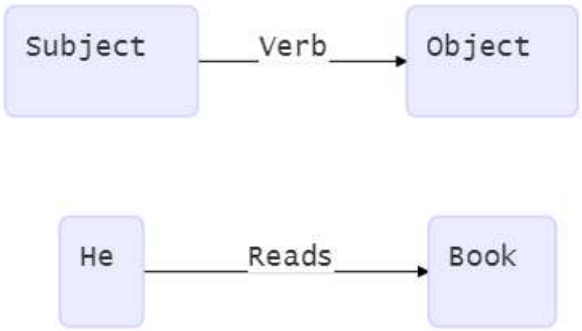
> PERMISSIONS

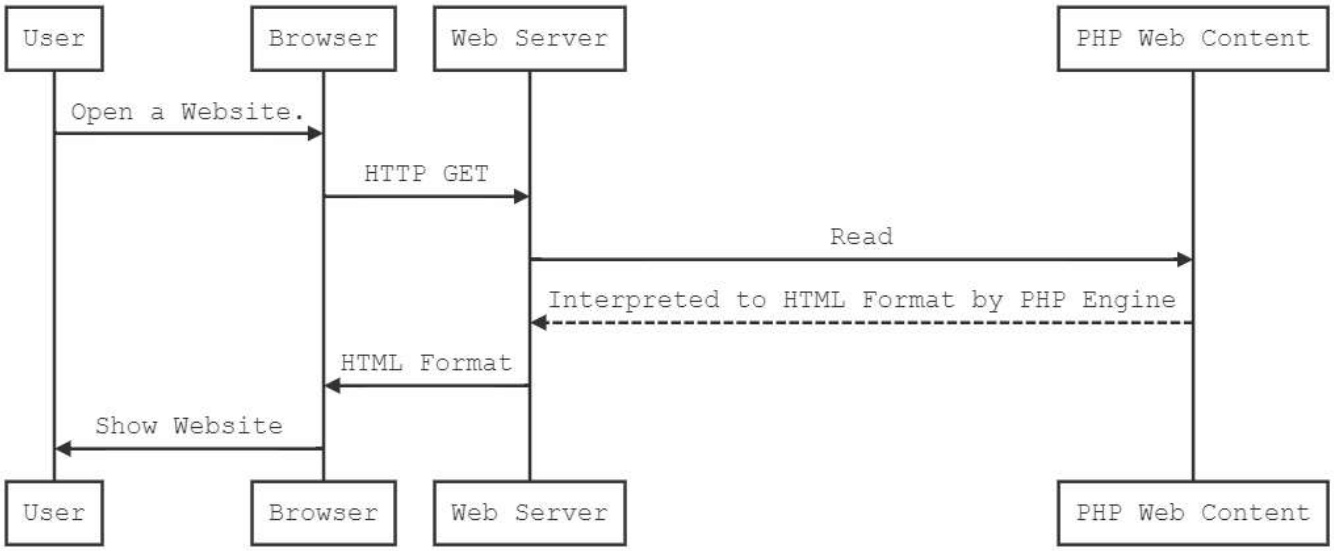
Permission ဆိုတာကတော့ အခြေခံအားဖြင့် အကြောင်းအရာတစ်ခုခုကို ခွင့်ပြုခြင်း သို့မဟုတ် ခွင့်မပြုခြင်း ကိုသတ်မှတ်ထားသော အကြောင်းအရာတစ်ခုပဲဖြစ်ပါတယ်။ Linux, Windows, MacOS, Android, iOS စသည်တို့တွင်သာမက အပြင်လောကတွင်လည်း Permission ရှိပါတယ်။ ဥပမာအားဖြင့် "အလှူခံမဝင်ရ" တို့ "အပြင်လူမဝင်ရ" တို့ ဆိုသော ဆိုင်းဘုတ်များသည်လည်း Permission ပင်ဖြစ်ပါတယ်။
Permission ကို အရိုးရှင်းဆုံးလေ့လာကြည့်ရအောင်။ အရိုးရှင်းဆုံးလေ့လာကြည့်ရင် Permission ဆိုတာ ငယ်ငယ်က သင်ခဲ့ရတဲ့ English ဝါကျ တည်ဆောက်ပုံလောက်လွယ်ပါတယ်။ အောက်က ပုံကိုကြည့်ပါ။



Subject ဆိုတဲ့သူဟာ အမြဲ ပြုလုပ်သူပါ။ Object ဆိုတာ အမြဲတမ်း အပြုလုပ်ခံရသူပါ။ "He reads book." ဆိုတဲ့ စာကြောင်းမှာလည်း "သူ သည် စာအုပ်ကို ဖတ်သည်" လို့ ဘာသာပြန်နိုင်ပါတယ်။ "သူ" က "စာအုပ်" ကို "ဖတ်" တာဖြစ်ပါတယ်။ "စာအုပ်" က "သူ" ကို "ဖတ်" တာမဟုတ်ပါ။ ဆက်လက်ပြီး တကယ့် နည်းပညာ နယ်ပယ်ထဲက ဥပမာ နဲ့ အစားထိုးကြည့်ပါမယ်။

အောက်မှာဖေါ်ပြထားတဲ့ ပုံမှာတော့ "Subject" နေရာမှာ "User or Process" ကို အစားထိုးထားပြီး "Object" နေရာမှာတော့ "File/Folder" ကို အစားထိုးထားပါတယ်။ "Verb" နေရာမှာတော့ "Read" "Write" "Execute" ဆိုပြီးအသီးသီး အစားထိုးထားပါတယ်။ System တစ်ခုမှာတော့ ပြုလုပ်သူ "Subject" ဟာ "User" ချည့်ပဲ ရှိတာမဟုတ်ဘဲ "Process"ကနေလည်း တစ်စုံတစ်ခု ပြုလုပ်နိုင်ပါသေးတယ်။ "User" တစ်ယောက်ကနေ သူကြိုက်တဲ့ File/Folder ကို Read/Write/Execute ပြုလုပ်ပုံကတော့ ရှင်းပါတယ်။တိုက်ရိုက် Access ပြုလုပ်တာပါပဲ။

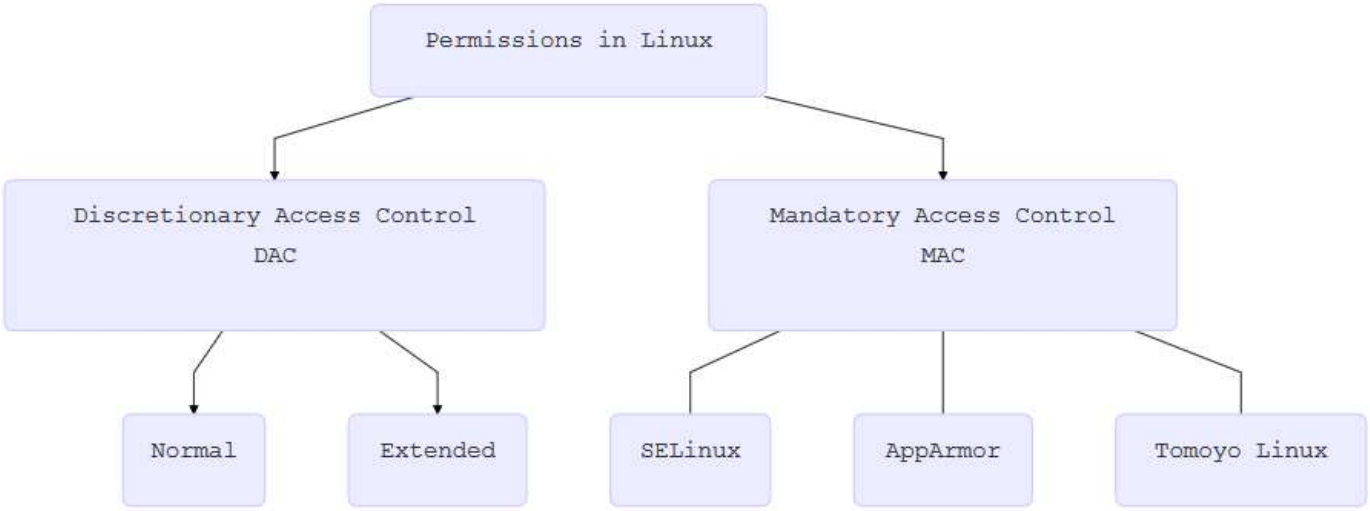
"Process" တစ်ခုကနေ File/Folder တွေကို Access ပြုလုပ်ခြင်းကိုတော့ အချို့သောသူများအနေဖြင့် နားလည်ရန် ခက်ခဲပါတယ်။ ဒါကြောင့် ဥပမာအနေနဲ့ Web Server Process ဟာ ဘယ်လို File တွေကိုဖတ်လဲ ဆိုတာ အောက်က ပုံကိုကြည့်ပါ။ (မှတ်ချက်။ Permission ကိုသာ အသားပေး ပြခြင်းဖြစ်၍ မဆိုင်သော ကိစ္စများအား ဖေါ်ပြထားခြင်းမရှိပါ။)



ဒါဆိုရင် Permission ရဲ့ အခြေခံ Concept ကို အနည်းငယ်တော့ ပိုမို မြင်လာနိုင်မယ်လို့ ယူဆပါတယ်။

Permissions in Linux

Linux မှာ ရှိတဲ့ Permission တွေကိုတော့ အကြမ်းအားဖြင့် အောက်ပါအတိုင်း ခွဲခြားထားနိုင်ပါတယ်။



Linux ရဲ့ Permission ပိုင်းမှာတော့ အဓိကအားဖြင့် ၂ ပိုင်းခွဲခြားထားနိုင်ပါတယ်။ ပထမ တစ်ပိုင်းကတော့ "Discretionary Access Control" လို့ ခေါ်တဲ့ "DAC" အပိုင်းဖြစ်ပြီး နောက်တစ်ပိုင်းကတော့ "Mandatory Access Control" လို့ခေါ်တဲ့ "MAC" Permission တွေဖြစ်ပါတယ်။ "MAC" ဆိုတာကို Network Interface Card တွေမှာပါတဲ့ Media Access Control နဲ့ မမှားကြဖို့ သတိပြုကြဖို့လိုပါတယ်။

DAC Permission တွေကိုတော့ အသေးငယ်ဆုံး Linux System များမှစ၍ အကြီးမားဆုံးသော Server Cluster ကြီး များအထိ နေရာအနှံ့မှာ တွေ့မြင်နိုင်ပါတယ်။ DAC ကို Linux တွင်သာမက အခြားသော Operating System များဖြစ်ကြတဲ့ Windows, MacOS, Unix စသည်တို့တွင်လည်း တူညီတဲ့သဘောတရားများနှင့်အညီ တွေ့မြင်နိုင်ပါတယ်။ ဒီစာအုပ်မှာတော့ ကျွန်တော့်အနေနဲ့ DAC Permission ကို လူအများနားလည်နိုင်ရန်အတွက် ၂ ပိုင်းခွဲပြီး ရေးသားပေးမှာပါ။ ပုံမှန်မြင်နေကျ တွေ့နေကျအတိုင်း DAC (Normal) နဲ့ DAC ကိုအသေးစိတ် ထပ်မံထိန်းချုပ်တဲ့ DAC - Extended ဆိုပြီး ၂ ပိုင်းရေးသားသွားမှာပဲဖြစ်ပါတယ်။

MAC Permission တွေကိုတော့ နေရာတိုင်းမှာ မတွေ့မြင်နိုင်ပါဘူး။ MAC Permission ကို အသုံးပြုထားတဲ့ System များမှာသာ တွေ့မြင်နိုင်ပါတယ်။ နောက်ပြီး MAC Permission သည် DAC Permission များမရှိဘဲ သီးခြားရပ်တည်နေခြင်းမဟုတ်ပါဘူး။ တစ်နည်းအားဖြင့်ဆိုရရင် DAC Permission တွေကို လုံးဝမသုံးဘဲ MAC ကိုချည့်ပဲ သုံးရင်လည်း အဆင်မပြေပါဘူး။ DAC Permission တွေအပေါ် ပိုမိုတိကျတဲ့ ထိန်းချုပ်မှုတွေပေးပြီး DAC Permission များနှင့်အတူ ရှိနေနိုင်တာဟာ MAC Permission တွေပါပဲ။

MAC Permission မှာတော့ Security Model တွေ အမျိုးမျိုးရှိပါတယ်။ ထင်သာမြင်သာအောင် ဥပမာ ၃ ခု ပေးထားပါတယ်။

- SELinux - US ရဲ့ National Security Agency (NSA) မှ စတင်ရေးသားပြီး 2000 ခုနှစ်ကုန်ပိုင်းလောက်မှာ OpenSource အနေနဲ့ ထုတ်ပေးခဲ့တဲ့ SELinux ကိုတော့ 2003 ခုနှစ် မှာထုတ်တဲ့ Linux Kernel Version 2.6 မှာ စတင်ထည့်သွင်းခဲ့ပါတယ်။ RedHat Enterprise Linux နဲ့ သူ့ကို အခြေခံကာ ထုတ်ထားတဲ့ Linux Distro များဖြစ်ကြတဲ့ Fedora, CentOS တွေမှာ Native အသုံးပြုကြပါတယ်။
- AppArmor - "SubDomain" ဆိုတဲ့နာမည်နဲ့ စ ခဲ့တဲ့ AppArmor ကတော့ Immunix Linux ကို Novell မှ ဝယ်ယူပြီးနောက် အဲဒီ AppArmor ကို SUSE မှ ယခုအချိန်ထိ ပိုင်ဆိုင်ထားပါတယ်။ AppArmor ကို အတွေ့များပြီး Native အနေနဲ့ အသုံးပြုကြတဲ့ Linux OS တွေကတော့ Ubuntu, Kali စတာတွေပဲ ဖြစ်ပါတယ်။
- Tomoyo Linux - NTT DATA Corporation မှ စတင်ထုတ်ဝေတဲ့ 2003 ခုနှစ်ကစပြီး ကစပြီး 2012 ခုနှစ် မတ်လအထိ Sponsor ပေးခဲ့တဲ့ Tomoyo Linux ကတော့ အပေါ်မှာ ပြောခဲ့တဲ့ SELinux ရယ် AppArmor ရယ်နဲ့လို Platform Specific မဟုတ်ပဲ 3rd Party MAC Security Model သဘောမျိုး ဖြစ်ပါတယ်။ ဒါပေမယ့် Tomoyo Linux ကတော့ AppArmor အတိုင်းပဲ File Path တွေကို လုံခြုံရေး ထိန်းချုပ်တဲ့ပုံစံမျိုးကိုသာ အသားပေးထားပါတယ်။

Discretionary Access Control (DAC)

DAC Permissions

DAC ဆိုတဲ့ Permission ကတော့ သာမန် မြင်နေကျ တွေ့နေကျအတိုင်းပါပဲ။ အခြေခံအားဖြင့် "Read" ရယ် "Write" ရယ် "Execute" ရယ် အပြင် User(Owner) နဲ့ Group Permission တွေပါတယ်။ ဒီအချက် ၅ ချက်ပေါ်ပဲ မူတည်ထားတာပါပဲ။ File သို့မဟုတ် Folder တစ်ခုမှာ DAC Permission တွေဘယ်လိုပေးထားလဲဆိုတာ အလွယ်တကူ ကြည့်ရှုချင်ရင်တော့ `ls -l` ဆိုတာလေးနဲ့ ကြည့်နိုင်ပါတယ်။

```
[root@localhost permission]# ls -l
total 8
drwxr-xr-x. 2 root root 4096 Dec 25 04:00 folder
-rw-r--r--. 1 root root  548 Dec 25 03:59 test.txt
```

CLI-1 မှာပြထားသလို မြင်ရပါလိမ့်မယ်။ `ls -l` ရိုက်ထည့်လိုက်လို့ ထွက်လာတဲ့ Output ရဲ့ ဘယ်ဘက်ဆုံး ကော်လံမှာ မြင်တွေ့ရတဲ့ `drwxr-xr-x` ဆိုတဲ့ စာလုံးလေးတွေဟာ အဲဒီ သက်ဆိုင်ရာ File/Folder ကိုပေးထားတဲ့ DAC Permission တွေပါပဲ။ အဲဒါလေးတွေကို ဘယ်လိုဖတ်ရမလဲဆိုရင်တော့ အောက်က Table လေးကို ဆက်ကြည့်ပါ။

d	r	w	x	r	w	x	r	w	x	
	4	2	1	4	2	1	4	2	1	
	Read	Write	Execute	Read	Write	Execute	Read	Write		Execute
	User(Owner)			Group			Others			

ပေါ်နေတဲ့ထဲမှာ `rwx` အစုံ ၃ စုံရှိပါတယ်။ အားလုံးပေါင်းရင် စာလုံး ၉ လုံးရှိပါတယ်။ ဘယ်ဘက်ဆုံး ၃ လုံးကတော့ အဲဒီ File ကို ပိုင်ဆိုင်တဲ့သူ (Owner) ရဲ့ ပြုလုပ်လို့ရတဲ့ Permission တွေကို ရည်ညွှန်းပါတယ်။ အလယ်က ၃ လုံးကတော့ အဲဒီ File ကိုပိုင်ဆိုင်တဲ့အဖွဲ့ (Group) ရဲ့ ပြုလုပ်လို့ရတဲ့ Permission တွေကို ရည်ညွှန်းပြီး ညာဘက်ဆုံး ၃ လုံးကတော့ Owner လည်းမဟုတ် Group ထဲလည်း မပါတဲ့ သူတွေ အားလုံးရဲ့ ပြုလုပ်လို့ရတဲ့ Permission တွေကို ရည်ညွှန်းပါတယ်။ ပိုပြီးနားလည်အောင် တစ်ဘက်စာမျက်နှာက Figure လေးကိုကြည့်ပါ။

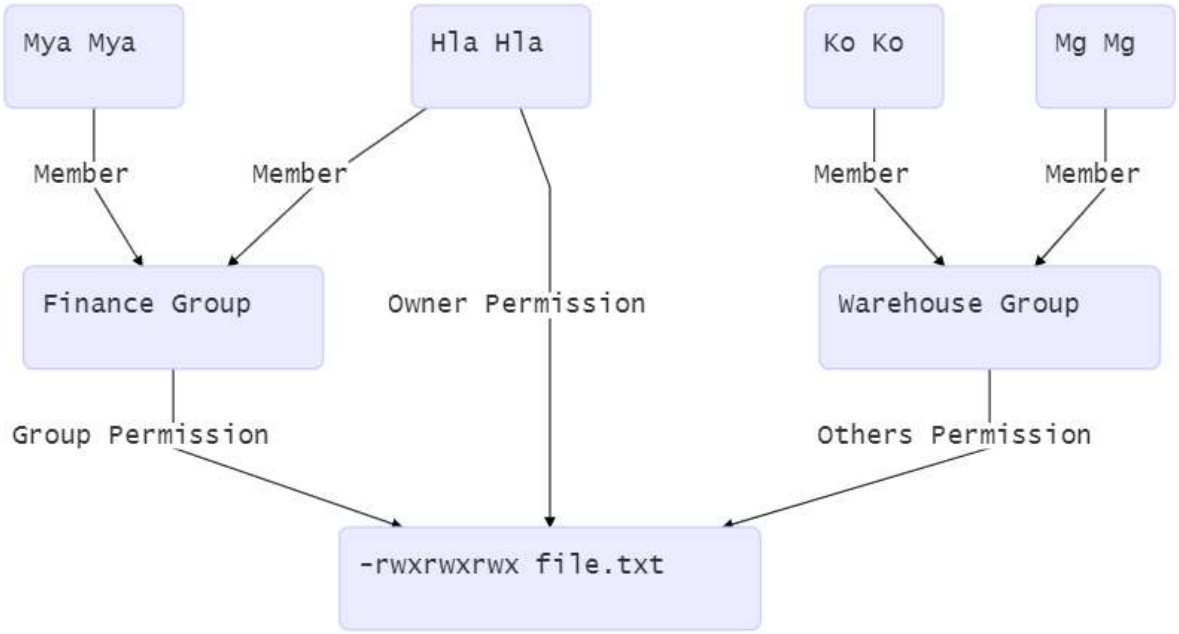


Figure မှာတော့ File.txt ဆိုတာလေးရှိပါတယ်။ အဲဒီ File လေးအတွက် Permission တွေကတော့ `-rwxrwxrwx` တွေဖြစ်ပါတယ်။ အဲဒီ File လေးရဲ့ Owner က "Hla Hla" ဆိုပါစို့။ ဒါဆိုရင် ဘယ်ဘက်ဆုံး `rwx` လေးက "Hla Hla" ဆိုတဲ့ User ကနေ အသုံးပြုလို့ရတဲ့ Permission တွေဖြစ်ပါတယ်။ file.txt လေးကို "Finance Group" က ပိုင်တယ်ဆိုရင် အဲဒီ "Finance Group" ထဲမှာ ရှိနေတဲ့ User တွေ အားလုံးဟာ အလယ်က rwx နေရာလေးမှာ ပေးထားတဲ့ Permission တွေကို အသုံးပြုလို့ရမှာပါ။ "Hla Hla" (Owner) လည်းမဟုတ် "Finance Group" ထဲလည်း မပါတဲ့ "Warehouse Group" နဲ့တကွ အဲဒီ Group ထဲမှာပါတဲ့ "Ko Ko" နဲ့ "Mg Mg" တို့လုပ်လို့ရတဲ့ Permission တွေကတော့ ညာဘက်ဆုံးက rwx လေးနဲ့ သက်ဆိုင်ပါလိမ့်မယ်။

