🕮 Problems to discuss:

- What is the necessary condition that lets you change POSIX ACL permissions to a given object?