****

**המחלקה להנדסת מערכות מידע ותכנה**

אבטחת מחשבים ורשתות תקשורת

**(372-1-4601)**

עבודה 2

תאריך ההגשה – 25.05.2019

הוראות כלליות:

* הגשה ביחידים או בזוגות.
* העבודה מורכבת משני חלקים – חלק תיאורטי וחלק מעשי.
* הוראות הגשה מופיעות בסוף החלק המעשי.
* שאלות על העבודה יש לשאול **אך ורק** בפורום העבודות במודל, שאלות אשר ישאלו בדוא"ל לא יענו!
* יש לכתוב תשובות מלאות ומפורטות.
* כחלק מתהליך בדיקת העבודה תתבצע בדיקה לזיהוי עבודות מועתקות – **אל תעתיקו.** כל מקרה של העתקה יטופל על ידי ועדת משמעת אוניברסיטאית.

חלק תיאורטי (60 נק')

# פרוטוקולי אימות (10 נקודות)

נתון פרוטוקול המאפשר לשני לקוחות ו- לייצר מפתח סימטרי משותף בעזרת שרת אמון .

הפרוטוקול **איננו מאובטח**

* 1. **(3 נק')** תאר את ההתקפה והדגם את תרחיש התקיפה?
  2. **(3 נק')** מהן היכולות הנדרשות מן התוקף בכדי לממש את ההתקפה?
  3. **(4 נק')** כיצד ניתן לתקן את הפרוטוקול בכדי להתמודד עם ההתקפה שתיארת?

# Kerberos (10 נקודות)

1. **(5 נק')** הסבירו על מנגנון Kerberos.   
   בתשובתכם, פרטו:
   1. מהי מטרת הפרוטוקול
   2. מי הם השחקנים והרכיבים במערכת
   3. מהו התהליך שמבוצע מרגע התחלת הבקשה לשירות (service) מסוים ועד קבלת גישה עבורו?
2. **(5 נק')** פרוטוקול Kerberos מבוסס על פרוטוקול Needham-Schroeder.
   1. תאר את פרוטוקול Needham-Schroeder המבוסס על הצפנה באמצעות מפתח פומבי.
   2. תאר התקפה על פרוטוקול Needham-Schroeder.
   3. תאר כיצד ניתן לתקן את הפרוטוקול כדי להימנע מסוג תקיפה זה.

# Firewalls (15 נקודות)

לפניכם תרשים של רשת ארגונית:



כפי שניתן לראות בתרשים, הרשת מורכבת משלוש עמדות קצה (A,B,C), שרת FTP (FS), שרת HTTP (HS) ושרת מייל SMTP (MS). כמו כן, הרשת מוגנת ע"י שלושה חומות אש: שתיים מסוג Stateless packet filter ללא deep packet inspection (PF1, PF2) ואחת מסוג שרת פרוקסי המסוגלת לפעול ברמת האפליקציה ((PRX.

1. **(5 נק')** ענו נכון/לא נכון והסבירו בקצרה את תשובתכם לגבי כל אחת מהטענות הבאות:
2. כל פנייה לשרת FS (בין אם הפניה מגיעה מחוץ לארגון או מתוכו) עוברת דרך לפחות חומת אש אחת.
3. כל בקשה של שירות (service) אשר מגיעה מחוץ לרשת הארגונית עוברת דרך שתי חומות אש.
4. לא ניתן ליצור חיבור ישיר אל שרת HS.
5. ניתן ליצור חוק המונע גישה של עמדה C לשרת MS.
6. **(3 נק')** בארגון עלה חשד כי העובד בעמדה A שולח בקשות HTTP זדוניות לשרת HS. האם יכול הארגון לזהות ולחסום תקשורת HTTP מעמדה A לשרת HS וזאת ללא שינוי מבנה הרשת. **פרטו!**
7. **(8 נק')** מלאו את טבלת החוקים של PF2 לפי המדיניות הבאה (יש להקפיד על סדר החוקים):
8. מותר לגשת מחוץ לארגון אל שרת HS, אך רק ע"י הפרוטוקול והפורט המתאים (מהו?).
9. פרט לגופים מתחרים (אשר כתובות ה-IP שהם מתחילות ב-211), מותר לגשת מחוץ לארגון אל שרת MS, אך רק ע"י הפרוטוקול והפורטים המתאימים (מהם?).
10. יש לאפשר מענה על כל בקשה מתוך הרשת הארגונית.
11. כל השאר אסור.