အကယ်၍များ file.txt လေးမှာ `-rw-r--r--` ဆိုတဲ့ Permission လေးထားထားမယ်ဆိုရင် "Hla Hla" က "Read" နဲ့ "Write" လုပ်လို့ ရမှာဖြစ်ပါတယ်။ "Finance Group"ထဲမှာပါတဲ့ "Mya Mya" ကတော့ "Read" ပဲလုပ်လို့ရမှာ ဖြစ်ပြီး "Hla Hla" လည်းမဟုတ် "Finance Group" ထဲမှာလည်း မပါဝင်တဲ့ "Others" ထဲ အကျုံးဝင်တဲ့ "Warehouse Group" နဲ့တကွ အဲဒီ Group ထဲပါတဲ့ "Ko Ko" နဲ့ "Mg Mg" ကတော့ "Read" ပဲလုပ်လို့ရမှာပါ။

`-rw-r--r--` ဆိုတဲ့ Permission လေးမှာ ပထမ ၃ လုံးမှာ 'r' နဲ့ 'w' ပါဝင်တဲ့အတွက် Table 1 က သူတို့ရဲ့ Value ဇ တွအတိုင်း တွက်ကြည့်မယ်ဆိုရင် r = 4, w = 2 ဖြစ်ပါတယ်။ ၂ ခုပါဝင်တဲ့အတွက် အဲ့ ၂ ခုကို ပေါင်းလိုက်ရင် 4 + 2 = 6 ဖြစ်ပါတယ်။ အလယ်က ၃ လုံးနေရာမှာတော့ 'r' ပဲ ပါဝင်တဲ့အတွက် အလယ်နေရာအတွက် '4' ဖြစ်ပြီး ညာဘက်ဆုံးမှာလည်း 'r' ပဲ ပါဝင်တဲ့အတွက် '4' ပါပဲ။ ဒါကြောင့် `-rw-r--r--` ဆိုတဲ့ Permission ကို နံပါတ်နဲ့ ဖေါ်ပြတဲ့အခါ 644 ဆိုတာ ဖြစ်ပေါ်လာပါတယ်။ အကယ်၍ `-rwxrw-rw-` သာဖြစ်ခဲ့မယ်ဆိုရင်တော့ 755 ဖြစ်မှာပါ။ အဲဒီ Permission တွေကို ကိုယ်လိုသလို ပြောင်းလဲ ပစ်လို့ရပါတယ်။

Changing DAC Permissions

DAC Permission တွေကို ပြောင်းလဲဖို့အတွက်ကတော့ `chmod` ဆိုတဲ့ Command လေးကို အသုံးပြုနိုင်ပါတယ်။ chmod command ကို Table 1 မှာပြထားတဲ့ နံပါတ်လေးတွေနဲ့ တွဲဖက် ပေါင်းစပ်ပြီး အသုံးပြုနိုင်ပါတယ်။ အကယ်၍ file.txt လေးမှာ "Owner" ကို "Read" နဲ့ "Write" ပေးမယ် (r = 4, w = 2, r + w = 6) "Group" ကို "Read" ပေးမယ် (r = 4) "Others" တွေကိုတော့ ဘာ Permission မှမပေးဘူး (No Permission = 0) ဆိုရင်တော့ အောက်ပါ Command အတိုင်း ရေးနိုင်ပါတယ်။

```
chmod 640 file.txt
```

ပြောင်းသွားလားဆိုတာ သေချာအောင်ပြန်စစ်ချင်ရင်တော့ အောက်ပါအတိုင်းစစ်ဆေးနိုင်ပါတယ်။

```
[root@localhost permission]# ls -l file.txt
-rw-r-----. 1 root root 0 Dec 25 15:35 file.txt
```

အကယ်၍ "User/Owner" ကော "Group" ကော "Others" ကော အတွက် "Execute" များပေါင်းထည့်ချင်တယ် ဆိုရင် အထက်က ဖော်ပြထားတဲ့ 640 Permission မှာ Execute ဟာ 1 ဖြစ်တဲ့အတွက် နေရာတိုင်းကို '1' ပေါင်းလိုက်ရင် ထွက်လာတဲ့ '751' ဆိုတာက ကိုယ်လိုချင်တဲ့ Permission နံပါတ်ပါပဲ။

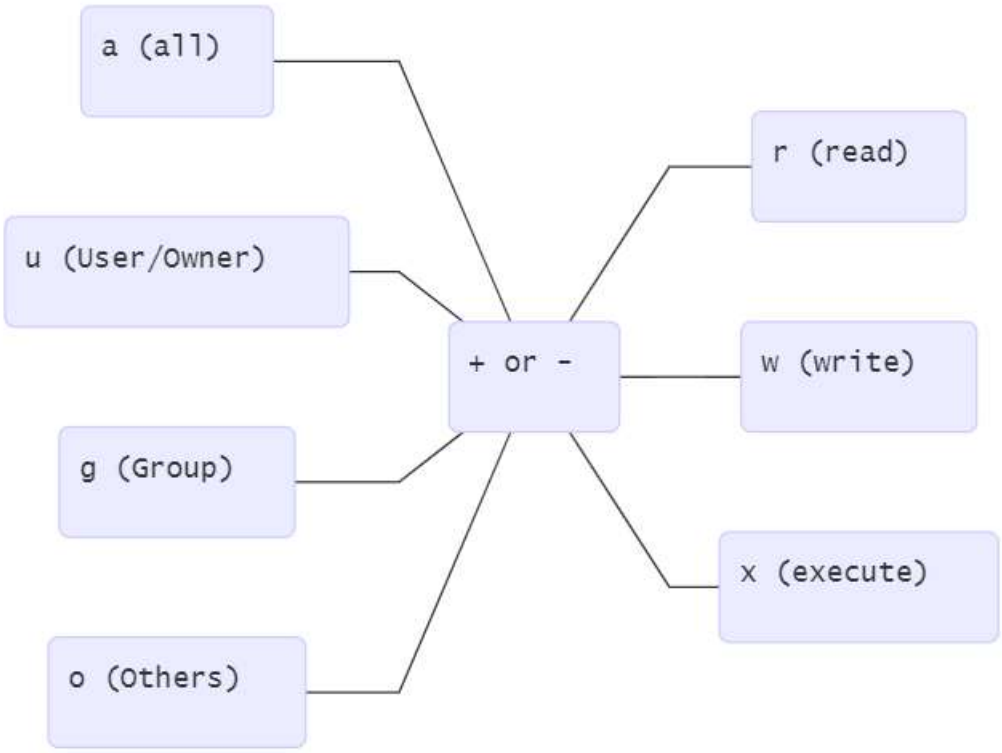
```
[root@localhost permission]# chmod 751 file.txt
[root@localhost permission]# ls -l file.txt
-rwxr-x--x. 1 root root 0 Dec 25 15:35 file.txt
```

ခုချိန်ထိ ဖော်ပြသွားတာတွေဟာ နံပါတ်များနဲ့ Permission ကို ပြောင်းလဲခြင်းပဲဖြစ်ပါတယ်။ ဒါပေမယ့် ဒီလို နံပါတ်တွေနဲ့ ပြောင်းလဲရတာ သဘောမတွေ့ဘူး (သို့မဟုတ်) တခြားနည်းလမ်းရှိလားဆိုရင် ရှိပါတယ်။ စာတွေနဲ့ပြောင်းလဲလို့လဲရပါတယ်။

အကယ်၍ file.txt မှာ `-rw-r--r--` ဆိုတဲ့ 644 Permission ရှိတယ်ဆိုပါစို့။ အဲဒီမှာ ခုနကလို နေရာတိုင်းအတွက် Execute Permission ပေါင်းထည့်ချင်တယ်ဆိုရင် ခုနကလို ကဏန်းတွေ တွက်စရာမလိုပဲ အောက်ပါ Command လေးအတိုင်း ပြောင်းလို့ရပါတယ်။

```
chmod a+x file.txt
```

ဒီ Command လေးကတော့ a+x လုပ်တာပါ။ ထပ်ရှင်းရမယ်ဆိုရင်တော့ 'a' ဆိုတာဟာ "User/Owner" ကော "Group" ကော "Others" ကော အားလုံးဖြစ်တဲ့ "All" ကိုပြောချင်တာပါ။ 'x' ဆိုတာကတော့ Execute Permission ကိုပြောချင်တာပါ။ '+' ကတော့ ပေါင်းထည့်ခိုင်းတာပဲဖြစ်ပါတယ်။ ဆိုလိုရင်းကို လိုရင်းတိုရှင်း ပြောရရင်တော့ အားလုံးကို Execute Permission ပေါင်းထည့်ပေးခြင်း ပဲဖြစ်ပါတယ်။ 'a+x' သုံးသွားတဲ့နေရာမှာ ဘာတွေထပ်သုံးလို့ရမလဲဆိုတာ အောက်က Figure ကိုကြည့်နိုင်ပါတယ်။



`chmod a+x file.txt` ဆိုတဲ့ Command ထဲက `a+x` ဆိုတဲ့ကောင်လေးနေရာမှာ ဖေါ်ပြထားသကဲ့သို့ ပြောင်းလဲထည့်သွင်းအသုံးပြုနိုင်ပါသေးတယ်။ ဥပမာအားဖြင့် 755 Permission ဖြစ်တဲ့ `-rwxr-xr-x` မှာ "Others" နဲ့ သက်ဆိုင်တဲ့ "Execute" Permission ကို ဖယ်ရှားချင်တယ်ဆိုပါက အောက်ပါ Command အတိုင်း ရိုက်နိုင်ပါတယ်။

```
chmod o-x file.txt
```

အကယ်၍များ 600 Permission ဖြစ်တဲ့ `-rw-------` တွင် "Group" နေရာအတွက် "Read" နဲ့ "Write" ထည့်ချင်ပါက အဲဒီ ၂ ခုကို အောက်ပါ Command အတိုင်း ပေါင်းရေးလို့ရပါသေးတယ်။

```
chmod g+rw file.txt
```

ထို့အတူပဲ "Group" နဲ့ "Others" တို့ကို Execute Permission ပေါင်းထည့်ပေးချင်ပါကလည်းအောက်ပါအတိုင်း ပေါင်းစပ်ရေးသားနိုင်ပါတယ်။

```
chmod go+x file.txt
```

မှတ်ချက်။ ။ Folder/Directory တစ်ခုအောက်မှာ ရှိတဲ့ File/Folder တွေ အားလုံးရဲ့ Permission တွေ အားလုံးကိုပြောင်းလဲချင်ရင်တော့ `chmod -R` အတိုင်း `-R` ဆိုတဲ့ Recursive Tag လေးထည့်ကာ ပြောင်းလဲပစ်နိုင်ပါတယ်။

Changing Ownership

File/Folder တွေကို ပိုင်ဆိုင်တဲ့ User နဲ့ Group တွေကို chown ဆိုတဲ့ Command နဲ့ ပြောင်းလဲပစ်လို့ရပါတယ်။ ဒါပေမယ့် `chown` ဆိုတဲ့ Command ကိုတော့ System ထဲမှာ ရှိတဲ့ Super user ကမှသာ အသုံးပြုနိုင်ပါတယ်။ အကြောင်းအမျိုးမျိုးကြောင့် File/Folder တွေကို ပိုင်ဆိုင်တဲ့ "Owner" တွေကို ပြောင်းလဲဖို့လိုအပ်လာပြီဆိုရင် တော့ အောက်ပါ Command Syntax လေးအတိုင်း ပြောင်းလဲပစ်နိုင်ပါတယ်။

```
chown user:group file/folder
```

အပေါ်မှာ ဖော်ပြလိုက်တာကတော့ Syntax သီးသန့်ပါပဲ။ file.txt လေးကို ownership ပိုင်းဆိုင်ရာ ကြည့်ရှု ပြောင်းလဲတာတွေ လုပ်ကြည့်ရအောင်။ စစချင်း file.txt လေးကို ဘယ်သူပိုင်လဲဆိုတာကို `ls -l` နဲ့ အရင် ထုတ်ကြည့်ပါမယ်။

```
[root@localhost permission]# ls -l
total 0
-rw-r--r--. 1 orcakrilozona testgroup 0 Dec 25 17:46 file.txt
```

file.txt လေးကို orcakrilozona ဆိုတဲ့ User နဲ့ testgroup ဆိုတဲ့ Group က ပိုင်ဆိုင်ထားတာ တွေ့ရမှာပါ။ ဆက်လက်ပြီး testuser1 ဆိုတဲ့ User ကို ပိုင်ဆိုင်မှု လွှဲပြောင်းပေးမယ်ဆိုရင် အောက်ပါအတိုင်း ပြောင်းလဲပြီး ရလဒ်ကို ကြည့်ရှုနိုင်ပါတယ်။

```
[root@localhost permission]# chown testuser1 file.txt
[root@localhost permission]# ls -l
total 0
-rw-r--r--. 1 testuser1 testgroup 0 Dec 25 17:46 file.txt
```

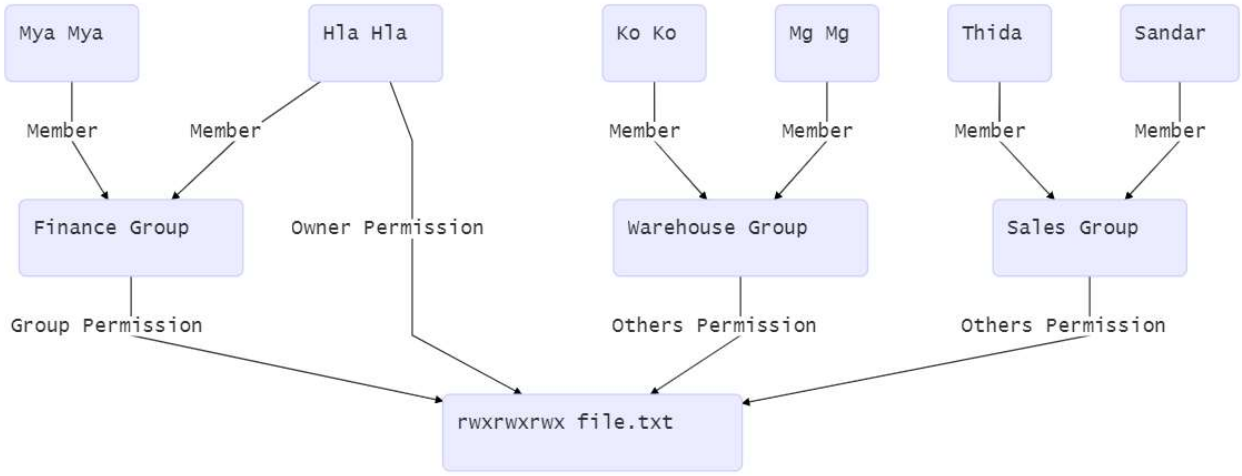
အကယ်၍များ User ကော Group ကောပါ ပြောင်းလဲချင်ရင်တော့ အောက်ပါအတိုင်း ပြောင်းလဲနိုင်ပါတယ်။

```
[root@localhost permission]# ls -l
total 0
-rw-r--r--. 1 testuser1 testgroup 0 Dec 25 17:46 file.txt
[root@localhost permission]# chown testuser2:testgroup2 file.txt
[root@localhost permission]# ls -l
total 0
-rw-r--r--. 1 testuser2 testgroup2 0 Dec 25 17:46 file.txt
```

မှတ်ချက်။ ။ Folder/Directory တစ်ခုအောက်မှာ ရှိတဲ့ File/Folder တွေ အားလုံးရဲ့ User & Group Ownership တွေ အားလုံးကိုပြောင်းလဲချင်ရင်တော့ `chown -R` အတိုင်း `-R` ဆိုတဲ့ Recursive Tag လေးထည့်ကာ ပြောင်းလဲပစ်နိုင်ပါတယ်။

Reason for DAC Normal to Extended

အထက်က Figure 1.5 မှာတော့ အရိုးရှင်းဆုံးအခြေအနေကို ဖော်ပြခဲ့ပြီးဖြစ်ပါတယ်။ ဒါပေမယ့် အောက်က Figure လိုအခြေအနေကို ဆက်လက်ကြည့်ရှုရအောင်။



အပေါ်က Figure မှာတော့ file.txt လေးကို ပိုင်ဆိုင်တဲ့ User က "Hla Hla" ဖြစ်ပြီး ပိုင်ဆိုင်တဲ့ Group ကတော့ "Finance Group" ဖြစ်ပါတယ်။ "Hla Hla" လည်းမဟုတ် "Finance Group" ထဲမှာ ပါဝင်တဲ့သူတွေလည်း မဟုတ်တဲ့ အခြားသောသူများဟာအားလုံးအပေါ်ကို "Others" Permission က သက်ရောက်မှုရှိပါတယ်။ ဒါပေမယ့် အဲဒီ File လေးကို ဥပမာအားဖြင့် "Ko Ko" နဲ့ "Sales Group" အဖွဲ့ဝင်တွေ ကတော့ ဖတ်ခွင့်ရှိစေချင်တယ် အခြားသူတွေအားလုံးကိုတော့ ဖတ်ခွင့်မရှိစေချင်ဘူးဆိုရင် ဘယ်လိုလုပ်ကြပါမလဲ? ဒါဆိုရင်တော့ ကျွန်တော် သတ်မှတ်ထားတဲ့ DAC Extended အပိုင်းကို ရောက်ရှိလာပါပြီ။ DAC Extended ကိုတော့ ACL များကို လိုသလို ထည့်ပေးခြင်းဖြင့် ထိန်းနိုင်ပါတယ်။

Access Control List (facl)

Figure 1.7 လိုအခြေအနေ ဆိုရင်တော့ Access Control List လို့ခေါ်တဲ့ ACL တွေ ထည့်သွင်းဖို့လိုအပ်လာပါပြီ။ ACL တွေသုံးခြင်းအားဖြင့် File/Folder တွေမှာ User တစ်ယောက်ချင်း Group တစ်ခုချင်းစီအတွက် Permission တွေကို အသေးစိတ် သတ်မှတ်ပေးနိုင်ပါတယ်။ စစချင်း File/Folder တွေရဲ့ ACL တွေ ဘယ်လိုရှိလဲ အထဲမှာ ဘာရေးထားလဲ ဆိုတာ အရင် ထုတ်ကြည့်ရအောင် အောက်ပါ getfacl ဆိုတဲ့ Command ကို အသုံးပြုနိုင်ပါတယ်။

```
[root@localhost permission]# ls -l
total 4
-rw-r--r--. 1 testuser2 testgroup2 12 Dec 25 18:09 file.txt
[root@localhost permission]# getfacl file.txt
# file: file.txt
# owner: testuser2
# group: testgroup2
user::rw-
group::r--
other::r--
[root@localhost permission]# cat file.txt
Hello World
```

ဒီပုံစံအတိုင်းဆိုရင် file.txt လေးကို 'testuser2' ဆိုတဲ့ User က ပိုင်ဆိုင်ပြီး 'testgroup2' ဆိုတဲ့ Group က ပိုင်ဆိုင်ထားပါတယ်။ အထဲမှာတော့ Hello World ဆိုတဲ့ စာလေးရေးထားပါတယ်။ အခြား သီးခြား Permission မရှိပါဘူး။ တစ်လက်စထဲ `chmod 640 file.txt` ပေးပြီး 'testuser2' နဲ့ 'testgroup2' ကလွဲလို့ အခြား ဘယ်သူမှ ဒီ File ကို 'Read' လုပ်လို့မရအောင် Permission ကန့်လိုက်လို့ရပါတယ်။ အဲဒီနောက်မှာတော့ 'testuser1' က ဒီ File ကို Read လုပ်လို့ရအောင် ACL တစ်ခုကို အောက်ပါ Command နဲ့ ထည့်ကြည့်ပါမယ်။

```
[root@localhost permission]# setfacl -m u:testuser1:r file.txt
[root@localhost permission]# getfacl file.txt
# file: file.txt
# owner: testuser2
# group: testgroup2
user::rw-
user:testuser1:r--
group::r--
mask::r--
other::r--
```

