NetApp Knowledge Base

How to understand UNIX permissions

https://kb.netapp.com/Advice_and_Troubleshooting/Data_Storage_Software/ONTAP_OS/How_to_und…

Updated: Wed, 31 Aug 2022 13:32:48 GMT

Applies to

- UNIX security style
- POSIX-compatible file systems

Description

The following article describes the functionality of UNIX permissions

'NetApp provides no representations or warranties regarding the accuracy or reliability or serviceability of any information or recommendations provided in this publication or with respect to any results that may be obtained by the use of the information or observance of any recommendations provided herein. The information in this document is distributed AS IS and the use of this information or the implementation of any recommendations or techniques herein is a customers responsibility and depends on the customers ability to evaluate and integrate them into the customers operational environment. This document and the information contained herein may be used solely in connection with the NetApp products discussed in this document.'

Procedure

- In a multiuser environment, it is necessary to ensure that a user cannot access or modify files or directories that they are not supposed to
- File permissions provide a protection mechanism for controlling access to files and directories
- Under UNIX security style, a file or directory can be accessed or modified by the superuser (typically uid 0), owner, or a group of users who have been given permission to do so
- Permissions can also be defined for other users that do not belong to either of these two categories

- Every file (and directory) has an owner, an associated Unix group, and a set of permission flags that specify separate read, write, and execute permissions for the "user" (owner), "group", and "other"
- File ownership is an important component of UNIX that provides a secure method for storing files

  Every file in Unix has the following attributes:

  - Owner permissions − The owner's permissions determine what actions the owner of the file can perform on the file
  - Group permissions − The group's permissions determine what actions a user, who is a member of the group that a file belongs to, can perform on the file
  - Other permissions − The permissions for others indicate what action all other users can perform on the file

- While reviewing security information of a file or directory, you will see permissions via [file-directory show]() commands

  The UID or Owner along with GID or Group will be listed:

```
cluster::> vserver security file-directory show -vserver svm0 -path /vol1


                Vserver: svm0
              File Path: /vol1
      File Inode Number: 64
         Security Style: unix
        Effective Style: unix
         DOS Attributes: 10
 DOS Attributes in Text: ----D---
Expanded Dos Attributes: -
           UNIX User Id: 0
          UNIX Group Id: 0
         UNIX Mode Bits: 777
UNIX Mode Bits in Text: rwxrwxrwx
                  ACLs: -
```

- From the output, we can see the Unix User Id (UID) and the Unix Group Id (GID)

  These two Id's will show you who owns the file or directory along with the group that will grant access based on the Mode Bits set

- Before moving forward, let's discuss Mode Bits

  What is a Mode Bit?

  - A Mode Bit is a bit that indicates the current mode of execution
  - A better way to say this is, it is basically the permissions you are setting for UID, GID, and other
- You will see the Mode Bits as numbers or letters

  Example:

  - 770 or rwxrwx---
  - For numeric, the first number represents UID, the second number represents GID, and the third number represent OTHER
  - With alpha, or letters, the first three numbers is UID, the second, three numbers is GID, the last three are OTHER
- How are Mode Bits calculated? (What does 770 mean?)
  - For each number, there is a calculation
  - Each application permission has an associated value
  - These values are added to make a numeric Mode Bit
    - Mode Bit values:
      - Read = 4
      - Write = 2
      - Execute = 1
  - **Example:**
    - 7 means read, write, and execute
    - The get to 7, you add the bits for each

Number	**Permissions**	**rwx**
0	No Permissions	---
1	Execute Only	--x

Number	Permissions	rwx
2	Write Only	-w-
3	Write and Execute	-wx
4	Read Only	r--
5	Read and Execute	r-x
6	Read and Write	rw-
7	Read, Write and Execute	rwx

- We understand numeric Mode Bits now, but what doe rwxrwxrwx mean?
    - This is a numeric representation of the Mode Bits
        - "r" means Read
        - "w" means Write
        - "x" means Execute
    - This will be the same permissions seen previously in numeric format
- Now that we understand what Owner, Group, and Other is along with Mode Bits. And we know where to find the UID and GID of a file or directory. Let’s talk about how to decipher this information.
    - When you look at file-directory show, and determine the GID or UID, this is the first step to determine the permissions.
    - If the UID or GID match, the applicable permissions are applied.
    - But, in the case a user does not match, they are grouped into OTHER.
- An easy way to about what a UID or GID will fall under, is to picture them in silos, containers, or buckets!

Additional Information

- How to change permissions on a UNIX or UNIX-like host with chmod
- How to change ownership on a UNIX or UNIX-like host with CHOWN