(\*) גודל הטבלה אינו מעיד על מס' החוקים הדרושים.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Action | Dst. port | Src. port | Protocol | Dst. IP | Src. IP | In/Out | Rule ID |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

# POSIX File Permissions (10 נקודות)

לאחר הרצת הפקודה במערכת מסוג Linux התקבל הפלט הבא:

-r-xr-sr-x 1 charlie acct 70483 2008-01-04 22:53 accounting

-r--rw---- 1 alice acct 139008 2008-05-13 14:53 accounts

-rwxr-xr-x 1 system system 230482 1997-04-27 22:53 chmod

-rw-r--r-- 1 alice users 7072 2008-06-01 22:53 cv.txt

-r--r----- 1 bob gurus 19341 2008-06-03 13:29 exam

-r--r----- 1 alice gurus 6316 2008-06-03 16:25 solutions

* המשתמשים alice ו-bob חברים בקבוצה users
* המשתמש charlie חבר בקבוצה gurus
* האפליקציה chmod הינה כפי שמתוארת ב-manual של Unix (משתמשים יכולים להריץ את האפליקציה במידה ויש להם הרשאות לכך).
* האפליקציה accounting מאפשרת לצרף רשומה (append) לקובץ ה accounts

מלא את טבלת הרשאות הבאה עבור המשתמשים: bob, alice ו-charlie והקבצים accounts, cv.txt, exam, solutions אשר מראה עבור כל משתמש האם הוא יכול לבצע קריאה (**R**ead) , כתיבה (**W**rite) או רק צרוף של רשומה (**A**ppend) עבור כל אחד מהקבצים.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| solutions | | | exam | | | cv.txt | | | accounts | | |  |
| A | W | R | A | W | R | A | W | R | A | W | R |
|  |  |  |  |  |  |  |  |  |  |  |  | alice |
|  |  |  |  |  |  |  |  |  |  |  |  | bob |
|  |  |  |  |  |  |  |  |  |  |  |  | charlie |

(\*) לא קיים שיתוף פעולה בין משתמשים במערכת.

# POSIX File Permissions (15 נקודות)

בבנק הפעלולים עושים שימוש במערכת מחשוב מבוססת POSIX. המערכת כוללת את סוגי המשתמשים הבאים:

* עובד-בנק  (BankEmployee)
* עובד-בנק-מורשה-חתימה (PrivilegeBankEmployee)
* עובד-בנק-צוות-טכני (IT)
* מנהל המערכת (root)

כמו כן, המשתמשים השונים חברים בקבוצות כפי שמתואר בטבלה הבאה:



**הערה:** בשני הסעיפים הבאים עליכם לנקוט בעקרון Least Privileges כלומר, כל הרשאה שלא כתובה במפורש כי מותרת הינה אסורה (למעט מנהל המערכת אשר בידו הרשאת root).