`setfacl -m u:testuser1:r file.txt` ဆိုတဲ့ Command မှာတော့ setfacl ဆိုတဲ့ Main Command ကို အသုံးပြုထားပြီး `-m` ဆိုတာကတော့ အဲဒီ File လေးရဲ့ ACL ကို 'Modify' လုပ်မယ်လို့ပြောချင်တာပါ။ `u:testuser1:r` ဆိုတဲ့အပိုင်းမှာတော့ ':' ၂ ခု နဲ့ခြားထားတဲ့ အပိုင်း ၃ ပိုင်းပါပါတယ်။ ပထမအပိုင်းကတော့ 'u' ပါ။ ACL ကို သတ်မှတ်တာသည် User တစ်ယောက်အပေါ်ကို သတ်မှတ်မယ်လို့ ပြောချင်တာပါ။
အကယ်၍ 'u' နေရာမှာ 'g' ဆိုရင်တော့ Group ကိုသတ်မှတ်တာဖြစ်ပြီး Group Name ကို testuser1 နေရာမှာ အစားထိုးရိုက်ထည့်ပေးရမှာပဲဖြစ်ပါတယ်။ testuser1 ကတော့ Permission ကို ပေးချင်တဲ့ username ဖြစ်ပါတယ်။ နောက်ဆုံးက 'r' လေးကတော့ အဲဒီ User ကို ပေးမယ့် Permission 'Read' ကို ဆိုလိုခြင်းဖြစ်ပါတယ်။ နောက်ဆုံး

file.txt ကတော့ အဲဒါတွေကိုသက်ရောက်စေမယ့် File နာမည်ဖြစ်ပါတယ်။ Permission တကယ်အလုပ်ဖြစ်မဖြစ် ပြန်လည် စမ်းသပ်ကြည့်ရင်တော့ အောက်ပါအတိုင်းမြင်ရမှာပါ။

```
[root@localhost permission]# su testuser1
[testuser1@localhost permission]$ cat file.txt
Hello World
[testuser1@localhost permission]$ exit
exit
[root@localhost permission]# su testuser2
[testuser2@localhost permission]$ cat file.txt
Hello World
[testuser2@localhost permission]$ exit
exit
[root@localhost permission]# su testuser3
[testuser3@localhost permission]$ cat file.txt
cat: file.txt: Permission denied
```

testuser1 ကတော့ ACL Permission အရ 'Read' ပြုလုပ်နိုင်ပါတယ်။ testuser2 ကတော့ နဂိုထဲက Owner User ဖြစ်တဲ့အတွက် 'Read' အပြင် 'Write' ပါရနေမှာပါ။ ဒါပေမယ့် ACL ထဲလည်းမပါ၊ Owner လည်းမဟုတ်၊ Owner Group ထဲလည်းမပါတဲ့ testuser3 ကတော့ 'Read' Permission တောင် Denied ပြနေတာ တွေ့ရမှာပါ။ ACL တွေကို ပြန်ဖျက်ရင်တော့ အောက်ပါအတိုင်း `setfacl -b` လေးနဲ့ ပြန်ဖျက်နိုင်ပါတယ်။

```
[root@localhost permission]# getfacl file.txt
# file: file.txt
# owner: testuser2
# group: testgroup2
user::rw-
user:testuser1:r--
group::r--
mask::r--
other::---

[root@localhost permission]# setfacl -b file.txt
[root@localhost permission]# getfacl file.txt
# file: file.txt
# owner: testuser2
# group: testgroup2
user::rw-
group::r--
other::---
```

Special Permissions (Sticky Bit, setUID bit, setGID bit)

ကြိုတင်မှတ်သားထားရန်

`Unix-Like System` များမှာ ပုံမှန်အားဖြင့် `File/Folder` များကို စတင်တည်ဆောက်သောသူများသည် အဲဒီ သက်ဆိုင်ရာ `File/Folder` များ၏ `"Owner"` များဖြစ်လာကြသည်။

File သို့မဟုတ် Folder တွေအပေါ်မှာ Special Permission တွေပေးခြင်းအားဖြင့် အချို့သော Permission များကို ထိန်းသိမ်းပေးနိုင်ပါတယ်။ Special permission တွေကို ပြောင်းလဲခြင်းအတွက် အထက်က `chmod` Command ကိုပဲ ဆက်လက် အသုံးပြုနိုင်ပါတယ်။ ဒီနေရာမှ မှတ်သားထားဖို့ကတော့ ပုံမှန်အတိုင်းဆိုရင် ကဏ်များဖြင့် Permission ပေးပါက ကဏ်း ၃ လုံးသာသုံးစွဲခဲ့သော်လည်း Special Permission များထည့်သွင်းပါက ကဏ်း ၄ လုံးဖြစ်သွားမှာဖြစ်ပါတယ်။

Sticky Bit (1)

Sticky Bit ကိုတော့ Folder/Directory ပေါ်မှာထည့်ပေးလိုက်ပါက အဲဒီ Folder/Directory ထဲမှာရှိနေတဲ့ File တွေအားလုံးကို သူတို့ရဲ့ သက်ဆိုင်ရာ "Owner" မှသာ ပြင်ဆင်နိုင်ခွင့်ရှိပါတယ်။ အခြားသူမှ ဝင်ရောက်ပြင်ဆင်ပြောင်းလဲခြင်းမျိုး ပြုလုပ်လို့မရတော့ပါဘူး။ Sticky Bit ကို ထည့်သွင်းချင်ရင်တော့ အောက်ပါ Command အတိုင်း ထည့်သွင်းနိုင်ပါတယ်။

```
[root@localhost permission]# chmod 1755 folder/
[root@localhost permission]# ls -l
total 4
drwxr-xr-t. 2 root root 4096 Dec 25 16:11 folder
```

အထက်ဖော်ပြပါ Permission နေရာမှာတော့ နောက်ဆုံးမှာ 't' အသေးလေးကိုမြင်ရပါလိမ့်မယ်။ 't' ကိုယ်စားပြုတာကတော့ အဲဒီနေရာမှာ 'Sticky Bit' ကော "Execute" Permission ကော ရှိတယ်လို့ ကိုယ်စားပြုပါတယ်။ အကယ်၍ "Execute" ကို အဲဒီနေရာ (Others) မှ ဖယ်ရှားလိုက်ပါက 'T' အကြီး ဖြစ်သွားတာကို တွေ့ရမှာပါ။

```
[root@localhost permission]# chmod o-x folder/
[root@localhost permission]# ls -l
total 4
drwxr-xr-T. 2 root root 4096 Dec 25 16:11 folder
```

စာနဲ့ ရေးသားချင်ရင်တော့ 't' လေးကို "Others" နေရာမှာ ပေါင်းခြင်း နှုတ်ခြင်းပြုလုပ်နိုင်ပါတယ်။ ယခု ထည့်သွင်းထားတဲ့ Sticky Bit လေးကို ဖယ်ရှားချင်ရင်တော့ အောက်ပါ Command အတိုင်း ရေးသားနိုင်ပါသေးတယ်။ Command ရိုက်ထည့်ပြီးရင်တော့ ပြန်လည်ကြည့်တဲ့အခါ 'T' မရှိတော့တာ တွေ့ရမှာပါ။ အကယ်၍ ပေါင်းထည့်ချင်ရင်တော့ o+t ကို အစားထိုးပြီး ရေးသားနိုင်ပါတယ်။

```
[root@localhost permission]# chmod o-t folder/
[root@localhost permission]# ls -l
total 4
drwxr-xr--. 2 root root 4096 Dec 25 16:11 folder
```

setUID Bit (4) setGID Bit (2)

setUID နဲ့ setGID Bit များကတော့ Permission များထဲက Execute ကို ပိုမိုပြီး ထိန်းချုပ်ပေးသော Special Permission များဖြစ်ကြသည်။ ပုံမှန်အားဖြင့် User တစ်ယောက်သည် Program တစ်ခုကို Run/Execute လုပ်သည့်အခါ အဲဒီ User အနေနဲ့ပြုလုပ်ကြပါတယ်။ ဒါပေမယ့် setUID Bit ရှိသော Program တစ်ခုကို Run/Execute ပြုလုပ်သည့်အခါတွင်တော့ အဲဒီ Program ကို ပိုင်ဆိုင်သော User အနေနဲ့ Run ကြပါတယ်။ setUID Bit ကို ထည့်သွင်းချင်ရင်တော့ အောက်ပါအတိုင်းထည့်သွင်းနိုင်ပါတယ်။

```
chmod 4755 program
```

သို့မဟုတ်

```
chmod u+s program
```

setGID Bit ကို ထည့်သွင်းချင်ရင်တော့ အောက်ပါအတိုင်း ထည့်သွင်းနိုင်ပါတယ်။

```
chmod 2755 program
```

သို့မဟုတ်

```
chmod g+s program
```

setUID Bit ကို အထင်ရှားဆုံးပြသနိုင်သော File ကတော့ /bin/passwd ဆိုတဲ့ File ပဲဖြစ်ပါတယ်။ သူ့ရဲ့ Permission အနေအထား ဘယ်လိုရှိလဲဆိုတာ ထုတ်ကြည့်ရအောင်။

```
[root@localhost permission]# ls -l /bin/passwd
-rwsr-xr-x. 1 root root 29008 Apr 12  2018 /bin/passwd
```

ဘယ်ဘက်ဆုံးက User/Owner နေရာရဲ့ ပုံမှန်ဆိုရင် 'x' ပေါ်ရမယ့်နေရာမှာ 's' လေးကို တွေ့ပါလိမ့်မယ်။ အဲဒါလေးကတော့ အဲဒီ File မှာ "setUID Bit" နဲ့တကွ "Execute" Permission ပါ ရှိနေတယ်ဆိုတာကို ဖေါ်ပြနေတာပါ။ "Execute" Permission မရှိပါက 's' အသေးအစား 'S' အကြီးပေါ်နေပါလိမ့်မယ်။ ဖေါ်ပြထားတဲ့ /bin/passwd ဆိုတာကတော့ User ရဲ့ Password ကိုပြောင်းလဲတဲ့ Command ပဲဖြစ်ပါတယ်။ သာမန် User တစ်ယောက်ကနေ Password ပြောင်းလဲတဲ့အခါ သူပြောင်းလဲလိုက်တဲ့ Password ကို မှတ်သားထားတဲ့ နေရာတွေမှာ ပြင်ဆင်ဖို့အတွက် သာမန် User ရဲ့ Permission Right တွေနဲ့ အလုပ်မဖြစ်ပါဘူး။

ဒါကြောင့်အဲဒီ သာမန် User လေးက Permission Right မရှိတဲ့နေရာတွေကိုပါ သက်ရောက်မှုရှိအောင် `/bin/passwd` ဆိုတဲ့ setUID Bit ရှိတဲ့ File ကို စတင် Execute လုပ်တော့သာ သာမန် User အနေနဲ့ စတင် Execute တာဖြစ်ပြီး တကယ်အလုပ်လုပ်တဲ့အချိန်မှာ အဲဒီ File လေးကို ပိုင်ဆိုင်တဲ့ 'root' user အနေနဲ့ run သွားပါတယ်။ ကျွန်တော်ကသာ စာထဲမှာ ဒီလိုရေးနေတာ တကယ်ဟုတ်လားမဟုတ်လားဆိုတာ သေချာအောင် စမ်းသပ်ကြည့်ရအောင်။

C Language နဲ့ရေးထားတဲ့ Program တစ်ခုကို Compile ပြုလုပ်ပြီး စမ်းသပ်ပါမယ်။ အဲဒီ Program လေး ထုတ်ပြမှာကတော့ အဲဒီ Program ကို Compile လုပ်ပြီး Run တဲ့အခါမှာ Run တဲ့ User ရဲ့ ID နဲ့ အမှန်တကယ် Permission သက်ရောက်တဲ့ User ရဲ့ ID ကိုထုတ်ပြပေးမှာပါ။ အဲဒီ Program ရဲ့ ရိုးရှင်းတဲ့ Source Code လေးကို ကိုယ်တိုင်ပြန်လည်စမ်းသပ်နိုင်အောင် အောက်မှာ ဖေါ်ပြလိုက်ပါတယ်။

```
#include <stdio.h>
#include <unistd.h>
 int main () {
  int real = getuid();
  int euid = geteuid();
  printf("The REAL UID =: %d\n", real);
  printf("The EFFECTIVE UID =: %d\n", euid);
}
```

အောက်ပါ Command နဲ့ အဲဒီ Source Code လေးကို Compile ပြုလုပ်ရအောင်။ နောက်ပြီး Compile လုပ်ပြီး နောက် ထွက်လာတဲ့ File လေးရဲ့ Default Permission ဘယ်လိုရှိမလဲဆိုတာ တခါထဲ ကြိုတင် ကြည့်ရှုပါမယ်။

```
[root@localhost permission]# gcc -o setuid setuid.c
[root@localhost permission]# ls -l setuid
-rwxr-xr-x. 1 root root 8368 Dec 25 16:40 setuid
```

"setUID Bit" များမထည့်ခင်မှာ 'root' User နဲ့ကော သာမန် User နဲ့ကော ၂ ခုစလုံးနဲ့ Run ကြည့်ပါမယ်။

```
[root@localhost permission]# ./setuid
The REAL UID =: 0
The EFFECTIVE UID =: 0
[root@localhost permission]# id -u
0
[root@localhost permission]# exit
exit
[orcakrilozona@localhost permission]$ ./setuid
The REAL UID =: 1000
The EFFECTIVE UID =: 1000
[orcakrilozona@localhost permission]$ id -u
1000
```

ပထမတစ်ခေါက် 'root' User နဲ့ Run တဲ့အခါ root user ရဲ့ ID က 0 (zero) ဖြစ်တဲ့အတွက် အားလုံး 0 (zero) နဲ့ ပြသသွားတာ တွေ့ရမှာဖြစ်ပြီး ဒုတိယတစ်ခေါက် သာမန် User ဖြစ်တဲ့ 'orcakrilozona' နဲ့ Run တဲ့အခါမှာ သူ့ရဲ့ ID ဖြစ်တဲ့ 1000 နဲ့ အားလုံးကိုပြသ သွားတာဖြစ်ပါတယ်။ 'setUID Bit' မထည့်သွင်းထားရသေးခင်မှာ id 0 နဲ့

User က Run ရင်လည်း id 0 နဲ့ပဲ သက်ရောက်မှုရှိသလို id 1000 နဲ့ Run ရင်လည်း id 1000 နဲ့ပဲ သက်ရောက်မှု ရှိနေပါတယ်။ ဆက်လက်ပြီး 'setUID Bit' ထည့်သွင်းကြည့်ပြီး Run ကြည့်ပါမယ်။

```
[root@localhost permission]# chmod 4755 setuid
[root@localhost permission]# ls -l setuid
-rwsr-xr-x. 1 root root 8368 Dec 25 16:40 setuid
[root@localhost permission]# ./setuid
The REAL UID =: 0
The EFFECTIVE UID =: 0
[root@localhost permission]# id -u
0
[root@localhost permission]# exit
exit
[orcakrilozona@localhost permission]$ ./setuid
The REAL UID =: 1000
The EFFECTIVE UID =: 0
[orcakrilozona@localhost permission]$ id -u
1000
```

ခုဆိုရင်တော့ ပြောင်းလဲ သွားတာတွေ သတိထားမိမှာပါ။ သာမန် User ဖြစ်တဲ့ 'orcakrilozona' အနေနဲ့ Execute ပြုလုပ်တဲ့အခါ Execute လုပ်တဲ့ User ID က 1000 ဖြစ်ပေမယ့် Permission ကို အမှန်တကယ်သက်ရောက်တဲ့ User ID ကတော့ 0 (zero) [root user] ဖြစ်တယ်ဆိုတာ မြင်တွေ့ရမှာပါ။ setGID Bit သည်လည်း User နေရာမှာ Group ပြောင်းသွားတာကလွဲရင် ဒီသဘောပဲဖြစ်တဲ့အတွက် တွဲဖက် မှတ်သားနိုင်ပါတယ်။

DAC Permission တွေ နဲ့တင်တော်တော်လုပ်လို့ရနေပြီ။ ဒါပေမယ့် လုံလောက်မှု ရှိရဲ့လား?

ပြောရရင်တော့ Linux Security ပိုင်းမှာတစ်ပိုင်းဖြစ်တဲ့ DAC Permission မှာ အကြီးမားဆုံး ယိုပေါက်ကြီးတစ်ခု ရှိနေပါတယ်။ ဒါပေမယ့် အဲဒီ အကြီးမားဆုံး လုံခြုံရေး ယိုပေါက်ကြီးကလည်း DAC Permission ရဲ့ Feature တစ်ခု ဖြစ်နေပါသေးတယ်။ အဲဒါကတော့ Owner User တွေဟာ သူတို့ ပိုင်ဆိုင်တဲ့ File/Folder တွေရဲ့ Permission ကို သူတို့ကြိုက်သလို ပြောင်းလဲလို့ရနေပါတယ်။ ပုံမှန်အားဖြင့်တော့ မသိသာပေမယ့် "Production Environment" မှာတော့ အကယ်၍များ Permission ကို မှားယွင်းပေးမိတာပဲဖြစ်ဖြစ်၊ လိုအပ်တာထက် ပိုပေးမိတာပဲဖြစ်ဖြစ် ပြုလုပ်မိပါက မသမာတဲ့ User သို့မဟုတ် Hacker မှ System တစ်ခုလုံးကို ထိုးဖောက်ဝင်ရောက်နိုင်တဲ့အထိ လုံခြုံရေးယိုပေါက် ဖြစ်နေနိုင်သေးတာပါပဲ။ ဒါကြောင့် DAC Permission တွေအပေါ်မှာ သူတို့နဲ့အတူတကွ Permission များကို ထိန်းကျောင်းပေးမယ့် Mandatory Access Control (MAC) လို့ခေါ်တဲ့ Permission များ လိုအပ်လာပါတော့တယ်။

Mandatory Access Control (MAC)

Mandatory Access Contrl (MAC) တွေကတော့ အစောတုန်းက ပြောထားခဲ့သလိုပဲ DAC Permission တွေနဲ့ တွဲဖက် ပေါင်းစပ်ပြီး System မှာ ရှိတဲ့ Permission များကို ထိန်းသိမ်းပေးပါတယ်။ ထို့အပြင် MAC Permission များသုံးခြင်းဖြင့် DAC Permission များဖြင့် ထိန်းချုပ်လို့မရသော သို့မဟုတ် ထိန်းချုပ်ရန်ခက်ခဲသော Permission များကိုလည်း ပိုမို အသေးစိတ် ထိန်းသိမ်းပေးနိုင်ပြီး System တစ်ခုလုံးရဲ့ လုံခြုံရေးကို အတော်လေး ပိုမိုကောင်းမွန်စေပါတယ်။

MAC Permission ကိုအသုံးပြုလို့ရတဲ့ Program များထဲတွင်တော့ အမျိုးအစား များစွာတွေ့ရှိရမှာဖြစ်ပြီး နာမည်ကြီးတဲ့ MAC Permission ထိန်းချုပ်ပေးတဲ့ Program များထဲမှာတော့ 'SELinux' 'AppArmor' 'SMACK' 'Tomoyo Linux' ကဲ့သို့သော Security Model များ ပါဝင်ပါတယ်။ 'SELinux' ကော 'AppArmor' ကော ၂ ခုစလုံးကတော့ 'Linux Security Module (LSM)' System တွေဖြစ်ကြပြီး ၂ ခုစလုံးဟာ Linux Kernel Security Module တွေဖြစ်ကြပါတယ်။ အထက်မှာပြောခဲ့သလိုပဲ AppArmor ကိုတော့ SUSE, Debian, Ubuntu, Kali စ တဲ့ Linux Distribution တွေမှာတွေ့မြင်ရမှာဖြစ်ပြီး SELinux ကိုတော့ RedHat Enterprise Linux, CentOS, Fedora ကဲ့သို့သော Linux Distribution တွေမှာ အဓိက မြင်တွေ့ရမှာပါ။

Difference between AppArmor & SELinux

'AppArmor' နဲ့ 'SELinux' နဲ့ ဘာတွေဘယ်လို ကွဲပြားချက်တွေရှိလဲဆိုတာ သိရှိထားဖို့လိုပါတယ်။ ဒါမှ ဘယ်အရာကို အသုံးပြုမလဲဆိုတာ ကိုယ်တိုင် ရွေးချယ်လို့ရမှာဖြစ်ပါတယ်။ အဓိက ကွဲပြားတဲ့အချက်တွေကတော့ အောက်ပါအတိုင်းဖြစ်ပါတယ်။

- AppArmor ဟာ SELinux ထက် နားလည်ရ လွယ်ကူတယ် လို့ဆိုကြပါတယ်။ အကျိုးဆက်ကတော့ SELinux ဟာ ခက်ခဲတဲ့အတွက် Administrator များအဖို့ Setup ပြုလုပ်ခြင်း ဆက်လက်ထိန်းသိမ်းခြင်းများ ပြုလုပ်ရာမှာ ခက်ခဲမှုတွေ ဆက်လက်ဖြစ်ပေါ်နေတာ များပါတယ်။
- AppArmor ဟာ File System Object တွေကို Path များအနေနဲ့မှတ်သားထားပါတယ်။ SELinux ကတော့ inode များအနေနဲ့မှတ်သားထားပါတယ်။ ဒါကြောင့် Access လုပ်လို့မရတဲ့ File တစ်ခုကို အကယ်၍များ Hard Link ပြုလုပ်လိုက်တဲ့အခါမှာတော့ AppArmor မှာ မတူညီတဲ့ Path ဖြစ်လာတဲ့အတွက် Access လုပ်လို့ရသွားနိုင်ပေမယ့် SELinux ကတော့ inode အတူတူပဲ ဖြစ်နေတဲ့အတွက် Access လုပ်လို့မရတာက မရတာပါပဲ။
- AppArmor မှာ Multi-Level Security နဲ့ပတ်သက်ပြီးဖေါ်ပြထားခြင်းမရှိတဲ့အတွက် Bell-La Padula Model (BLP) တို့ Biba Integrity Model (Biba) တို့ကိုထောက်ပံ့ပေးနိုင်ခြင်းမရှိပါဘူး။
- အခြားသော ကွဲပြားချက် များစွာရှိပါသေးတယ်။

ဆက်လက်ပြီး ကျွန်တော့်အနေနဲ့ MAC Permission တွေကိုထိန်းသိမ်းမယ့် Security Model ကိုတော့ SELinux နဲ့ဆက်လက်ပြီး ရှင်းပြသွားပါမယ်။

Security Enhanced Linux (SELinux)

SELinux ကိုတော့ ကနဦးပိုင်းမှာ အမေရိကန် အမျိုးသားလုံခြုံရေးအေဂျင်စီ (NSA) မှစတင်ရေးသားခဲ့ပါတယ်။ 2000 ခုနှစ် ဒီဇင်ဘာလထဲမှာတော့ Open Source အဖြစ် ဖြန့်ဝေခဲ့ပြီး 2003 ခုနှစ် မတ်လထဲမှာတော့ Linux Kernel Version 2.6 မှာ စတင် အသုံးပြုခဲ့ပါတယ်။ https://github.com/SELinuxProject/ ကတော့ လက်ရှိ SELinux ရဲ့ Source Code တွေကို ထားရှိတဲ့ Repository တွေရှိပါတယ်။

What is NOT SELinux?

ကြုံတွေ့ရသလောက် SELinux ကို ထိတွေ့ဖူးတဲ့အပိုင်းပေါ်မူတည်ပြီး အမျိုးမျိုးသုံးသပ် ပြောဆိုနေတာတွေ ရှိတဲ့အတွက် "SELinux ဟာ ဘာလဲ"ဆိုတာမပြောခင်မှာ "SELinux ဟာ ဘာမဟုတ်ဘူးလဲ"ဆိုတာ ပြောဖို့ လိုအပ်ပါတယ်။

- SELinux သည် Firewall တစ်ခုမဟုတ်ပါဘူး။ SELinux က Application/Service/User များ Network Interface/Port များအကြားတွင် Permission ကန့်သတ်ချက်တွေ လုပ်နိုင်ပေမယ့် SELinux ဟာ Firewall တစ်ခုမဟုတ်ပါဘူး။

- SELinux ဟာ Anti-Virus တစ်ခုမဟုတ်ပါဘူး။ SELinux ထဲမှာ Virus Scan စစ်တဲ့ Engine မပါဝင်သလို Virus/Malicious Code တွေကိုလည်း စစ်ဆေးမပေးပါဘူး။ Application/Service/User များနှင့် သူတို့ အသုံးပြုမယ့် အရာများကြားမှာ လုပ်ပိုင်ခွင့် Permission များကို သတ်မှတ် ကန့်သတ်ပေးတာပဲ လုပ်ပါတယ်။

- SELinux ဟာ All-In-One Security Solution တစ်ခုမဟုတ်ပါဘူး။ ဥပမာအားဖြင့် Backdoor ပြုလုပ်ခြင်းများကို ကာကွယ်ပေးနိုင်ပြီး Buffer-Overflow Attack များကို လျော့ကျစေနိုင်ပေမယ့် SELinux ကို အသုံးပြုခြင်းဟာ သက်ဆိုင်ရာ Security ယိုပေါက်တွေကို အလိုလျောက် ပြင်ဆင်ပေးတဲ့ Tool မဟုတ်ပါ။

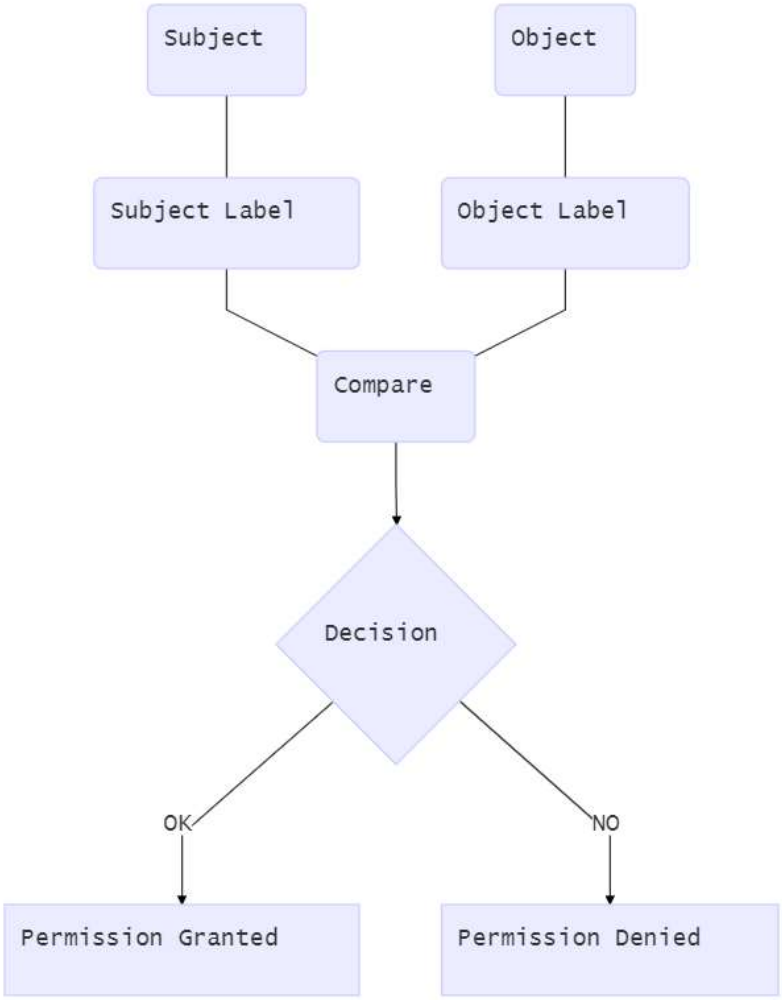
- SELinux ဟာ Security Audit/Fix Tool တစ်ခုမဟုတ်ပါဘူး။ ဆိုလိုတာက Security ယိုပေါက်များ ဖြစ်ပေါ်နေတဲ့ Program တစ်ခုရှိရင် အဲဒီ Program ထဲက Code တွေကို ပြင်ဆင်ပေးတဲ့ Tool တစ်ခု မဟုတ်ပဲ အဲဒီ လုံခြုံရေးယိုပေါက်ရှိတဲ့ Program ကိုအသုံးချပြီး System ကို ဆက်လက် တိုက်ခိုက်မှုတွေကို Permission ပိုင်းအရ ကာကွယ်ပေးတဲ့ Program တစ်ခုပဲဖြစ်ပါတယ်။

What is SELinux?

SELinux ကို အမျိုးမျိုးပြောကြပေမယ့် သေချာကြည့်ရင်တော့ သူ့ကို "Labeling Engine" လို့ပဲ ခေါ်ရမှာပါ။ SELinux ရဲ့ အဓိက အလုပ်တစ်ခုဟာ System ထဲက ရှိရှိသမျှနေရာတွေ အားလုံးကို Label လိုက်ထိုးပေးပါတယ်။ ရှိရှိသမျှ နေရာတွေဆိုတာ အောက်ပါ နေရာများအပါအဝင်အားလုံးနီးပါးပါပဲ။

- Users
- Processes & Services
- Files & Directories
- Network Interfaces
- Network Ports
- Firewall Rules

အဲဒီ Label လေးတွေကို "SELinux Label" သို့မဟုတ် "Security Context" လို့ခေါ်နိုင်ပါတယ်။ ထိုကဲ့သို့ နေရာတိုင်းနီးပါးမှာ Label တွေ ထိုးပေးပြီးတဲ့အခါမှာတော့ အဲဒီ Label တွေကို ကြည့်ကာ အချင်းချင်း နှိုင်းယှဉ်ပြီး သူ့ရဲ့ Security Policy အတိုင်း Permission ကို ကန့်သတ်ထိန်းသိမ်းတာတွေကို ဆက်လက် လုပ်ဆောင်ပေးပါတယ်။ ပိုပြီးရှင်းအောင်ပြောရရင် အစပိုင်းမှာပြောခဲ့တဲ့ "Subject" လို့ခေါ်တဲ့ လုပ်တဲ့လူရဲ့ "Security Context" တွေနဲ့ "Object" လို့ခေါ်တဲ့ ခံရတဲ့ အရာတွေမှာ ရှိတဲ့ "Security Context" တွေကို နှိုင်းယှဉ်ပြီး ဘယ်အရာတွေပြုလုပ် လို့ရလဲဆိုတာကို ထိန်းချုပ်ပေးပါတယ်။ ဥပမာအားဖြင့် Apache Server ရဲ့ Process မှာ ရှိတဲ့ "Security Context" နဲ့ File/Folder တွေမှာ ရှိတဲ့ "Security Context" နဲ့ကို နှိုင်းယှဉ်ပြီး Apache Process ကနေ သက်ဆိုင်ရာ File/Folder တွေကို "Read" "Write" "Execute" ထဲက မည်သည့် Access ရရှိမယ် မရရှိဘူးဆိုတာကို ထိန်းချုပ်ပေးနိုင်ပါတယ်။ မျက်စိထဲ ပိုမို လွယ်ကူစွာ မြင်ပြီး နားလည်နိုင်အောင် အောက်က Diagram ကိုကြည့်ရှုနိုင်ပါတယ်။



SELinux Security Context

SELinux ရဲ့ Security Context တွေကို ကြည့်ရှုချင်တယ်ဆိုရင်တော့ ဘယ်ဟာရဲ့ Security Context ကို ကြည့်မလဲဆိုတာပေါ်မူတည်ပြီး Command တွေ အမျိုးမျိုးရှိပါတယ်။
File/Folder တွေရဲ့ Context ကိုကြည့်ချင်ရင်တော့ အောက်ပါအတိုင်းကြည့်နိုင်ပါတယ်။

```
[root@localhost permission]# ls -lZ
total 4
-rw-r--r--. 1 root root unconfined_u:object_r:usr_t:s0    0 Dec 26 03:05 file.txt
drwxr-xr-x. 2 root root unconfined_u:object_r:usr_t:s0 4096 Dec 26 03:05 folder
```

Process တွေရဲ့ Context တွေကိုကြည့်ချင်ရင်တော့ အောက်ပါအတိုင်းကြည့်နိုင်ပါတယ်။

```
[root@localhost permission]# ps -Z
LABEL                             PID TTY          TIME CMD
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 3642 pts/1 00:00:00 su
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 3654 pts/1 00:00:00 bash
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 4322 pts/1 00:00:00 ps
```

User တွေရဲ့ Context တွေကိုကြည့်ချင်ရင်တော့ အောက်ပါအတိုင်းကြည့်နိုင်ပါတယ်။

```
[root@localhost permission]# id -Z
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

အခြားအရာတွေကို ကြည့်ရှုထိန်းသိမ်းတဲ့ Command တွေလည်းရှိပါသေးတယ်။ များသောအားဖြင့် သက်ဆိုင်ရာ Command တွေရဲ့နောက်မှာ -Z လိုက်ပါတယ်။ ပေါ်လာတဲ့ထဲမှာ အားလုံး `unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023` ဆိုတဲ့ အရာတွေကို သတိထားမိမှာပါ။ အဲဒါတွေကတော့ SELinux ရဲ့ Security Context သို့မဟုတ် Label တွေပဲဖြစ်ပါတယ်။ အဲဒီ Security Context မှာတော့ ':' ၄ ခုနဲ့ ခြားထားတဲ့ အပိုင်း ၅ ပိုင်းပါရှိပါတယ်။ ဒါပေမယ့် နောက်ဆုံးက `s0-s0:c0.c1023` ဆိုတာကိုတော့ "Security Level" ကိုသတ်မှတ်တဲ့အပိုင်း ဆိုပြီး ၁ ပိုင်းထဲအနေနဲ့ သတ်မှတ်နိုင်ပါတယ်။ ရံဖန်ရံခါမှာ `c0.c1023` ကိုဖေါ်ပြလေ့မရှိပဲ ဖျောက်ထားတတ်တာမို့ SELinux ရဲ့ Security Context မှာ အပိုင်း ၄ ပိုင်းရှိတယ်လို့ပဲ သတ်မှတ်ရမှာပါ။ အဲဒီအပိုင်း ၄ ပိုင်းကိုတော့ အောက်ပါအတိုင်း နားလည်ထားနိုင်ပါတယ်။

unconfined_u	**unconfined_r**	**unconfined_t**	**s0-s0:c0-c**1023
SELinux User	SELinux Role	SELinux Type	Security Level

- `unconfined_u` ကတော့ SELinux ရဲ့ User ဖြစ်ပါတယ်။ Linux ရဲ့ System ကိုရည်ညွှန်းခြင်းမဟုတ်ပါဘူး။ "Unconfined" ဆိုတာ "Confined" ဆိုတဲ့စကားလုံးကို ရှေ့မှာ "Un-" ကို Prefix ကပ်ထားတာပါ။ Confine ဆိုတာကတော့ ချုပ်ကိုင်ထားတယ် လို့ အဓိပ္ပါယ်ရတဲ့အတွက် "Unconfied" ဆိုတာကတော့ မချုပ်ကိုင်ထားဘူးလို့ ဘာသာပြန်ကြည့်နိုင်ပါတယ်။ ဒါကြောင့် ဒီလို Label ရှိတဲ့ User လုပ်သမျှကိုတော့ SELinux ကနေ ချုပ်ကိုင်ထားခြင်း မရှိပါဘူး။

- `unconfined_r` ကတော့ SELinux ရဲ့ Role ဖြစ်ပါတယ်။ SELinux ရဲ့ Role ကတော့ SELinux ရဲ့ User နဲ့ ပတ်သက်ဆက်နွယ်မှုရှိပါတယ်။ SELinux ရဲ့ User က ဘာတွေလုပ်လို့ရလဲဆိုတဲ့ လုပ်ပိုင်ခွင့်တွေဟာ SELinux ရဲ့ Role ကနေ တစ်ဆင့် သတ်မှတ်ထားတာတွေ ရှိပါတယ်။ Role-Based Access Control လို့ ခေါ်တဲ့ RBAC ရဲ့ အစိတ်အပိုင်းတစ်ခုလည်းဖြစ်ပါတယ်။

- `unconfined_t` ကတော့ SELinux က သတ်မှတ်ပေးတဲ့ Type ကို ကိုယ်စားပြုပါတယ်။ File/Folder အမျိုးအစား၊ Process အမျိုးအစား၊ User အမျိုးအစား၊ Network Port အမျိုးအစား စသည်ဖြင့် အမျိုးမျိုးရှိပါတယ်။ SELinux ရဲ့ အလွယ်ကူဆုံး Security Policy ဖြစ်တဲ့ "Type Enforcement (TE)" မှာတော့ ဒီ Type တွေကို အဓိက နှိုင်းယှဉ်ပါတယ်။

- `s0-s0:c0-c1023` ဆိုတာကတော့ SELinux ရဲ့ အခက်ခဲဆုံးအပိုင်းဖြစ်တဲ့ Multi-Level Security (MLS) နဲ့ Multi-Category Security (MCS) အပိုင်းတွေကို ထိန်းချုပ်ရင် အသုံးပြုပါတယ်။ ဒီ နေရာမှာတော့ 's' နဲ့ 'c' ဆိုပြီး နှစ်မျိုးကွဲပါတယ်။ 's' ကိုတော့ "Sensitivity Level" လို့ခေါ်ပြီး 'c' ကိုတော့ "Cateogry Level" လို့

ခေါ်ပါတယ်။ 's' မှာတော့ '0' မှ '15' အထိ စုစုပေါင်း Sensitivity Level 16 ခုရှိနိုင်ပြီး 'c' မှာတော့ '0' မှ '1023' အထိ စုစုပေါင်း Category Level 1024 ခုထိ ကွဲနိုင်ပါတယ်။

SELinux ရဲ့ အဓိက အသက်ဖြစ်တဲ့ ဟောဒီ Security Context တွေကို ယခုကဲ့သို့ သေချာ နားလည်ထားမှသာ အလွယ်ကူဆုံး Type Enforcement/Targeted Environment Policy မှ အခက်ဆုံး Multi-Level Security/Multi-Category Security အထိအပြင် ဆက်လက်လေ့လာသင့်တဲ့ Virtualization/Container/Cloud Security ပိုင်း တွေကိုထိန်းချုပ်တဲ့ sVirt တွေအထိ ဆက်လက် လေ့လာနိုင်မှာဖြစ်ပါတယ်။

SELinux Architecture (How SELinux works)

SELinux ဟာ Linux Security Module (LSM) တစ်ခုဖြစ်ပြီး Kernel ထဲထိပါ တည်ဆောက်ထားပြီးသား ဖြစ်ပါတယ်။ ဒါကြောင့် SELinux ကို အသုံးပြုထားတဲ့အချိန်မှာ - ဥပမာအားဖြင့် Apache Web Server မှ File တစ်ခုကို ဖတ်ဖို့အတွက် လုပ်ဆောင်ပါက တကယ် မဖတ်ရသေးခင်မှာ အဲဒီ လုပ်ဆောင်ချက် ကို Kernel ထဲသို့ ကြားဖြတ် ဖြတ်ယူ (Intercept) လိုက်ပါတယ်။ အဲဒီနောက်မှာတော့ DAC Permission တွေနဲ့တကွ SELinux ကိုယ်တိုင်ရဲ့ Policy တွေနဲ့ တိုက်စစ်ပါတယ်။ ကနဦးမှာ DAC Permission အရ ခွင့်ပြုသင့်လား ခွင့်မပြုသင့်လား ဆိုတာကို စတင်စစ်ဆေးပါတယ်။ DAC အရ ခွင့်ပြုလို့မရဘူးဆိုရင်တော့ ဆက်လက် တိုက်စစ်တာ မလုပ်တော့ဘဲ ချက်ချင်း Permission Denied/Drop လုပ်လိုက်ပါတယ်။ DAC Permission အရ ခွင့်ပြုထားတယ်ဆိုရင်တော့ MAC Permission အရ သို့မဟုတ် SELinux ရဲ့ Security Policy တွေအရ ခွင့်ပြုချက် ရှိမရှိ ကို ဆက်လက် စစ်ဆေးပါတယ်။ MAC Permission အရ ခွင့်မပြုတာမျိုး သို့မဟုတ် ခွင့်ပြုသည် မပြုသည်ကို Policy ထဲမှာ ရှာမတွေ့တာ မျိုးဆိုရင်တော့ Permission Denied/Drop ဆက်လက်လုပ်ဆောင်ပါတယ်။ MAC Permission တွေထဲမှာ ခွင့်ပြုထားတယ်ဆိုတဲ့ အကြောင်းအရာ ရှိမှသာ Apache Web Server ဟာ သက်ဆိုင်ရာ File ကို တကယ်ဖတ်ခွင့် ရသွားပြီး အဲဒီ Process ဟာ တကယ် Run သွားပါတယ်။
ထို့အပြင် SELinux မှ "ခွင့်ပြုလိုက်တယ်" ဆိုတဲ့ "Allow" အတွက်ကော "ခွင့်မပြုလိုက်ပါ" ဆိုတဲ့ "Denied" အတွက်ကော ၂ ခုစလုံးကို "Access Vector Cache (AVC)" ဆိုတဲ့ထဲမှာ Cache အနေနဲ့ သွားရောက် မှတ်သားလိုက်ပါတယ်။ အဲဒီလို Cache လုပ်ရတဲ့အကြောင်းကတော့ Cache မိပြီးသား Permission တစ်ခုကို ထပ်မံကြုံတွေ့ပါက SELinux ရဲ့ Policy ကြီးတစ်ခုလုံးနဲ့ ထပ်မံ တိုက်ဆိုင်စစ်ဆေးတာတွေ ထပ်မလုပ်တော့ပဲ Cache မှ Decision ကို စွဲယူကာ System ရဲ့ Performance ကို မထိခိုက်စေပဲ မြန်မြန်အသုံးပြုနိုင်အောင်ပဲဖြစ်ပါတယ်။

SELinux Modes

SELinux ရဲ့ အလုပ်လုပ်တဲ့ ပုံစံ ၃ မျိုး ရှိပါတယ်။ အဲဒါတွေကတော့ အောက်ပါအတိုင်းဖြစ်ပါတယ်။

- Enforcing Mode
- Permissive Mode
- Disabled Mode

Enforcing Mode

Enforcing Mode ဆိုတာကတော့ SELinux တကယ်အလုပ်လုပ်တဲ့ Mode ပဲဖြစ်ပါတယ်။ Security Policy တွေ အတိုင်း "Allow" သို့မဟုတ် "Deny" ကို သေချာအလုပ်လုပ်နေပါတယ်။ AVC မှာလည်း Cache ယူနေသလို File/Folder အသစ်တွေမှာ Label တွေကိုလည်း အမြဲထိုးပေးနေပါတယ်။

Permissive Mode

Permissive Mode မှာတော့ SELinux ဟာ "Allow" နဲ့ "Deny" ကို သေချာ မလုပ်ပါဘူး။ User/Process အားလုံး လုပ်သမျှရဲ့ MAC Permission အားလုံးကို ခွင့်ပြုပေးပါတယ်။ DAC Permission တွေသာ သက်ရောက်မှုရှိနေတဲ့ Mode ဖြစ်ပါတယ်။ ဒါပေမယ့် SELiux ရဲ့ Security Policy ထဲမှာ မပါဝင်တာနဲ့ ကန့်သတ်ထားမှုတွေရှိလာပါက 'root' User ကို Notify လုပ်ပေးပါတယ်။ ဘယ်လိုပဲဖြစ်ဖြစ် AVC မှာတော့ ပုံမှန်အတိုင်း Cache ယူနေသလို File/Folder အသစ်တွေမှာလည်း Label တွေ ထိုးပေးနေဆဲပါပဲ။
မှတ်သားထားရန်။ ။SELinux နဲ့ပတ်သက်ပြီး Troubleshooting ပြုလုပ်တဲ့အခါ ဒီ Permissive Mode ကို အသုံးပြုပြီး Log တွေ စစ်ထုတ်တဲ့အချိန်မျိုးတွေနဲ့ SELinux ကို အခြားအကြောင်းအမျိုးမျိုးနဲ့ ယာယီ ပိတ်ထားချင်တဲ့အချိန်မျိုးတွေမှာ အသုံးပြုပါတယ်။

Disabled Mode

Disabled Mode ကတော့ SELinux ဟာ လုံးဝကို အလုပ်မလုပ်တော့ပါဘူး။ Policy Rule တွေကို တိုက်စစ်တာတွေ မလုပ်တော့သလို AVC မှာလည်း Cache မယူတော့ပါဘူး။ File/Folder အသစ်တွေမှာလည်း Label တွေ ထိုးမပေးတော့ပါဘူး။

Checking & *Switching Modes*

SELinux ရဲ့ အလုပ်လုပ်တဲ့ပုံစံ Mode တွေကို ချက်ချင်း ယာယီပြောင်းလဲလို့ရပါတယ်။ လက်ရှိမှာ SELinux က ဘယ်လို Mode နဲ့ Run နေလဲဆိုတာကို getenforce ဆိုတဲ့ Command နဲ့ စစ်ကြည့်နိုင်ပြီး `setenforce` ဆိုတဲ့ command နဲ့ SELinux Mode တွေကို ပြောင်းလဲနိုင်ပါတယ်။ `setenforce 1` ဆိုရင်တော့ Enforcing Mode ကို ယာယီ ပြောင်းလဲပေးမှာဖြစ်ပါတယ်။ setenforce 0 ဆိုရင်တော့ Permissive Mode ကို ယာယီပြောင်းလဲပေးမှာပါ။