מנהל המערכת הגדיר את הדרישות הבאות לניהול הסיסמאות של המשתמשים השונים:

1. כלל סיסמאות הכניסה לארגון מנוהלות בקובץ בשם passwords. קובץ זה ניתן לעריכה **אך ורק** באמצעות האפליקציות UserUpdatePassword ו- ITUpdatePassword
2. **כל משתמש במערכת** יכול לעדכן לעצמו את סיסמת הכניסה למחשב הארגוני באמצעות אפליקציית UserUpdatePassword
3. משתמש מסוג עובד-בנק-צוות-טכני יכול לעדכן את הסיסמאות של **כלל** המשתמשים באמצעות אפליקציית ITUpdatePassword

סעיף א׳ **(7 נקודות) –** לפנייך טבלה המכילה את הקבצים במערכת, עבור כל קובץ השלם את סדרת ההשראות (הכוללת הרשאות User, Group, Other , שם בעל הקובץ והקבוצה אליה שייך הקובץ) הנדרשות בכדי לעמוד בדרישות הנ״ל.



בבנק מוגדרת המדיניות הבאה לניהול העברות כספים (Transactions):

1. באפשרות כל עובד-בנק לבצע פעולות של העברות כספים (Transactions). העברת כספים זו איננה מאושרת באופן ישיר ומיוצגת במערכת ע״י קובץ אשר נמצא בתיקיית PendingTransactions. יצירה\מחיקה של קבצים ב\מ תיקיה זו אפשרית לכל עובד בנק ומתבצעת ללא כל אפליקציה ייעודית.
2. כל הפעולות המאושרות מנוהלות בתיקייה בשם ApprovedTransactions.
   1. אך ורק עובד בנק-מורשה-חתימה יכול ליצור קובץ פעולה **ישירות** אל תוך תיקיית ApprovedTransactions
   2. מחיקת קובץ **ישירות** מתיקיית ApprovedTransaction אפשרית אך ורק ליוצר קובץ הפעולה (כלומר מחיקת פעולה מתיקיה זו אפשרית אך ורק ע״י הבנקאי שייצר אותה).
   3. בנוסף, מידי לילה מריץ אחד מאנשי ה-IT את אפליקציית ,ApproveAllPendingTransactions אפליקציה זו מוודא את כלל הפעולות אשר נמצאות בתיקיית PendingTransactions. פעולות שעברו את תהליך הווידוא מאושרות ופעולות שנכשלו בתהליך הווידוא נמחקות. אישור פעולה בא לידי ביטוי במחיקה של קובץ הפעולה מתיקיית PendingTransactionsוהעברתו לתיקיית ApprovedTransaction.

סעיף ב׳ **(8 נקודות) –** לפנייך טבלה המכילה את הקבצים במערכת, עבור כל קובץ השלם את סדרת ההשראות (הכוללת הרשאות User, Group, Other , שם בעל הקובץ והקבוצה אליה שייך הקובץ) הנדרשות בכדי לעמוד בדרישות הנ״ל.



חלק מעשי (40 נק')

POSIX File Permission

# Setup

The practical part of this assignment will be conducted using a virtual machine of a Ubuntu Server 14.04.5 (32bit).

You can download the image of the server from this [link](https://www.osboxes.org/vmware-images/). You can use either VMware or VirtualBox virtualization software to mount the image file, just download the correct image for your software. The size of the file is about 250MB (zipped), after extraction it will take about 1.3GB, please ensure that you have such free space in your computer.

If you are not familiar of how to mount a virtual machine using existing image, you can use [this](https://kb.vmware.com/s/article/2010196) guide for VMware and [this](https://blogs.oracle.com/oswald/importing-a-vdi-in-virtualbox) for VirtualBox – don’t forget to select the correct operating system (i.e., Linux Ubuntu). The resources that you should allocate for the server are about 2GB of memory and 1 processor core, for the network adapter settings use NAT.

After mounting the image, you will request to enter a username and password.

The username of the VM is – osboxes

The password of the VM is – osboxes.org

Now you can start completing the task ☺.

# Overview

The purpose of this exercise is to introduce you to filesystem and network access control schemes and the "principle of least privilege" through the use of POSIX filesystem permissions.