```
[root@lab ~]# getenforce
Enforcing
[root@lab ~]# setenforce 0
[root@lab ~]# getenforce
Permissive
[root@lab ~]# setenforce 1
[root@lab ~]# getenforce
Enforcing
```

ဒါပေမယ့် SELinux မှာ Enforcing Mode နဲ့ Permissive Mode နှစ်ခုအကြားသာ `setenforce` Command ကိုအသုံးပြုပြီး Switching သွားနိုင်တာပါ။ Disabled Mode နဲ့ Enforcing Mode သို့မဟုတ် Permissive Mode ကို အလွယ်တကူ ပြောင်းလို့မရပါဘူး။ Enforcing Mode သို့မဟုတ် Permissive Mode ဖြစ်နေပါက Disabled Mode ကိုပြောင်းဖို့အတွက် SELinux ရဲ့ Configuration File ထဲမှာ `SELinux=disabled` ဆိုတာလေး သွားပြောင်းပြီး reboot ချရမှာဖြစ်ပါတယ်။
Disabled Mode ဖြစ်နေတဲ့အခါမှာ အသစ်ဆောက်တဲ့ File/Folder တွေကို Label တွေမထိုးပါဘူး။ ဒါကြောင့် Disabled Mode ကနေ Permissive/Enforcing Mode ကိုပြောင်းတဲ့အခါမှာတော့ `touch /.autorelabel` ဆိုတဲ့ Command လေးရိုက်ပြီး System တစ်ခုလုံးမှာရှိတဲ့ File/Folder တွေမှာ ရှိတဲ့ Label တွေကို ပြန်စစ်ဆေးခိုင်းရပါတယ်။ Label မရှိတဲ့ File/Folder တွေကိုတော့ Label အသစ်ထိုးပေးမှာဖြစ်ပြီး Label ရှိပြီးသား File/Folder တွေကို SELinux ရဲ့ Security Policy နဲ့ ပြန်တိုက်ပြီး သင့်တော်တဲ့ Label တွေထိုးပေးသွားမှာပါ။

! သတိပြုမှတ်သားရန် !
လက်ရှိ အသုံးပြုနေတဲ့ Server တွေမှာတော့ Disabled Mode ကနေ Enforcing Mode ကို တိုက်ရိုက်ပြောင်းခြင်း မပြုသင့်ပါဘူး။ ဘာလို့လဲဆိုတော့ Enforcing Mode လုပ်လိုက်တဲ့အခါ Security Policy ကို တကယ် ထိထိရောက်ရောက်ကိုင်တွယ်မှာဖြစ်တဲ့အတွက် Server ကြီး Boot မတက်လာတော့တာမျိုး ဒါမှမဟုတ် တချို့သောနေရာမှာ Error တက်တာမျိုး ကြုံတွေ့လာရနိုင်ခြေရှိပါတယ်။ ဒါကြောင့် Disabled Mode ကနေ Permissive Mode ကိုအရင်ပြောင်းပေးပြီး Log များကို စစ်ဆေးသင့်ပါတယ်။ SELinux ရဲ့ Log တွေထဲမှာ Error တွေကင်းရှင်းတော့မှ Permissive Mode ကို Enforcing ပြောင်းရမှာပါ။

SELinux Configuration File

SELinux ရဲ့ Configuration File ကိုတော့ `/etc/selinux/config` မှာ တွေ့ရှိနိုင်ပါတယ်။ Config ထဲမှာတော့ အောက်ပါအတိုင်း တွေ့မြင်နိုင်ပါတယ်။

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=enforcing
# SELINUXTYPE= can take one of three two values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected
processes are protected.
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

`SELINUX=enforcing` ဆိုတာလေးကတော့ SELinux ကို Boot တက်ထဲက ဘယ်လို Mode မျိုးနဲ့ Run မလဲဆိုတာကို အသေသတ်မှတ်ထားခြင်းပဲဖြစ်ပါတယ်။ အကယ်၍ SELinux ကို Boot တက်ထဲက Permissive ပြောင်းထားချင်ရင်တော့ `SELINUX=permissive` ထားရမှာဖြစ်ပြီး လုံးဝပိတ်လိုက်ချင်တယ်ဆိုရင်တော့ အဲဒီနေရာမှာ SELINUX=disabled ထားထားရမှာပါ။

`SELINUXTYPE=targeted` နေရာကတော့ Boot စ တက်ထဲက ဘယ်လို Security Policy Type တွေနဲ့ Run မလဲဆိုတာကို သတ်မှတ်ပေးလို့ရပါတယ်။ `SELINUXTYPE=targeted` ကတော့ SELinux ရဲ့ အလွယ်ကူဆုံး "Type Enforcement" သို့မဟုတ် "Targeted Environment" Security Policy ကို ရည်ညွှန်းတာပါ။ အဲဒီနေရာမှာ `SELINUXTYPE=minimum` ဆိုရင်တော့ Targeted Environment လောက် Security ထိန်းချုပ်မှုရှိမှာမဟုတ်တော့ပဲ အနည်းငယ်လျော့နည်းသွားမှာပါ။ `SELINUXTYPE=mls` ကတော့ SELinux ရဲ့ အခက်ခဲဆုံးအပိုင်းဖြစ်တဲ့ "Multi-Level Security" Policy တွေကို System တစ်ခုလုံးအပေါ်ကို သက်ရောက်စေပါလိမ့်မယ်။

SELinux Policy Types

SELinux ရဲ့ Policy Types တွေကို တော့ အပေါ်က Configuration File ထဲကအတိုင်းပါပဲ။ Policy Type ၃ မျိုးရှိပါမယ်။

- Targeted Environment (သို့မဟုတ်) Type Enforcement (သို့မဟုတ်) အတိုခေါက် "TE"
- Minimum
- Multi-Level Security

***Targeted Environment* (*targeted*)**

Targeted Environment ကတော့ SELinux Security Context (Label) တွေထဲက "Type" အပိုင်းကို အသားပေးပြီး Permission များကို ထိန်ချုပ်တဲ့ လုံခြုံရေးပုံစံပါပဲ။ Network ပေါ်ကနေ ချိတ်ဆက်လို့ရတဲ့ Application တွေ Service တွေဟာ Server တစ်လုံးအတွက်တော့ လုံခြုံရေးကို ချိုးဖောက်ဝင်ရောက်ချင်သူတွေ အတွက် တံခါးပေါက်တွေပါပဲ။ ဒါကြောင့် အဲဒီ Application/Service တွေကိုသာ Domain တွေခွဲပြီး ချုပ်ကိုင်လိုက်မယ်ဆိုရင် အထိုက်အသင့် လုံခြုံရေးကို ပိုမိုပေးစွမ်းနိုင်မယ် ဆိုတာဟာ Targeted Environment ကို အကောင်အထည်ဖော်သူတွေရဲ့ Concept ပါပဲ။ ဒါကြောင့် Type Label တွေမှာ Domain တွေခွဲထားပြီး Domain အချင်းချင်း ချိတ်ဆက်မှုတွေကို Policy များရေးသားထားခြင်းဟာ Target Environment ပဲဖြစ်ပါတယ်။ မှတ်ချက်။ ။လူတော်တော်များများဟာ "Domain" လို့ပြောလိုက်ရင် .com တွေ .net တွေ စတဲ့ DNS Name တွေကို ပြေးမြင်ကြပါတယ်။ ဒါပေမယ့် "Domain" ဆိုတာ အစုအဖွဲ့ ကို ပြောချင်တာပါ။ ဒါကြောင့် Type Label ရဲ့ Domain ဆိုတဲ့နေရာမှာ အစုအဖွဲ့တစ်ခုခုကို သတ်မှတ်ပေးတဲ့ Label တွေသာဖြစ်ပါတယ်။

***Minimum* (*minimum*)**

Minimum Policy ကတော့ အထူးသဖြင့် Low Resource Machine တွေအတွက် ထုတ်ပေးထားတာဖြစ်ပါတယ်။ Server တွေလောက် Compute Power, RAM တွေ မမြင့်မားတဲ့ ဥပမာ Phone တွေ၊ Single Board Computer တွေ (Example: RaspberryPi) မှာ SELinux အသုံးပြုတဲ့အခါ RAM အသုံးပြုမှု အနည်းဆုံးဖြစ်အောင် စီမံပေးထားတာပဲဖြစ်ပါတယ်။ Minimum ထဲမှာတော့ Base Policy တွေနဲ့တကွ Unconfined Domain တွေကိုသာ Enable ပြုလုပ်ထားပေမယ့် အခြား Domain နဲ့ Policy တွေလည်းထည့်သွင်းထားပါတယ်။ ဒါပေမယ့် အဲဒီ အခြား Policy တွေကိုတော့ RAM အသုံးပြုမှု အနည်းဆုံးဖြစ်အောင် Enable ပြုလုပ်ထားခြင်းမရှိပါဘူး။

***Multi-Level Security* (*mls*)**

MLS မှာတော့ SELinux Security Context (Label) တွေထဲက အနောက်ဆုံးအပိုင်းဖြစ်တဲ့ Levels ဆိုတဲ့အပိုင်းကို အဓိကထားပြီး အရင်ဆုံး ကိုင်တွယ်ပါတယ်။ နောက်ပြီး MLS Policy ကတော့ "Server" တွေအတွက်ပဲ အဓိကထားပြီး ထုတ်ထားပေးတာပါ။ Desktop/Workstation တွေအတွက်ဆိုရင်တော့ MLS ကိုသုံးရတာ အတော်လေး လက်ပေါက်ကပ်ပါလိမ့်မယ်။ MLS Policy ဟာ Targeted Environment နဲ့ တွဲဖက်အလုပ်လုပ်တယ်ဆိုပေမယ့် MLS ရဲ့ Sensitivity Level (s0 မှ s15 ထိထားနိုင်) နဲ့ Category Level (c0 မှ c1023 အထိ ထားနိုင်) ဆိုတဲ့အပိုင်းနှစ်ပိုင်းကို အလေးပေးပါတယ်။ အဲဒီ Level တွေဟာ တစ်ခုစီ သီးခြားစီ ဖြစ်နိုင်သလို Range တွေအနေနဲ့လည်း ဖြစ်နိုင်ပါတယ်။
သတိပြုရန်။ ။MLS ဟာ စတင်လေ့လာသူတွေအဖို့ အရမ်းခက်ခဲပြီး ရှုပ်ထွေးတဲ့အတွက် "TE" Policy တွေကို ပိုင်နိုင်မှသာ ဆက်လက်လေ့လာတာ ပိုကောင်းပါတယ်။

SELinux Tools

SELinux ကို ကောင်းကောင်းအသုံးပြုဖို့အတွက် ထည့်သွင်းထားသင့်တဲ့ Package လေးတွေ ရှိပါတယ်။ အဲဒါလေးတွေကတော့ အောက်ပါ Package လေးတွေဖြစ်ပါတယ်။ မိမိစိတ်ကြိုက် တစ်ခုချင်းစီကိုသော်လည်းကောင်း အားလုံးကိုသော်လည်းကောင်း yum install နဲ့ ထည့်သွင်းနိုင်ပါတယ်။

- `policycoreutils-python`
- `setools-console`

`policycoreutils-python` ကတော့ SELinux ကို Command Line Interface ကနေ သေချာထိန်းကျောင်းပေးနိုင်မယ့် အဓိက Tool/Command တစ်ခုပဲဖြစ်ပါတယ်။ SELinux ကိုအသုံးပြုဖို့အတွက် အဓိကမဖြစ်မနေ လိုအပ်တဲ့ Tool ပဲဖြစ်ပါတယ်။

`setools-console` ကတော့ အသုံးပြုနေတဲ့ SELinux ထဲမှာရှိတဲ့ Information တွေကို ပြန်လည်ဖတ်ရှုတဲ့အချိန် ပေ့လာတဲ့အချိန်တွေမှာ သုံးလို့ရတဲ့ Command တွေပါဝင်ပါတယ်။

Playing with Security Contexts on Files

File/Folder တွေပေါ်မှာ ရှိတဲ့ SELinux ရဲ့ Label လေးတွေကို လိုအပ်သလို အလွယ်တကူ ပြောင်းလဲပစ်လို့ရပါတယ်။ ပြောင်းလဲပစ်တဲ့အခါမှာ ၂ မျိုးပြောင်းလဲလို့ရပါတယ်။ ပထမတစ်နည်းကတော့ မိမိ စိတ်ကြိုက် Label ကို chcon ဆိုတဲ့ Command ကိုအသုံးပြုပြီး ပြောင်းလဲခြင်းဖြစ်ပါတယ်။ ဒါပေမယ့် chcon နဲ့ပြောင်းလဲလိုက်ခြင်းက System Reboot ဖြစ်တဲ့အခါမှာ ပြောင်းထားသမျှတွေ အားလုံးပျောက်သွားပြီး သူ့ရှိရင်စွဲအတိုင်း ပြန်ဖြစ်သွားပါတယ်။ အမြဲတမ်းအတွက်ပြောင်းလဲလိုက်ခြင်းမျိုးမဟုတ်ပါဘူး။ ဒုတိယတစ်နည်းကတော့ SELinux ရဲ့ Security Policy ထဲအထိ ဘယ် Path လမ်းကြောင်းမှာ ဘယ်လို Label မျိုးရှိသင့်တယ်ဆိုတာကို ထည့်ရေးလိုက်ခြင်းပဲဖြစ်ပါတယ်။ ဒုတိယ နည်းလမ်းအတွက်တော့ အမြဲတမ်းပြောင်းလဲလိုက်ခြင်းမျိုး ဖြစ်ပါတယ်။ System Reboot ကျသွားရင်လည်း ပြောင်းထားတဲ့အတိုင်း ဆက်လက်ကျန်ရှိနေမှာပဲဖြစ်ပါတယ်။

File/Folder တွေအပေါ်မှာ ရှိတဲ့ SELinux Security Context တွေကို ကြည့်လို့ရတဲ့ Command ကတော့ ကျွန်တော်တို့သိပြီးသား Command တစ်ခုကို -Z ထည့်လိုက်တာပါပဲ။

```
[root@lab selinux]# ls -Z file.txt
-rw-r--r--. root root unconfined_u:object_r:usr_t:s0   file.txt
```

အပေါ်ကပြထားတဲ့ Output လေးမှာဆိုရင်တော့ file.txt လေးရဲ့ SELinux Label တွေဟာ `unconfined_u:object_r:usr_t:s0` တွေဖြစ်နေတယ်ဆိုတာ တွေ့ရမှာပါ။

Manually Changing Security Contexts of Files

`chcon` Command ကိုအသုံးပြုပြီး File/Folder တွေရဲ့ Label တွေကို ပြောင်းလို့ရပါတယ်။ အောက်က Example လေးကတော့ file.txt လေးရဲ့ Type ကို `usr_t` ကနေ `user_home_t` ကိုပြောင်းလဲခြင်းဖြစ်ပါတယ်။

```
[root@lab selinux]# ls -Z file.txt
-rw-r--r--. root root unconfined_u:object_r:usr_t:s0   file.txt
[root@lab selinux]# chcon -t user_home_t file.txt
[root@lab selinux]# ls -Z
-rw-r--r--. root root unconfined_u:object_r:user_home_t:s0 file.txt
```

Folder တွေအောက်မှာရှိတဲ့ အခြားသော File Folder တွေ အားလုံးကိုပြောင်းချင်ရင်တော့ Recursive အနေနဲ့ `-R` ထည့်ပေးလို့ရပါတယ်။