After this exercise, you will:

* Understand the POSIX permissions structure including SUID and SGID bits.
* Understand the essence of the **sudo** utility and how to configure and use it securely.
* Be able to apply that knowledge to configure permissions in multiple scenarios, such as:
  + shared system directories
  + user home directories and private directories
  + privileged system directories
  + unprivileged temporary directories
  + editing important configuration files
  + restarting system processes
  + potential privilege escalation problems

# Practical Assignment

You are interviewing to the system administrator role in a company. You will be presented with some theoretical questions and tasks. You will need to answer the question and complete the tasks. Please follow the submission guidelines located in the end of the practical assignment.

The following section includes several permissions and file creation exercises. You are fully encouraged to use the Internet, man pages, help screens, and any other resources available to you in the execution and answering of these problems.

Please read and use these [disambiguation rules](#_Rules_for_resolving) for setting the correct permissions. Also, make sure that you test your answers - POSIX permissions are simple in theory, but in practice many combinations have counterintuitive effects!

## Home Directory Security (20 points)

https://www.isi.deterlab.net/file.php?file=/share/shared/POSIXPermissions/alert.png **Note:** Admins - members of the wheel group have sudo access. However, unless instructed to do so, use only standard UNIX ACLs - don't give user accounts sudo permissions or consider the sudo access the 'wheel' group already has. Of course, *you* need to use sudo to create the accounts, edit root files, etc... this is exactly what sudo is for.

Your server needs two home directories, the usual /home and also /admins for your team. Normal home directories are private, while the home directories in /admins will be publicly viewable and somewhat collaborative.

https://www.isi.deterlab.net/file.php?file=/share/shared/POSIXPermissions/idea.png You can test the permissions settings on server by logging in as the various accounts you've created! For example, after creating the account larry, you can log in by executing ssh larry@localhost and entering the password you used for larry. In this way, you can see if the permissions you set meet the requirements.

1. Create the /admins directory.
2. Create the groups emp and wheel
3. Create the user accounts larry, moe, and curly. You may set the passwords to anything you like. In addition, create a home directory for each user at /home/username
4. Add the three accounts you just made to the emp group. In our system, accounts in the emp group are "normal" (i.e., non-admin) accounts.
5. Next, we will set up our administrators. Create the user accounts ken, dmr, and bwk - specifying that the home directory for each admin should be located at /admins/username - where username is ken, dmr or bwk. In other words, admin homedirs are **not** in /home. You may set the passwords to anything you like.
6. Add the admin accounts to the wheel group (Ensure that admins are **not** part of the emp group.) Being in the wheel group is what gives these users administrator rights (i.e., sudo powers).
7. On this system, default permissions for new home directories allow other users (i.e., users that are not the owner or in the specified group) to read and execute files in the directory. This is too permissive for us. Set the mode on the **home directories** in /home so that owners (i.e., currly, larry and moe) have read, write, and execute privileges to their home directory , group can read and execute and other has no permissions. (Set the mode on the homedir only -- do not set it recursively.)
8. Individual home directories should now be inaccessible to other users. Now, set the permission mode on /home itself so that normal users can't list the contents of /home but can still access their home directories *and* so that members of the wheel group have full access to the directory (without using sudo).
9. By default, each homedir is *owned* by its user, and the homedir's *group* is set to the group named after the user. (For example, ken's homedir is set to ken:wheel - i.e., ken is the owner and the group is set to ken's group.) Set the permission *modes* recursively on the individual home directories in /admins (see man chmod) so that:
   * owners have *full access*
   * group users (users who are in the group associated with a user's home directory) can read and create files in that homedir
   * other users can read and execute any files (unlike the home directories in /home)
   * files created by a group member in that homedir should be set to the homedir owner's group. (Hint: Look up what the SUID and SGID bits do on directories.)

## Editing Configuration Files (4 points)

All members of the wheel group do system administration on the server. Because of this, they all have full sudo access to root privilege with the /etc/sudoers directive '%wheel ALL=(ALL) NOPASSWD: ALL'.