```
[root@lab selinux]# ls -Z
drwxr-xr-x. root root unconfined_u:object_r:usr_t:s0   folder
[root@lab selinux]# ls -Z folder/
-rw-r--r--. root root unconfined_u:object_r:usr_t:s0   file.txt
[root@lab selinux]# chcon -R -t user_home_t folder/
[root@lab selinux]# ls -Z
drwxr-xr-x. root root unconfined_u:object_r:user_home_t:s0 folder
[root@lab selinux]# ls -Z folder/
-rw-r--r--. root root unconfined_u:object_r:user_home_t:s0 file.txt
```

မှတ်သားထားရန်။ ။`chcon` နဲ့ ပြောင်းလိုက်တဲ့ Label တွေဟာ အကယ်၍ Server Reboot ကျသွားရင်ပဲဖြစ်ဖြစ် `restorecon` နဲ့ Label တွေကို Restore ပြုလုပ်ရင်ပဲဖြစ်ဖြစ် မူရင်း Label တွေကို ပြန်လည်ပြောင်းသွားမှာပါ။

Using fcontext to Change File Security Contexts Automatically

`semanage fcontext` ဆိုတဲ့ Command ကိုသုံးပြီး Path လမ်းကြောင်းတစ်ခုအောက်က File/Folder တွေမှာ (သို့မဟုတ်) Path တစ်ခုမှာရှိတဲ့ File/Folder တစ်ခုကို Security Context ကိုသတ်သတ်မှတ်မှတ်ပြောင်းထားလို့ရပါတယ်။ `semanage fcontext` ကိုသုံးပြီးပြောင်းလိုက်တဲ့ Label တွေကတော့ အသေမှတ်သွားမှာဖြစ်တဲ့အတွက် Reboot ကျပြီးပဲဖြစ်ဖြစ် `restorecon` Command ကြောင့်ပဲဖြစ်ဖြစ် ပထမ မပြောင်းခင်ရှိခဲ့တဲ့ Label ကို ပြန်မပြောင်းတော့ပါဘူး။

```
[root@lab permission]# touch file.txt
[root@lab permission]# ls -Z
-rw-r--r--. root root unconfined_u:object_r:usr_t:s0   file.txt
[root@lab permission]# semanage fcontext -a -t httpd_sys_content_t
"/opt/permission(/.*)?"
[root@lab permission]# restorecon -rv .
restorecon reset /opt/permission context
unconfined_u:object_r:usr_t:s0-
>unconfined_u:object_r:httpd_sys_content_t:s0
restorecon reset /opt/permission/file.txt context
unconfined_u:object_r:usr_t:s0-
>unconfined_u:object_r:httpd_sys_content_t:s0
[root@lab permission]# touch file2.txt
[root@lab permission]# ls -Z
-rw-r--r--. root root unconfined_u:object_r:httpd_sys_content_t:s0
file2.txt
-rw-r--r--. root root unconfined_u:object_r:httpd_sys_content_t:s0
file.txt
[root@lab permission]# touch file{3..5}.txt
[root@lab permission]# ls -Z
-rw-r--r--. root root unconfined_u:object_r:httpd_sys_content_t:s0
file2.txt
-rw-r--r--. root root unconfined_u:object_r:httpd_sys_content_t:s0
file3.txt
-rw-r--r--. root root unconfined_u:object_r:httpd_sys_content_t:s0
file4.txt
-rw-r--r--. root root unconfined_u:object_r:httpd_sys_content_t:s0
file5.txt
-rw-r--r--. root root unconfined_u:object_r:httpd_sys_content_t:s0
file.txt
```

`semanage fcontext -a -t httpd_sys_content_t "/opt/permission(/.*)?"` ဆိုတဲ့ Command လေးမှာတော့ `semanage fcontext` နဲ့ စပြီး `-a` ကတော့ အသစ်ထည့်မယ်ဆိုတဲ့ 'Add' ကိုဆိုလိုချင်တာပါ။ ထည့်တဲ့အခါ ကိုယ်ထည့်မယ့် Path လမ်းကြောင်းထဲက File တွေကို ဘယ် Label ပေးမလဲဆိုတာက "User" "Role" "Type" "Level" တွေကို သက်ဆိုင်ရာ Tag တွေနဲ့ သတ်မှတ်ပေးလို့ရပါတယ်။ ကျွန်တော် ရေးထားတာက `httpd_sys_content_t` ဆိုတဲ့ "Type" ကို သတ်မှတ်မယ်ဆိုပြီး `-t httpd_sys_content_t` ကို ရေးသားထားတာပါ။ နောက်ဆုံးက `"/opt/permission(/.*)?"` ဆိုတာကတော့ `/opt/permission` အောက်မှာ ရှိတဲ့ File တွေ Folder တွေ အားလုံးကို ဆိုလိုချင်တာပါ။

`restorecon -rv .` ဆိုတာကတော့ လက်ရှိ Shell ရောက်နေတဲ့ Directory အောက်က File တွေ Folder တွေ အားလုံးကို SELinux ရဲ့ fcontext ထဲမှာ မှတ်ထားတဲ့အတိုင်း Label တွေ အလိုလျောက် ပြန်ပြင်ဖို့အတွက်

စေခိုင်းတဲ့ Command ဖြစ်ပါတယ်။ `-r` ကတော့ Recursive (ထည့်သွင်းထားတဲ့ Path '.' ရဲ့ အောက်က File/Folder တွေအားလုံး) ကိုဆိုလိုတာဖြစ်ပြီး `-v` ကတော့ ဘာတွေ ဘယ်လို ပြောင်းသွားလဲဆိုတာကို Terminal ကို ထုတ်ပြပေးအောင် Verbose ထုတ်ထားပေးတာပါ။

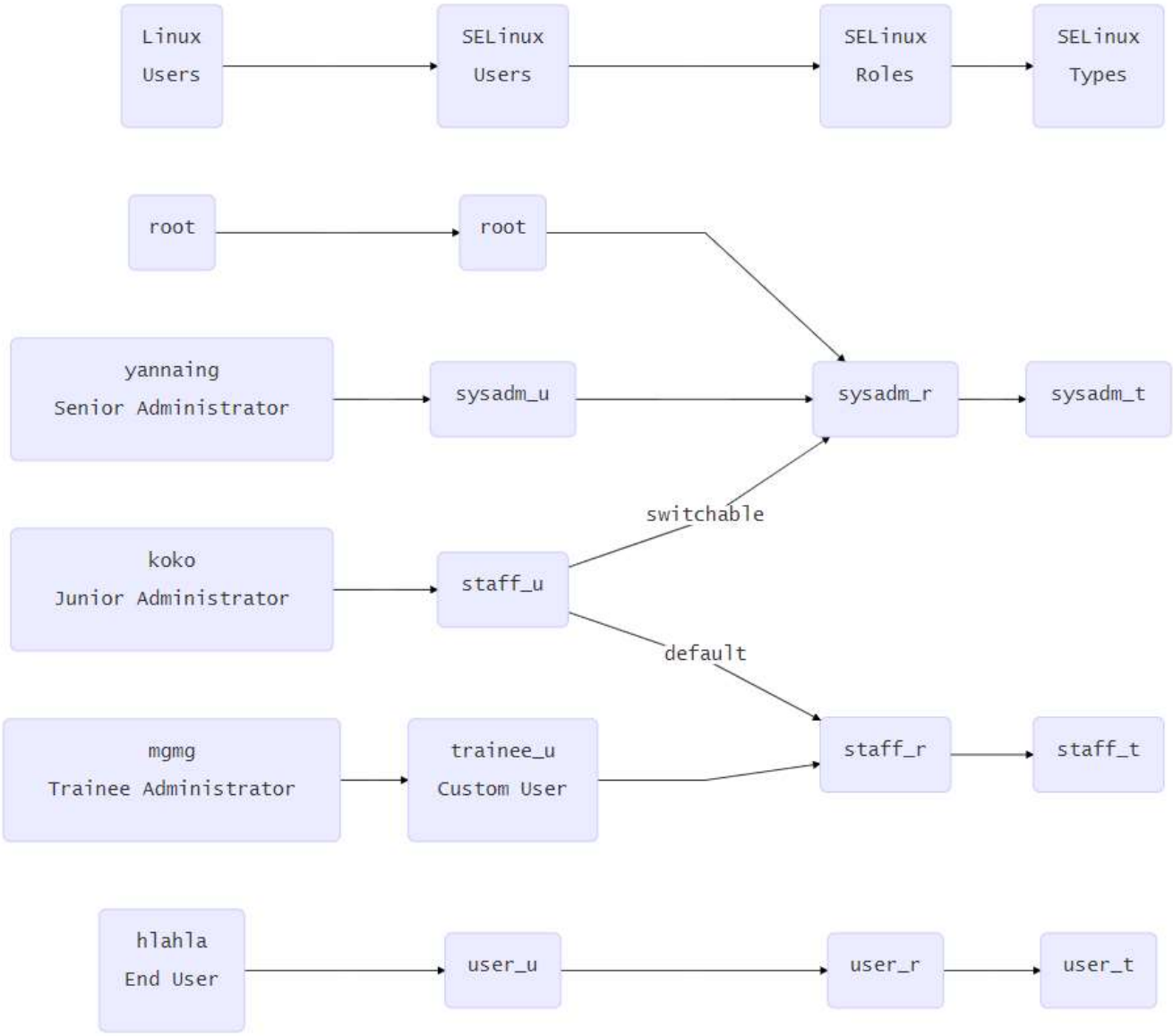
`fcontext` နဲ့ `restorecon` လုပ်ပြီးနောက်မှာတော့ လက်ရှိ File ထဲမှာ File တွေဘယ်လောက်ထပ်ပြီး ဆောက်ဆောက် `semanage fcontext` ထဲထည့်ခဲ့တဲ့ Label တွေ ထိုးပြီးသားဖြစ်နေတယ်ဆိုတာ File တွေထပ် ဆောက်ကြည့်ပြီး ဆက်လက် စမ်းသပ်ပြထားတာပါပဲ။

တစ်နည်း မှတ်သားထားဖို့ရှိတာက System တစ်ခုလုံးမှာ ရှိတဲ့ Path လမ်းကြောင်းတွေမှာ ဘယ်လို Label တွေကို Default မှတ်ထားလဲဆိုတာကို `semanage fcontext -l` ဆိုတဲ့ Command နဲ့ထုတ်ကြည့်နိုင်ပါတယ်။ စာတွေ ဟာများလွန်းတာကြောင့် grep နဲ့တွဲပြီး မိမိလိုချင်တာပဲ ထုတ်ကြည့်ဖို့အတွက် အကြံပေးပါတယ်။ အောက်မှာ ဥပမာ ပြထားပေးပါတယ်။

```
[root@lab permission]# semanage fcontext -l | grep httpd_sys_content_t
/srv/([^/]*/)?www(/.*)?                            all files          system_u:object_r:httpd_sys_content_t:s0
/var/www(/.*)?                                     all files          system_u:object_r:httpd_sys_content_t:s0
/etc/htdig(/.*)?                                   all files          system_u:object_r:httpd_sys_content_t:s0
/srv/gallery2(/.*)?                                all files          system_u:object_r:httpd_sys_content_t:s0
/var/lib/trac(/.*)?                                all files          system_u:object_r:httpd_sys_content_t:s0
/var/lib/htdig(/.*)?                               all files          system_u:object_r:httpd_sys_content_t:s0
/var/www/icons(/.*)?                               all files          system_u:object_r:httpd_sys_content_t:s0
/usr/share/glpi(/.*)?                              all files          system_u:object_r:httpd_sys_content_t:s0
/usr/share/htdig(/.*)?                             all files          system_u:object_r:httpd_sys_content_t:s0
/usr/share/drupal.*                                all files          system_u:object_r:httpd_sys_content_t:s0
/usr/share/z-push(/.*)?                            all files          system_u:object_r:httpd_sys_content_t:s0
/var/www/svn/conf(/.*)?                            all files          system_u:object_r:httpd_sys_content_t:s0
/usr/share/icecast(/.*)?                           all files          system_u:object_r:httpd_sys_content_t:s0
/var/lib/cacti/rra(/.*)?                           all files          system_u:object_r:httpd_sys_content_t:s0
/usr/share/ntop/html(/.*)?                         all files          system_u:object_r:httpd_sys_content_t:s0
/usr/share/nginx/html(/.*)?                        all files          system_u:object_r:httpd_sys_content_t:s0
/usr/share/doc/ghc/html(/.*)?                      all files          system_u:object_r:httpd_sys_content_t:s0
/usr/share/openca/htdocs(/.*)?                     all files          system_u:object_r:httpd_sys_content_t:s0
/usr/share/selinux-policy[^/]*/html(/.*)?          all files          system_u:object_r:httpd_sys_content_t:s0
/opt/permission(/.*)?                              all files          system_u:object_r:httpd_sys_content_t:s0
```

Relations in Linux Users and SELinux Security Contexts

Linux System ထဲက User တွေကို SELinux ရဲ့ User တွေနဲ့ ပေါင်းစပ်ပြီး မတူညီတဲ့ Permission တွေကို ထိန်းချုပ်နိုင်ပါတယ်။ တကယ်တော့ SELinux ရဲ့ User ဆိုတာ Idenity အတွက် သတ်သတ်ပါပဲ။ တကယ်တန်း ဘာတွေလုပ်လို့ရလဲဆိုတာကို အထွေထွေ အဆင့်မှာ ထိန်းချုပ်တာဟာ SELinux ရဲ့ Role ဖြစ်ပါတယ်။ Type ကတော့ မတူညီတဲ့ Object တွေကို မတူညီတဲ့ Subject ကနေ 'Read' 'Write' 'Execute' လုပ်ဆောင်မှုတွေ ပြုလုပ်တဲ့အခါ အသေးစိတ် Permission တွေကို အသေချာဆုံးဖြစ်အောင် ထိန်းချုပ်ပေးတာပါပဲ။
ဒါကြောင့် Permission ထိန်းချုပ်ဖို့ ပြင်ဆင်ပုံအဆင့်ဆင့်မှာတော့ ပထမဆုံး Linux User တွေကို SELinux User တွေနဲ့ အရင် Mapping တွဲပေးသင့်ပါတယ်။ တကယ်လို့ Custom Permission ထားချင်ရင်တော့ SELinux User တွေကိုပါ အသစ်ဆောက်ပေးသင့်ပါတယ်။ ပြီးမှ ဘယ် SELinux User ကိုတော့ ဘယ်လို Role တွေနဲ့ တွဲဖက် ထားမလဲဆိုတာကို ထပ်မံ ထည့်သွင်းပေးရပါမယ်။ SELinux User တစ်ယောက်က မတူညီတဲ့ Role တွေကို ပြောင်း လဲအသုံးပြုနိုင်ပါတယ်။ အဲဒီ Role တွေပေါ်မူတည်ပြီးမှ ဘယ် Type တွေကိုဆက်ပြီး Mapping လုပ်ရမလဲဆိုတာ ထပ် စဉ်းစားရမှာပါ။ ဒါဟာ ပြုလုပ်တဲ့သူ 'Subject' ရဲ့ Label ကို ကိုယ် လိုသလို ပြင်ဆင်ခြင်းပဲဖြစ်ပါတယ်။ ဒါမှ ခံရမယ့် 'Object' တွေကို ဘယ် 'Subject' တွေကိုပဲ ဘာလုပ်ခွင့်ပေးမလဲဆိုတာကို ထိန်းချုပ်နိုင်မှာဖြစ်ပါတယ်။



အပေါ်က ပုံထဲမှာတော့ root User ကိုတော့ ထွေထွေထူးထူးပြောပြစရာမရှိပေမယ့် yannaing ရယ် koko ရယ် mgmg ရယ်ဆိုတဲ့ User ၃ ယောက်ရဲ့ ကွာဟချက်ကလေးကို အခြေအနေ ၂ ရပ်ပေါ်မူတည်ပြီး အဓိက ပြောပြရမှာပါ။

<u>အခြေအနေ (၁)</u>

yannaing နဲ့ koko နဲ့ နှစ်ယောက်အကြား ကွာခြားချက်ကတော့ အပြင်ထကယ့်လောကမှာ yannaing က Senior Server Administrator ရာထူးကို ကိုင်ထားပြီး mgmg ကတော့ Junior Server Administrator ရာထူးကို ကိုင်ဆောင်ထားတယ်ဆိုပါစို့။ ဒီအတွက်ကြောင့် yannaing ဟာ `sysadm_u` ဆိုတဲ့ SELinux User ကို Mapping တွဲထားပါတယ်။ koko ကတော့ staff_u ဆိုတဲ့ SELinux User ကို Mapping တွဲထားပါတယ်။ `sysadm_u` ပိုင်ဆိုင်ထားတဲ့ yannaing ဆိုတဲ့ User ကတော့ 'su' Command ကိုအသုံးပြုပြီး 'root' ကိုပြောင်းလဲလို့ရတဲ့ `sysadm_r` ကိုပါ တွဲပြီးပိုင်ဆိုင်ထားပါတယ်။ ဒါပေမယ့် koko ရဲ့ `staff_u` ဆိုတာကတော့ Default Mapping အရ `staff_r` ကိုပဲ တွဲထားပါတယ်။ `staff_r` ကတော့ Root User ကို ပြောင်းသုံးလို့မရပါဘူး။ ဒါကြောင့် koko ဆိုတဲ့ User က 'root'

ဖြစ်ချင်ရင် သူ့ရဲ့ Role ကို `sysadm_u` ဆီကို ပြောင်းပြီးမှ System Administration တွေကို ထိထိရောက်ရောက် လုပ်နိုင်ပါတယ်။ Role ပြောင်းရင် Password ထပ်တောင်းခံလေ့ရှိပါတယ်။

ဒါပေမယ့် mgmg ဆိုတဲ့ အလုပ်သင် Administrator လေးအတွက်တော့ SELinux ထဲမှာ မပါဝင်တဲ့ သီးခြား `trainee_u` ဆိုတာလေးကိုပဲ Mapping တွဲထားပေးပါတယ်။ `trainee_u` ကတော့ `staff_r` ကိုသာ Mapping တွဲထားတဲ့အတွက် သူ အများဆုံးရရှိတဲ့ Role ဟာ `staff_r` ပဲရှိပါတယ်။ ဒါကြောင့် mgmg ဟာ 'root' User ကို ပြောင်းသုံးဖို့အတွက် Permission လုံးဝ မရှိပါဘူး။

<u>အခြေအနေ (၂)</u>

SELinux ရဲ့ Default အရ `sysadm_u` တွေဟာ Network ကနေ SSH Login ပေးမလုပ်ပါဘူး။ တကယ်လို့ အဲဒီ Default Policy ကိုလည်း မပြင်ချင်ဘူးဆိုပါက yannaing ဟာ Server နဲ့အတူရှိနေတဲ့သူမဟုတ်ရင် Network ပေါ်ကနေတဆင့် SSH Login ဝင်နိုင်ဖို့အတွက် `staff_u` အဖြစ် Mapping တွဲထားနိုင်ပါတယ်။

Managing SELinux Users

SELinux ရဲ့ User တွေကို မိမိစိတ်ကြိုက် ထပ်ထည့်လို့ရပါသေးတယ်။ SELinux ရဲ့ User တွေကို ထပ်ထည့်ခြင်းအားဖြင့် Permission တွေကို အသေးစိတ် ထပ်မံထိန်းချုပ်လို့ရပါတယ်။ SELinux ထဲမှာ ပါဝင်ထားပြီးသား User တွေကို Detail နဲ့တကွ ကြည့်ချင်ရင်တော့ အောက်ပါအတိုင်း ကြည့်နိုင်ပါမယ်။

```
[root@lab permission]# semanage user -l

SELinux       Labeling   MLS/   MLS/
User          Prefix     MCS    MCS Range        SELinux Roles

guest_u       user       s0     s0               guest_r
root          user       s0     s0-s0:c0.c1023   staff_r sysadm_r system_r unconfined_r
staff_u       user       s0     s0-s0:c0.c1023   staff_r sysadm_r system_r unconfined_r
sysadm_u      user       s0     s0-s0:c0.c1023   sysadm_r
system_u      user       s0     s0-s0:c0.c1023   system_r unconfined_r
unconfined_u  user       s0     s0-s0:c0.c1023   system_r unconfined_r
user_u        user       s0     s0               user_r
xguest_u      user       s0     s0               xguest_r
```

`semanage user` ရဲ့ **Command Syntax** ကို လေ့လာနိုင်အောင် ထည့်သွင်းဖော်ပြထားပါတယ်။