The "NOPASSWD" means they don't have to enter a passwd upon sudo invocation.

1. Add the above directive to /etc/sudoers to allow users from wheel group to perform system administration (i.e., run sudo). You can use the vim text editor to do so.

## The Ballot Box (8 points)

All regular employees use this directory to submit votes for "employee of the month".

1. Create the /ballots directory.
2. Set the permissions on /ballots so that it is owned by root and users can write files into the directory but cannot list the contents of the directory.
3. Furthermore, set it so that members of the wheel group have no access (not including sudo).
4. **Short Answer 1:** Is there any way that employees can read the ballots of other users? If so, what could be done to stop this? If not, what prevents them from reading them?
5. **Short Answer 2:**What does the 'x' bit mean when applied to directories

## The TPS Reports Directory (8 points)

Admin employees submit TPS reports in this partially collaborative directory.

1. Create the /tpsreports directory.
2. Create the tps user.
3. Set the permissions on /tpsreports so that it is owned by tps and that tps and members of the wheel group have full access to the directory, but so that no one else has access to the directory.
4. Furthermore, configure it so that files written into it are still owned by the wheel group, but so that one member of wheel cannot delete another member's files.
5. **Short Answer 3:** Which users on the system can delete arbitrary files in the /tpsreports directory?
6. **Short Answer 4:** Is there any way that non-wheel employees can read files in the /tpsreports directory?

# Submission Guidelines

**Theoretical parts**

1. All theoretical parts of this assignment should be uploaded in **one** document file to moodle.
2. You should upload the document **only from one student account**.
3. The name of the document should be **ID1\_ID2**.

**Practical parts**

1. The practical parts should be uploaded to an FTP server on IP address 132.72.81.33 port 21.
2. The FTP server will be available only from the **University’s network** – take it into account, **we will not accept late submissions**.
3. In order to connect to the FTP server you will first need to configurate the network interface of your virtual machine. To do so, run the following commands:
   1. *sudo ifconfig eth0 up*
   2. *sudo dhclient eth0*

(\*) You can validate that network is available by running *ping google.com*.

1. Connect to the ftp server using the following command:  
   *ftp -p 132.72.81.33 21*
   1. The server will request a user name and password
      1. username – anonymous
      2. password – your email address
   2. After connecting to the server, you should turn off the passive mode by using the following command: *passive* (if everything work well, the output will be “Passive mode off”).
2. On the FTP server, you will find two directories **hw2** and **submission-script**
   1. In the **submission-script** directory you will find a script (submit.sh), download it to your machine (to do so, use the *get* command).
   2. Run the script on your local machine and write its output to a file named ID1\_ID2\_0.
   3. Validate the output of the script (e.g., using the *vim* text editor).
   4. After validating the output of the script you can upload the file to the FTP server to **hw2** directory (to do so, use the *put* command).

**NOTE:** you cannot edit the report after uploading it to the FTP server. If some how you made a mistake, and you wish to upload a second report to the FTP server, increment the version number (i.e., ID1\_ID2\_1) – we will test the latest version, however we allow only three resubmissions, take it into account.

**Done!**

# Rules for resolving filesystem permission ambiguities:

Permissions should *always* be set to reflect the **least privilege** required to fulfill the requirements. In POSIX permissions, every bit set represents *less security*, so we want to set as few bits as possible. Resolve any ambiguities with this cumulative list of guidelines:

1. Files you are instructed to create are to be owned by root unless otherwise specified. Doing the work as root should do this automatically. (You can become root by executing sudo su -.)
   * Exception: Files in homedirs should be owned by the home directory owner (useradd should do this by default for boilerplate files like .bash\_login, etc).
2. When setting ownership of a file, set the group class to the same thing as the user (owner) class unless otherwise specified.
3. Files whose permissions you are *not* otherwise instructed to change should stay at their default.
4. For any files, whose permissions you *are* instructed to change, unspecified permissions are **always** assumed to be 0 (no access). In other words, instructions for setting file permissions implicitly include all access groups (even those not explicitly mentioned).
5. Permissions are assumed to be for all classes unless specified.
6. When granting permissions, setuid, setgid, and sticky bits are **never** granted unless specifically required to solve the problem. These bits are special and must be required by the nature of the question or be otherwise mentioned to be granted.
7. If any tasks are not possible with the standard POSIX permissions available in this exercise, explain why.