```
semanage user [-h][-n] [-N][-S STORE] [ --add ( -L LEVEL -R ROLES -r RANGE
-s SEUSER selinux_name) | --delete selinux_name | --deleteall  | --extract  |
--list -C | --modify ( -L LEVEL -R ROLES -r RANGE -s SEUSER selinux_name ) ]
```

ဆက်လက်ပြီး SELinux User တစ်ခုကို Role တစ်ခုနဲ့ တွဲဖက်ပြီး တည်ဆောက်ကြည့်ပါမယ်။

```
[root@lab permission]# semanage user -a -R staff_r trainee_u
[root@lab permission]# semanage user -l

SELinux       Labeling   MLS/   MLS/
User          Prefix     MCS    MCS Range        SELinux Roles

trainee_u     user       s0     s0               staff_r
guest_u       user       s0     s0               guest_r
root          user       s0     s0-s0:c0.c1023   staff_r sysadm_r system_r unconfined_r
staff_u       user       s0     s0-s0:c0.c1023   staff_r sysadm_r system_r unconfined_r
sysadm_u      user       s0     s0-s0:c0.c1023   sysadm_r
system_u      user       s0     s0-s0:c0.c1023   system_r unconfined_r
unconfined_u  user       s0     s0-s0:c0.c1023   system_r unconfined_r
user_u        user       s0     s0               user_r
xguest_u      user       s0     s0               xguest_r
```

`semanage user -a -R staff_r trainee_u` ဆိုတဲ့ Command ကတော့

- `-a` ဆိုတာ User အသစ်ထည့်မယ်ဆိုတာကို ရည်ညွှန်းပါတယ်။
- `-R staff_r` ဆိုတာ အသစ်ထည့်မယ့် User ကို 'staff_r' ဆိုတဲ့ Role နဲ့တကွ ပေါင်းထည့်မယ်လို့ ရည်ညွှန်းပါတယ်။
- trainee_u ဆိုတာကတော့ အသစ်ထည့်မယ့် SELinux User နာမည်ပါ။

အကယ်၍ Role ၂ ခုထည့်ချင်တယ်ဆိုရင်တော့ အစထဲက `-R role_r` ၂ ခု နဲ့ ပဲဖြစ်ဖြစ် Double Quote ထဲကို ထည့်ချင်တဲ့ Role တွေ ရေးထည့်ပြီးပဲဖြစ်ဖြစ် ထည့်နိုင်ပါတယ်။ အခုအခြေအနေအတိုင်းကို Role ၂ ခု ဖြစ်အောင် ထည့်ဖို့အတွက်ဆိုရင်တော့ အောက်ပါအတိုင်း `-a` နေရာမှာ `-m` နဲ့ Modify ပြုလုပ်နိုင်ပါတယ်။

```
[root@lab permission]# semanage user -m -R staff_r -R sysadm_r trainee_u
[root@lab permission]# semanage user -l

SELinux       Labeling   MLS/   MLS/
User          Prefix     MCS    MCS Range        SELinux Roles

trainee_u     user       s0     s0               staff_r sysadm_r
guest_u       user       s0     s0               guest_r
root          user       s0     s0-s0:c0.c1023   staff_r sysadm_r system_r unconfined_r
staff_u       user       s0     s0-s0:c0.c1023   staff_r sysadm_r system_r unconfined_r
sysadm_u      user       s0     s0-s0:c0.c1023   sysadm_r
system_u      user       s0     s0-s0:c0.c1023   system_r unconfined_r
unconfined_u  user       s0     s0-s0:c0.c1023   system_r unconfined_r
user_u        user       s0     s0               user_r
xguest_u      user       s0     s0               xguest_r
```

အခြားသော Field တွေကိုပါ ထည့်သွင်းပြောင်းလဲချင်ရင်တော့

- `-L` နဲ့ MLS/MCS Level တွေ
- `-r` နဲ့ MLS/MCS Range တွေ

ထည့်သွင်း/ပြောင်းလဲလို့ ရပါသေးတယ်။

ခုထည့်လိုက်တဲ့ SELinux User ကို ပြန်ဖျက်ချင်ရင်တော့ -d နဲ့ ဖျက်ပစ်နိုင်ပါတယ်။

```
[root@lab permission]# semanage user -d trainee_u
[root@lab permission]# semanage user -l

SELinux       Labeling   MLS/   MLS/
User          Prefix     MCS    MCS Range        SELinux Roles

guest_u       user       s0     s0               guest_r
root          user       s0     s0-s0:c0.c1023   staff_r sysadm_r system_r unconfined_r
staff_u       user       s0     s0-s0:c0.c1023   staff_r sysadm_r system_r unconfined_r
sysadm_u      user       s0     s0-s0:c0.c1023   sysadm_r
system_u      user       s0     s0-s0:c0.c1023   system_r unconfined_r
unconfined_u  user       s0     s0-s0:c0.c1023   system_r unconfined_r
user_u        user       s0     s0               user_r
xguest_u      user       s0     s0               xguest_r
```

Confining User Logins

SELinux ကိုအသုံးပြုတယ်ဆိုရင် User တွေကိုပါ Label တပ်ပေးတယ်ဆိုတာ သိခဲ့ပြီးသားဖြစ်မှာပါ။ User တွေကို Login ဝင်တဲ့အချိန်ကစပြီး ဘယ် Label တွေ တပ်ဆင်ပြီး Permission ထိန်းချုပ်ထားမလဲဆိုတာ သတ်မှတ်ထားလို့ရပါတယ်။ နောက်တစ်နည်းပြောရရင်တော့ Linux User နဲ့ SELinux User နဲ့ကို Mapping တွဲတာဟာ ဒီနေရာမှာ တွဲတာပါ။ စစချင်း လက်ရှိ ရှိနေတဲ့ User Login တွေကို ကြည့်ရအောင်။

```
[root@lab users]# semanage login -l

Login Name           SELinux User         MLS/MCS Range        Service

__default__          unconfined_u         s0-s0:c0.c1023       *
root                 unconfined_u         s0-s0:c0.c1023       *
system_u             system_u             s0-s0:c0.c1023       *
```

`semanage login` ရဲ့ **Command Syntax** ကို ဆက်လက်လေ့လာနိုင်အောင် ထည့်သွင်းဖေါ်ပြပေးလိုက်ပါတယ်။

```
      semanage login [-h][-n] [-N][-S STORE] [ --add -s SEUSER -r RANGE LOGIN |
      --delete LOGIN | --deleteall  | --extract  | --list -C | --modify -s SEUSER
      -r RANGE LOGIN ]
```

ဆက်လက်ပြီး yannaing ဆိုတဲ့ User ကို staff_u အနေနဲ့ Mapping တွဲတာကို အောက်က အတိုင်းလုပ်နိုင်ပါတယ်။

```
[root@lab ~]# semanage login -a -s staff_u yannaing
[root@lab ~]# semanage login -l

Login Name           SELinux User         MLS/MCS Range        Service

__default__          unconfined_u         s0-s0:c0.c1023       *
root                 unconfined_u         s0-s0:c0.c1023       *
system_u             system_u             s0-s0:c0.c1023       *
yannaing             staff_u              s0-s0:c0.c1023       *
```

`semanage login -a -s staff_u yannaing` ဆိုတဲ့ Command ကတော့

- `-a` ဆိုတာ User အသစ်ထည့်မယ်ဆိုတာကို ရည်ညွှန်းပါတယ်။
- `-s staff_u` ဆိုတာ အသစ်ထည့်မယ့် User ကို 'staff_u' ဆိုတဲ့ Role နဲ့တကွ ပေါင်းထည့်မယ်လို့ ရည်ညွှန်းပါတယ်။
- yannaing ဆိုတာကတော့ ဒီ List ထဲကို အသစ်ထည့်မယ့် Linux User နာမည်ပါ။

တကယ်လို့ yannaing ကို sysadm_u အဖြစ် Mapping ပြောင်းလဲပေးချင်ရင်တော့ အောက်ကအတိုင်း -a နေရာမှာ -m ကို အစားထိုးအသုံးပြုပြီး Modify ပြုလုပ်နိုင်ပါတယ်။

```
[root@lab ~]# semanage login -m -s sysadm_u yannaing
[root@lab ~]# semanage login -l

Login Name           SELinux User         MLS/MCS Range        Service

__default__          unconfined_u         s0-s0:c0.c1023       *
root                 unconfined_u         s0-s0:c0.c1023       *
system_u             system_u             s0-s0:c0.c1023       *
yannaing             sysadm_u             s0-s0:c0.c1023       *
```

ထည့်သွင်းထားတဲ့ User ကို ပြန်ဖျက်ချင်ရင်တော့ `-d` နဲ့ ဖျက်ပစ်လို့ရပါတယ်။

```
[root@lab ~]# semanage login -d yannaing
[root@lab ~]# semanage login -l

Login Name           SELinux User         MLS/MCS Range        Service

__default__          unconfined_u         s0-s0:c0.c1023       *
root                 unconfined_u         s0-s0:c0.c1023       *
system_u             system_u             s0-s0:c0.c1023       *
```

Confining Network Ports

Service တွေ Application တွေဟာ Network အသုံးပြုတဲ့အခါ သက်ဆိုင်ရာ Protocol နဲ့ Port တွေကို အသုံးပြုရပါတယ်။ ဘယ် Label နဲ့ Run နေတဲ့ Service တွေက ဘယ် Network Port တွေကို ဘယ် Protocol နဲ့အသုံးပြုတာကို ခွင့်ပြုမယ် ခွင့်မပြုဘူးဆိုတဲ့ Permission ကို SELinux က ထိန်းချုပ်ပေးနိုင်ပါတယ်။
ဥပမာအားဖြင့် SELinux ရဲ့ သတ်မှတ်ချက်တွေထဲမှာ "http" ဆိုတဲ့စာလုံးပါတာတွေကို စွဲထုတ်ကြည့်ရအောင်။

```
[root@lab ~]# semanage port -l | grep http
http_cache_port_t              tcp      8080, 8118, 8123, 10001-10010
http_cache_port_t              udp      3130
http_port_t                    tcp      80, 81, 443, 488, 8008, 8009, 8443, 9000
pegasus_http_port_t            tcp      5988
pegasus_https_port_t           tcp      5989
```

Output အရ `http_port_t` ဟာ Port နံပါတ်တွေဖြစ်တဲ့ 80, 81, 443, 488, 8008, 8009, 8443, 9000ဆိုတဲ့ နံပါတ်တွေကို အသုံးပြုလို့ရပါတယ်။ SELinux ရဲ့ Policy တွေထဲမှာလည်း `httpd_t` ဆိုတဲ့ Label ရှိတဲ့ Service တစ်ခုဟာ `http_port_t` ကို Socket အနေနဲ့ Bind လုပ်ပြီး အသုံးပြုလို့ရတယ် ဆိုပြီးလည်း ရေးထားတဲ့အတွက် ပြောရရင် ဒီ သတ်မှတ်ချက်ဟာ Web Server တွေအတွက်ဖြစ်ပါတယ်။
ဆက်လက်ပြီး `http_port_t` ထဲကို TCP Port နံပါတ် 4000 ကို အောက်ကအတိုင်း ပေါင်းထည့်ကြည့်ပါမယ်။

```
[root@lab ~]# semanage port -a -p tcp -t http_port_t 4000
[root@lab ~]# semanage port -l | grep http_port_t
http_port_t                    tcp      4000, 80, 80, 81, 443, 488, 8008, 8009, 8443,
9000
pegasus_http_port_t            tcp      5988
```

`semanage port -a -p tcp -t http_port_t 4000` ဆိုတဲ့ Command မှာတော့

- `-a` ဆိုတာ Port နံပါတ်အသစ်ပေါင်းထည့်မယ်လို့ ရည်ညွှန်းပါတယ်
- `-p` tcp အသစ်ပေါင်းထည့်မယ့် Port နံပါတ်ဟာ TCP Protocol ကို အသုံးပြုမယ်လို့ ရည်ညွှန်းပါတယ်
- `-t http_port_t` အသစ်ပေါင်းထည့်မယ့် Port နံပါတ်ကို http_port_t ဆိုတဲ့ Label ရဲ့ အသုံးပြုခွင့်ပေးမယ်လို့ ရည်ညွှန်းပါတယ်။
- 4000 ကတော့ အသစ်ပေါင်းထည့်မယ့် Port နံပါတ်ဖြစ်ပါတယ်။

Web Server အနေနဲ့ Port နံပါတ် 4000 ကို အသုံးပြုလို့ရသင့်ပါပြီ။ သေချာအောင် စမ်းသပ်ဖို့အတွက် Web Server ကို Port နံပါတ် 4000 နဲ့ အောက်က အတိုင်း စမ်းသပ်ကြည့်ပါမယ်။

```
[root@lab ~]# sed -i 's/Listen 80/Listen 4000/g' /etc/httpd/conf/httpd.conf
[root@lab ~]# systemctl start httpd
[root@lab ~]# netstat -nlp | grep httpd
tcp6       0      0 :::4000                 :::*                    LISTEN
10914/httpd
unix  2      [ ACC ]     STREAM     LISTENING     184164   10915/httpd
/run/httpd/cgisock.10914
```

ဒါဆိုရင်တော့ Apache HTTP Server ဟာ Error ကင်းကင်းနဲ့ TCP Port နံပါတ် 4000 ကိုအသုံးပြုပြီး Run နေပါပြီ။ ဆက်လက်ပြီး `http_port_t` မှာ သတ်မှတ်ထားတဲ့ Port နံပါတ် 4000 ကို အောက်ကအတိုင်း ဖျက်ပါမယ်။

```
[root@lab ~]# semanage port -d -p tcp -t http_port_t 4000
[root@lab ~]# semanage port -l | grep http_port_t
http_port_t                    tcp      80, 80, 81, 443, 488, 8008, 8009, 8443, 9000
pegasus_http_port_t            tcp      5988
```

ဒါဆိုရင်တော့ Apache Web Server ဟာ Port နံပါတ် 4000 ကို အသုံးပြုလို့မရသင့်တော့ပါဘူး။ ဒါကြောင့် Apache HTTP Server ကို Restart ချကြည့်ရင် Error တက်ရတော့မှာပါ။

```
[root@lab ~]# systemctl restart httpd
Job for httpd.service failed because the control process exited with error code.
See "systemctl status httpd.service" and "journalctl -xe" for details.
```

Apache HTTP Server ဟာ Port နံပါတ် 4000 ကို အသုံးပြုလို့မရတော့တဲ့အတွက် Error တက်သွားပါပြီ။ ဒါပေမယ့် မပြီးသေးပါဘူး။ SELinux က အသုံးပြုခွင့်ကို ငြင်းပယ်လိုက်တာဆိုရင် သေချာပေါက် Error Log ထွက်ရမှာပါ။ အဲဒီ Error Log ကိုသွားကြည့်ရအောင်။

```
[root@localhost ~]# tail /var/log/audit/audit.log | grep denied
type=AVC msg=audit(1546091433.194:409): avc:  denied  { name_bind } for  pid=11273
comm="httpd" src=4000 scontext=system_u:system_r:httpd_t:s0
tcontext=system_u:object_r:unreserved_port_t:s0 tclass=tcp_socket permissive=0
```

Log ကတော့ အရှည်ကြီးပဲ။ တစ်ပိုင်းစီ ရှင်းပြပေးပါမယ်။

- `type=AVC` ဆိုတာက ကျွန်တော်တို့ အစောပိုင်းက ပြောခဲ့တဲ့ Access Vector Cache (AVC) အမျိုးအစား Log ဖြစ်ကြောင်း ပြတာပါ။

- `msg=audit(1546091433.194:409)` အပိုင်းက Message ဖြစ်ပါတယ်။ အထဲမှာ ၂ ပိုင်းပါဝင်ပါတယ်။ Linux Timestamp ရယ် Unique ID ရယ်ဖြစ်ပါတယ်။ (Timestamp:ID) ဖြစ်တဲ့အတွက် အဲဒီ ၂ ပိုင်းကို ':' နဲ့ ခွဲထားတာမြင်ရမှာပါ။
- denied ဆိုတာကတော့ တားမြစ်လိုက်တာပါ။
- name_bind ကတော့ Network Socket ကိုအသုံးပြုတဲ့ အကြောင်းအရာဖြစ်ပါတယ်။
- pid=11273 ဆိုတာကတော့ ပြုလုပ်တဲ့ Process ရဲ့ Process ID ကို ဖေါ်ပြထားတာဖြစ်ပါတယ်။
- comm="httpd" ကတော့ Process ရဲ့ နာမည်ဖြစ်ပါတယ်။ သက်ဆိုင်ရာ လုပ်တဲ့ (Subject) ရဲ့ Process ကို စတင်ခဲ့တဲ့ Command (Parameter များမပါ) ဆိုရင်လည်းမမှားပါဘူး။
- src=4000 ကတော့ ကျွန်တော်တို့ သုံးချင်တဲ့ Port နံ့ပါတ်ကိုရည်ညွှန်းတာပါ။ ဒီနေရာမှာ တခြားဖြစ်နိုင်တာတွေကတော့ name= တို့ path= တို့ capacity= တို့လည်းဖြစ်နိုင်ပါတယ်။ အလုပ်ခံရမယ့် Object or Target ပေါ်မူတည်ပြီးကွဲပြားနိုင်ပါတယ်။
- `scontext=system_u:system_r:httpd_t:s0` ဆိုတာကတော့ လုပ်တဲ့ `Process` ရဲ့ `Security Context` ပဲဖြစ်ပါတယ်။ အလွယ်အနေနဲ့ဆိုရင်တော့ `Source Context` သို့မဟုတ် `Subject Context` လို့ မှတ်သားနိုင်ပါတယ်။
- `tcontext=system_u:object_r:unreserved_port_t:s0` ဆိုတာကတော့ အလုပ်ခံရတဲ့ အရာရဲ့ `Security Context` ပဲဖြစ်ပါတယ်။ အလွယ်အနေနဲ့ဆိုရင်တော့ `Target Context` လို့ မှတ်ထားနိုင်ပါတယ်။
- tclass=tcp_socket ကတော့ အလုပ်ခံရတဲ့ အရာရဲ့ အမျိုးအစား Class ကိုဖေါ်ပြထားတာပါ။ ဆုိုင်းဆိုရင်တော့ TCP Network Socket ကို ရည်ညွှန်းထားပါတယ်။ တကယ်လို့ Object ရဲ့ အမျိုးအစားက File ဆိုရင်တော့ tclass=file လို့ပြမှာပါ။
- နောက်ဆုံးက permissive=0 ကတော့ ဒီ Log ကို မှတ်သားခဲ့တဲ့အချိန်က SELinux ဟာ Permissive Mode မှာရှိနေလား မရှိနေဘူးလားဆိုတာကို မှတ်သားလိုက်တာပါ။ permissive=0 ဖြစ်တဲ့အတွက် Permissive Mode မဟုတ်ဘူးလို့ ပြောရမှာပါ။ Permissive Mode ကိုပြောင်းထားမယ်ဆိုရင်တော့ permissive=1 ဖြစ်နေပါလိမ့်မယ်။

Overriding Predefined Policy

`semanage port -l | grep http` လို့ခေါ်လိုက်ရင် `http_port_t` အသုံးပြုခွင့်ရှိတဲ့ Port တွေထဲမှာ TCP Port 80 နဲ့ 443 အပြင် 81, 488, 8008, 8009, 8443, 9000 စတဲ့Port နံပါတ်တွေ ပါဝင်နေတာတွေ့ရမှာပါ။ မလိုအပ်တဲ့ Port တွေထဲက ဥပမာ Port 9000 ကို ဖျက်ကြည့်ရအောင်။

```
[root@lab ~]# semanage port -d -p tcp -t http_port_t 9000
ValueError: Port tcp/9000 is defined in policy, cannot be deleted
```

Policy ထဲမှာ Define လုပ်ထားတဲ့အတွက် ဖျက်လို့မရဘူးဆိုတဲ့ Error တက်ပါတယ်။ ဒါဆို လုံးဝ ဖျက်လို့မရတော့ဘူးတဲ့လား ဆိုရင်တော့ မဟုတ်ပါဘူး။ ကျွန်တော်တို့ အဓိက လုပ်ချင်တာဟာ `httpd_t` ဆိုတဲ့ Label ရှိတဲ့ Process တွေကို `http_port_t` ကသုံးနိုင်တဲ့ Port တွေထဲက TCP Port နံပါတ် 9000 ကို အသုံးပြုခွင့် ပိတ်ချင်တာပါ။ ဒါဆိုရင် အဲဒီ TCP Port 9000 ကို တခြား Type နဲ့ တွဲပေးလိုက်ရင်တော့ အဲဒီ Port ကို သုံးလို့ရတော့မှာမဟုတ်ပါဘူး။ အလုံခြုံဆုံနည်းလမ်းနဲ့ဆိုရင်တော့ အောက်ပါအတိုင်းလုပ်နိုင်ပါတယ်။