# Recommended Reading

This section contains information that will help you through the exercises in this assignment, though not mandatory, reading this section is recommended!

## The *sudo* command

Some applications simply require stronger guarantees, finer granularity, or other features that the traditional permissions cannot express. For those applications, users have many alternatives. sudo is one simple extension to traditional Unix permissions that has become very popular.

**sudo** is a setuid root application with its own ACL (stored in /etc/sudoers) that specifies, with fine granularity, tasks that users and groups can perform as root. For example, sudo could be used to allow a user to act as root in order to kill processes with a specific name. The user would otherwise have no additional privileges. This is an example of a perfect use for setuid root programs -- granting strongly-constrained privileges to unprivileged users.

However, sudo is a double-edged sword. On the one hand, it greatly enhances the expressiveness of Unix permissions without actually changing the permissions system. On the other hand, if improperly configured, it offers **easy access to root**. For more information on sudo and the sudoers ACL format, please see the manpages for sudo and sudoers (the configuration file), or other sudo-related material online (in particular regarding sudo exploits).

Beyond sudo, two broader and more revolutionary alternatives for Linux permissions include SELinux and grsecurity, while other options exist including LDAP, Novell eDirectory, and other permissions systems for other operating systems. (We won't be using any of these.)

## Software Tools (We discussed most of them on PS6)

### Add users to a system (related tools – adduser, chfn, passwd):

adduser is the tool available for adding users to the system. To create a new user, execute:

$ adduser username

adduser will copy files from the /etc/skel directory to become the new homedir of the new user in the /home/ directory. You can specify a different home directory or automatically add the new user to groups; those options can be found in the adduser manpage.

adduser does not set any finger information for the user (this is not strictly necessary anyway), but the tool chfn will do that:

$ sudo chfn jimbo

Changing finger information for jimbo.

Name []: ...

By default, the user is created with a "locked out" account with no password set. To set the password, use the command passwd:

$ sudo passwd jimbo

Changing password for user jimbo.

New Unix password:

Retype new Unix password:

passwd: all authentication tokens updated successfully.

passwd will complain if it doesn't like the password you enter, but will accept anything with enough prodding.

Once an account is created, you can log into it in a number of ways:

1. If logged in to the system where you created the account, you can execute ssh newuser@localhost to reconnect locally as the new user with the new password.
2. If local, you can also use the su command su newuser to change to that user. This does not always update all access credentials, however. This method can also be used to become root by entering sudo su -.
3. If logged in to users.create.iucc.ac.il, you can ssh to the node as the new user.

### Add groups to a system(related tool – groupadd)

groupadd adds a new group to the system with a unique ID (by default). Example:

$ groupadd newgroup

See man groupadd for more information.

### Modify a user (related tool – usermod)

usermod can be used to modify many details of a preexisting user account. For more information, see man usermod.

### Change ownership of a file (related tools – chown, chgrp)

chown stands for **ch**ange **own**ership and is (unsurprisingly) used to change the owner and group of a file.

The syntax is very simple. To change the owner of a file, execute:

$ chown newowner filename

To change the owner and group classes of a file, execute:

$ chown newowner:newgroup filename

chgrp stands for **ch**ange **gr**ou**p** and works very similarly to chown. To change the group of a file, execute:

$ chgrp newgroup filename

Recursive and other options exist; see man chown or man chgrp for more information.

### Change the mode of a file (related tool – chmod):

chmod stands for **ch**ange **mod**e and is used to change the permissions mode of a file. Earlier, we discussed how the POSIX ACL has three access classes. User (or owner), group, and other (or world). The permissions mode for each access class can be changed by the chmod command.

There are two ways to use chmod; one is an *absolute numeric mode* and the other is a *symbolic mode*.

### Absolute -- setting the permissions explicitly

In the absolute mode, chmod takes a file mode in 3 or 4 digits, each of which represents the absolute permission mode for one access class expressed in octal, where each number is a sum of the permission bits. Each permission has a unique value: read permission is 4, write permission is 2, and execute permission is 1. These values represent the position of the permission in a 3-bit value.

For example, the mode 777 means "full permissions" because all bits are set in each access class. Similarly, 000 means "no permissions" because all bits are unset in each access class.

The 3-digit mode 777 is the equivalent to the 4-digit mode 0777 where the leading 0 represents the "special" permission class of setuid, setgid, and sticky. Likewise, the modes 000 and 0000 are equivalent and represent the absence of permission. (The owner of the file and root still have the ability to change the file's mode by virtue of their ownership and superuser status.) Finally, if the 3-digit mode is used, chmod always assumes that the special access class is 0. Therefore, if you set an suid-root file to mode 777, chmod assumes that you meant mode 0777, which would take away the special permissions. This is consistent if you remember that octal modes represent *absolute permissions* and *the special group class is assumed to be 0 if a 3-digit mode is provided.*

The following is a table to help calculate permission modes:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **chmod absolute values** | | | | | | | | | | | |
| special | | | user | | | group | | | other | | |
| **s** | **g** | **t** | **r** | **w** | **x** | **r** | **w** | **x** | **r** | **w** | **x** |
| 4 | 2 | 1 | 4 | 2 | 1 | 4 | 2 | 1 | 4 | 2 | 1 |

For example, full access is 0777, which represents:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| special | | | user | | | group | | | other | | |
| **s** | **g** | **t** | **r** | **w** | **x** | **r** | **w** | **x** | **r** | **w** | **x** |
| 4 | 2 | 1 | 4 | 2 | 1 | 4 | 2 | 1 | 4 | 2 | 1 |
| 0 | | | **7** | | | **7** | | | **7** | | |

There are no special bits set, so the special octal digit is 0, while all three bits in each other class are set. Each set of bits totals 7 (4 + 2 + 1), so the mode is 0777.

Another example is mode 2755, which represents setgid (2), plus "full access" for user (4 + 2 + 1) and write and execute (4 + 1) access for both the group and other class.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| special | | | user | | | group | | | other | | |
| **s** | **g** | **t** | **r** | **w** | **x** | **r** | **w** | **x** | **r** | **w** | **x** |
| 4 | 2 | 1 | 4 | 2 | 1 | 4 | 2 | 1 | 4 | 2 | 1 |
| **2** | | | **7** | | | **5** | | | **5** | | |

Absolute modes are applied like this:

$ chmod 0755 somefile.sh

Absolute mode is great for setting things to be exactly what you want, or imposing a radically different order onto a file or directory, but it's not very good for adding the sticky bit to a file. For that kind of work (or if the octal modes just confuse you) the symbolic mode is well suited.

### Symbolic -- more user-friendly

The symbolic mode works pretty much the way you would expect. To add the execute bit to the user class, you would execute a command like this:

$ chmod u+x somefile.sh

Or to add execute to all three classes:

$ chmod ugo+x somefile.sh

To make a file setuid:

$ chmod u+s somefile.sh

To remove all permissions:

$ chmod ugo-rwxsgt somefile.sh

Some people are so put off by the absolute mode that they never learn it -- you'll find with experience that both methods of setting permissions can be expeditious depending on the situation. More information is available in the chmod manpage.

Regardless of how you set permissions, it is critical that you use the "principle of least privilege" and only grant the privileges that are necessary for proper operation.