```
[root@lab ~]# semanage port -l | grep unreserved
unreserved_port_t              tcp      61001-65535, 1024-32767
unreserved_port_t              udp      61001-65535, 1024-32767
[root@lab ~]# semanage port -m -p tcp -t unreserved_port_t 9000
[root@lab ~]# semanage port -l | grep unreserved
unreserved_port_t              tcp      9000, 61001-65535, 1024-32767
unreserved_port_t              udp      61001-65535, 1024-32767
```

TCP Port နံပါတ် 9000 ကို `unreserved_port_t` ဆိုတဲ့ Type နဲ့ တွဲပေးလိုက်တာပါ။ ဒါဆိုရင် Web Server အနေနဲ့ တကယ်သုံးလို့မရတော့တာလားဆိုတာ ဆက်ကြည့်ရအောင်။ Web Server အသုံးပြုမယ့် Port နံပါတ်ကို ပြောင်းပြီး Restart ချကြည့်ပါမယ်။
(ခုနက Web Server ရဲ့ Listen လုပ်ထားတဲ့ Port ကို 4000 ပြောင်းထားပါတယ်။ ပုံမှန်ဆိုရင်တော့ 80 ဖြစ်ပါတယ်။)

```
[root@lab ~]# sed -i 's/Listen 4000/Listen 9000/g' /etc/httpd/conf/httpd.conf
[root@lab ~]# systemctl restart httpd
Job for httpd.service failed because the control process exited with error code. See
"systemctl status httpd.service" and "journalctl -xe" for details.
```

Web Server တော့ Error တက်သွားပါပြီ။ Web Server ဟာ ကျွန်တော်တို့ ပြောင်းလဲသတ်မှတ်လိုက်တဲ့ `unreserved_port_t` ကို အသုံးပြုဖို့ကြိုးစားတဲ့အတွက် SELinux က ခွင့်မပြုပဲ Error တက်တာလားဆိုတာ သေချာအောင် Error Log ကိုပါ ဆက်ကြည့်ရအောင်။

```
[root@lab ~]# tail -f /var/log/audit/audit.log | grep denied
type=AVC msg=audit(1546246777.049:10518): avc:  denied  { name_bind } for  pid=19966
comm="httpd" src=9000 scontext=system_u:system_r:httpd_t:s0
tcontext=system_u:object_r:unreserved_port_t:s0 tclass=tcp_socket
```

ဒါဆိုရင်တော့ သေချာသွားပါပြီ။ ဒီလိုနည်းနဲ့ Policy ထဲမှာ ကြိုတင်သတ်မှတ်ထားပြီး ဖျက်လို့မရတဲ့ Port တွေကို ပြောင်းလဲသတ်မှတ်ပေးပြီး Permission မရအောင် ထိန်းချုပ်နိုင်ပါတယ်။

SELinux Booleans!

SELinux ကို ၂၀၀၃ ခုနှစ် မတ်လ လောက်ထဲက Kernel Version 2.6 မှာ စတင်အသုံးပြုလာခဲ့တာ ၂၀၁၉ ခုနှစ် ဆိုရင် ၁၅ နှစ်ကျော်ပြီး ၁၆ နှစ်တောင် ဖြစ်တော့မှာပါ။ Community ထဲက လူတွေအနေနဲ့ SELinux ကို အသုံးပြုရတာ အခက်အခဲတွေအများအပြားတွေ့ခဲ့ရပြီးသားဖြစ်ပြီး Report တွေလည်း အများကြီးပဲ ရှိခဲ့ပါတယ်။ ဒါကြောင့် SELinux ကို Maintain လုပ်တဲ့သူတွေအနေနဲ့ တူညီတဲ့အခက်အခဲတွေကို စုပြီး ဘယ် အခက်အခဲတော့ ကြုံလာတဲ့အခါ ဘယ် Boolean လေးကို ဖွင့်ပေးပြီး အမြန်ဆုံး ဖြေရှင်းပေးတာမျိုး ပြုလုပ်နိုင်အောင် SELinux အတွင်းမှာ Boolean Feature တစ်ခုကို ထပ်ထည့်ခဲ့ပါတယ်။ ထည့်သွင်းထားတဲ့ Boolean တွေ ထဲက "httpd" ဆိုတဲ့စာလုံးလေးပါတဲ့ Boolean တွေကို အောက်က အတိုင်း ထုတ်ကြည့်ရအောင်။

```
[root@lab ~]# semanage boolean -l | grep httpd
awstats_purge_apache_log_files (off  ,  off)  Determine whether awstats can purge httpd
log files.
httpd_anon_write                (off  ,  off)  Allow Apache to modify public files used
for public file transfer services. Directories/Files must be labeled
public_content_rw_t.
httpd_builtin_scripting         (on   ,   on)  Allow httpd to use built in scripting
(usually php)
httpd_can_check_spam            (off  ,  off)  Allow http daemon to check spam
httpd_can_connect_ftp           (off  ,  off)  Allow httpd to act as a FTP client
connecting to the ftp port and ephemeral ports
httpd_can_connect_ldap          (off  ,  off)  Allow httpd to connect to the ldap port
httpd_can_connect_mythtv        (off  ,  off)  Allow http daemon to connect to mythtv
httpd_can_connect_zabbix        (off  ,  off)  Allow http daemon to connect to zabbix
httpd_can_network_connect       (off  ,  off)  Allow HTTPD scripts and modules to
connect to the network using TCP.
httpd_can_network_connect_cobbler (off  ,  off)  Allow HTTPD scripts and modules to
connect to cobbler over the network.
httpd_can_network_connect_db    (off  ,  off)  Allow HTTPD scripts and modules to
connect to databases over the network.
httpd_can_network_memcache      (off  ,  off)  Allow httpd to connect to memcache server
httpd_can_network_relay         (off  ,  off)  Allow httpd to act as a relay
httpd_can_sendmail              (off  ,   on)  Allow http daemon to send mail
httpd_dbus_avahi                (off  ,  off)  Allow Apache to communicate with avahi
service via dbus
httpd_dbus_sssd                 (off  ,  off)  Allow Apache to communicate with sssd
service via dbus
httpd_dontaudit_search_dirs     (off  ,  off)  Dontaudit Apache to search dirs.
httpd_enable_cgi                (on   ,   on)  Allow httpd cgi support
httpd_enable_ftp_server         (off  ,  off)  Allow httpd to act as a FTP server by
listening on the ftp port.
httpd_enable_homedirs           (off  ,  off)  Allow httpd to read home directories
httpd_execmem                   (off  ,  off)  Allow httpd scripts and modules
execmem/execstack
httpd_graceful_shutdown         (off  ,  off)  Allow HTTPD to connect to port 80 for
graceful shutdown
httpd_manage_ipa                (off  ,  off)  Allow httpd processes to manage IPA
content
httpd_mod_auth_ntlm_winbind     (off  ,  off)  Allow Apache to use mod_auth_ntlm_winbind
httpd_mod_auth_pam              (off  ,  off)  Allow Apache to use mod_auth_pam
httpd_read_user_content         (off  ,  off)  Allow httpd to read user content
httpd_run_ipa                   (off  ,  off)  Allow httpd processes to run IPA helper.
httpd_run_preupgrade            (off  ,  off)  Allow Apache to run preupgrade
httpd_run_stickshift            (off  ,  off)  Allow Apache to run in stickshift mode,
not transition to passenger
```

```
httpd_serve_cobbler_files      (off  ,  off)  Allow HTTPD scripts and modules to server
cobbler files.
httpd_setrlimit                (off  ,  off)  Allow httpd daemon to change its resource
limits
httpd_ssi_exec                 (off  ,  off)  Allow HTTPD to run SSI executables in the
same domain as system CGI scripts.
httpd_sys_script_anon_write    (off  ,  off)  Allow apache scripts to write to public
content, directories/files must be labeled public_rw_content_t.
httpd_tmp_exec                 (off  ,  off)  Allow Apache to execute tmp content.
httpd_tty_comm                 (off  ,  off)  Unify HTTPD to communicate with the
terminal. Needed for entering the passphrase for certificates at the terminal.
httpd_unified                  (off  ,  off)  Unify HTTPD handling of all content
files.
httpd_use_cifs                 (off  ,  off)  Allow httpd to access cifs file systems
httpd_use_fusefs               (off  ,  off)  Allow httpd to access FUSE file systems
httpd_use_gpg                  (off  ,  off)  Allow httpd to run gpg
httpd_use_nfs                  (off  ,  off)  Allow httpd to access nfs file systems
httpd_use_openstack            (off  ,  off)  Allow httpd to access openstack ports
httpd_use_sasl                 (off  ,  off)  Allow httpd to connect to  sasl
httpd_verify_dns               (off  ,  off)  Allow Apache to query NS records
```

ထွက်လာတဲ့ Output ထဲမှာတော့ ဘယ်ဘက်ဆုံး ကော်လံကတော့ SELinux ရဲ့ Boolean နာမည်ဖြစ်ပါတယ်။ အလယ်က လက်သည်းကွင်းထဲမှာတော့ ၂ ပိုင်းပေါ်နေပါတယ်။ ဘယ်ဘက်အပိုင်းက "on" သို့မဟုတ် "off" ကတော့ လက်ရှိအခြေအနေ ဖွင့်ထားလား ပိတ်ထားလား ဆိုတာဖြစ်ပြီး ညာဘက်ကတော့ SELinux ရဲ့ Default အခြေအနေဖြစ်ပါတယ်။ ညာဘက်ဆုံးအခြမ်းကတော့ အဲဒီ သက်ဆိုင်ရာ Boolean က ဘာအတွက်လဲဆိုတာ ရှင်းပြထားပါတယ်။ Boolean တွေကို အောက်က အတိုင်း `getsebool` ဆိုတဲ့ Command နဲ့လည်း ထုတ်ကြည့်နိုင်ပါသေးတယ်။

```
[root@lab ~]# getsebool -a | grep httpd
httpd_anon_write --> off
httpd_builtin_scripting --> on
httpd_can_check_spam --> off
httpd_can_connect_ftp --> off
httpd_can_connect_ldap --> off
httpd_can_connect_mythtv --> off
httpd_can_connect_zabbix --> off
httpd_can_network_connect --> off
httpd_can_network_connect_cobbler --> off
httpd_can_network_connect_db --> off
httpd_can_network_memcache --> off
httpd_can_network_relay --> off
httpd_can_sendmail --> off
httpd_dbus_avahi --> off
httpd_dbus_sssd --> off
httpd_dontaudit_search_dirs --> off
httpd_enable_cgi --> on
httpd_enable_ftp_server --> off
httpd_enable_homedirs --> off
httpd_execmem --> off
httpd_graceful_shutdown --> off
httpd_manage_ipa --> off
httpd_mod_auth_ntlm_winbind --> off
httpd_mod_auth_pam --> off
httpd_read_user_content --> off
httpd_run_ipa --> off
httpd_run_preupgrade --> off
httpd_run_stickshift --> off
httpd_serve_cobbler_files --> off
httpd_setrlimit --> off
```

```
httpd_ssi_exec --> off
httpd_sys_script_anon_write --> off
httpd_tmp_exec --> off
httpd_tty_comm --> off
httpd_unified --> off
httpd_use_cifs --> off
httpd_use_fusefs --> off
httpd_use_gpg --> off
httpd_use_nfs --> off
httpd_use_openstack --> off
httpd_use_sasl --> off
httpd_verify_dns --> off
```

ဒါပေမယ့် `getsebool` ကတော့ Boolean တွေရဲ့ လက်ရှိအခြေအနေကိုပဲ ပြတာဖြစ်ပြီး SELinux ရဲ့ Default Value ကို ဖေါ်ပြတာ မပါဝင်သလို ရှင်းပြချက်တွေလည်း မပါဝင်ပါဘူး။ ဒါကြောင့် `getsebool` ကိုတော့ SELinux ကို စတင်လေ့လာတဲ့သူတွေ စ မသုံးသင့်ပါဘူး။ တကယ်တတ်ကျွမ်းသွားပြီးမှသာ အမြန်အနေနဲ့ အသုံးပြုလို့ ရပါတယ်။ ဆက်လက်ပြီး တကယ့်အပြင်က အဖြစ်အပျက်လေးတွေနဲ့ယှဉ်တွဲပြီး SELinux Boolean တွေကို ဘယ်လိုအချိန်မျိုးမှာ အသုံးပြုလဲဆိုတာ ကြည့်ရအောင်။

Web Server isn't sending emails!

SELinux ရဲ့ Default Policy အရ `httpd_t` Type ကိုအသုံးပြုတဲ့ Web Server တွေဟာ Email ပို့ခွင့်ကို ခွင့်မပြုထားပါဘူး။ အကယ်၍များ ကိုယ့်ရဲ့ Web Site ကနေ Email များပို့လွှတ်တဲ့အခါ SELinux က ခွင့်မပြုပဲ ကာထားတဲ့အတွက် ဘယ်လိုမှ Email တွေထွက်မှာမဟုတ်ပါဘူး။ ဒါကြောင့် SELinux Boolean တွေကို တချက် ကြည့်ရအောင်။ အပေါ်က ဖေါ်ပြခဲ့တဲ့ Output ထဲမှာ `httpd_can_sendmail` ဆိုတာ ရှိပါတယ်။ ဘာလေးလဲဆိုတာကြည့်ကြည့်ပါမယ်။

```
[root@lab ~]# semanage boolean -l | grep httpd_can_sendmail
httpd_can_sendmail             (off  ,  off)  Allow http daemon to send mail
```

"Allow http daemon to send mail" လို့ ပြနေတဲ့အတွက် Web Server ကနေ Email ပို့ချင်ရင်တော့ ဒီ Boolean လေးကို "on" ဖြစ်အောင် အောက်ကအတိုင်း လုပ်ပေးရမှာပါ။

```
[root@lab ~]# getsebool -a | grep httpd_can_sendmail
httpd_can_sendmail --> off
[root@lab ~]# setsebool -P httpd_can_sendmail on
[root@lab ~]# getsebool -a | grep httpd_can_sendmail
httpd_can_sendmail --> on
```

`setsebool` ဆိုတဲ့ Command မှာ `-P` ဆိုတာကတော့ Persistant ကိုပြောချင်တာပါ။ နားလည်အောင်ပြောရရင် တော့ "Permanent" ဖြစ်ပါတယ်။ အမြဲတမ်းအတွက် ပြောင်းလိုက်တာပါ။ အကယ်လို့ `-P` ဆိုတာလေးမပါခဲ့ရင် System Reboot ဖြစ်တဲ့အခါ SELinux ရဲ့ Default အတိုင်း ပြန်ဖြစ်သွားပါလိမ့်မယ်။ ခုလိုမျိုး `httpd_can_sendmail` ဆိုတဲ့ Boolean လေးဟာ "on" ဖြစ်သွားပြီဆိုရင်တော့ Web Server ကနေ Email တွေ ပို့ဖို့အတွက် SELinux ကနေ ပိတ်ပင်ထားတာ မရှိတော့ပါဘူး။

မှတ်သားထားရန်။ ။ ဒါကြောင့် SELinux နဲ့ပတ်သက်ပြီး အခက်အခဲတစ်ခုခု ကြုံတွေ့ရတိုင်း "Boolean" တွေကို ပထမဦးစွာ သွားရောက်စစ်ဆေးကြည့်ဖို့ အကြံပေးလိုပါတယ်။

Practical Security #1 - File Contexts & Web Server Security

သာမန် Web Server တစ်ခုပေါ်မှာရှိတဲ့ Web Content တွေအပေါ်မှာ SELinux ရဲ့ File Context တွေကို အသုံးပြုပြီး Security ချုပ်ကိုင်ပုံကို စတင်ပြသပေးပါမယ်။ Security ထိန်းချုပ်ပုံတွေပြောပြတဲ့အခါ နားလည်မှု အလွယ်ကူဆုံးဖြစ်စေဖို့ Web Content တွေအတွက်တော့ အလွယ်ကူဆုံးဖြစ်တဲ့ Wordpress ကိုပဲ ရွေးချယ်လိုက်ပါတယ်။

Pre-Configurations or Environment

ဆက်လက်မပြုလုပ်မီ ကျွန်တော်ပြသပေးမယ့် System နဲ့ပတ်သက်တာတွေကို စတင်ပြောပေးပါမယ်။

- Operating System: CentOS 7.5.1804 (Core)
- Web Server: Apache HTTP Server 2.4.6
- PHP Engine: PHP 5.4.16
- MariaDB: 5.5.60
- SELinux Targeted Policy: selinux-policy-targeted version 3.13.1
- SELinux Mode/Type: Enforcing/Targeted
- PHP Web Shell (Demo as Hacker): Madspot Shell Version 1.0

Operation

စစချင်း Apache HTTP Server ရဲ့ Run နေတဲ့အနေအထားလေးကို Process ထဲမှာ သွားကြည့်ရအောင်။

```
[root@lab wordpress]# ps -auxZ | grep http
system_u:system_r:httpd_t:s0    root     12597  0.0  0.6 314564 12140 ?        Ss
12:33   0:00 /usr/sbin/httpd -DFOREGROUND
system_u:system_r:httpd_t:s0    apache   12632  0.0  0.3 314696  6164 ?        S
12:33   0:00 /usr/sbin/httpd -DFOREGROUND
system_u:system_r:httpd_t:s0    apache   12633  0.0  0.3 314696  6164 ?        S
12:33   0:00 /usr/sbin/httpd -DFOREGROUND
system_u:system_r:httpd_t:s0    apache   12634  0.0  0.3 314696  6164 ?        S
12:33   0:00 /usr/sbin/httpd -DFOREGROUND
system_u:system_r:httpd_t:s0    apache   12635  0.0  0.3 314696  6164 ?        S
12:33   0:00 /usr/sbin/httpd -DFOREGROUND
system_u:system_r:httpd_t:s0    apache   12636  0.0  0.3 314696  6164 ?        S
12:33   0:00 /usr/sbin/httpd -DFOREGROUND
```

Apache HTTP Server ရဲ့ Run နေတဲ့ Process မှာရှိတဲ့ Label ကိုကြည့်မယ်ဆိုရင် `system_u:system_r:httpd_t:s0` လေးကို တွေ့ရမှာပါ။ သတိပြုရမှာက ကျွန်တော်တို့ဟာ SELinux ရဲ့ "Targeted Environment" Security Policy ကို အသုံးပြုနေတာဖြစ်တဲ့အတွက် `httpd_t` လေးကို သတိပြုမိရမှာဖြစ်ပါတယ်။ အဲဒါလေးကို စိတ်ထဲမှာမှတ်ထားပါ။

ဆက်လက်ပြီး Wordpress ရဲ့ File လေးတွေ အနေအထားကို သွားကြည့်ရအောင်။

```
[root@lab wordpress]# pwd
/var/www/html/wordpress
[root@lab wordpress]# ls -Z
-rw-r--r--. apache apache unconfined_u:object_r:httpd_sys_content_t:s0 index.php
-rw-r--r--. apache apache unconfined_u:object_r:httpd_sys_content_t:s0 license.txt
-rw-r--r--. apache apache unconfined_u:object_r:httpd_sys_content_t:s0 readme.html
-rw-r--r--. apache apache unconfined_u:object_r:httpd_sys_content_t:s0 wp-activate.php
drwxr-xr-x. apache apache unconfined_u:object_r:httpd_sys_content_t:s0 wp-admin
-rw-r--r--. apache apache unconfined_u:object_r:httpd_sys_content_t:s0 wp-blog-
header.php
-rw-r--r--. apache apache unconfined_u:object_r:httpd_sys_content_t:s0 wp-comments-
post.php
-rw-r--r--. apache apache unconfined_u:object_r:httpd_sys_content_t:s0 wp-config-
sample.php
drwxr-xr-x. apache apache unconfined_u:object_r:httpd_sys_rw_content_t:s0 wp-content
-rw-r--r--. apache apache unconfined_u:object_r:httpd_sys_content_t:s0 wp-cron.php
drwxr-xr-x. apache apache unconfined_u:object_r:httpd_sys_content_t:s0 wp-includes
-rw-r--r--. apache apache unconfined_u:object_r:httpd_sys_content_t:s0 wp-links-opml.php
-rw-r--r--. apache apache unconfined_u:object_r:httpd_sys_content_t:s0 wp-load.php
-rw-r--r--. apache apache unconfined_u:object_r:httpd_sys_content_t:s0 wp-login.php
-rw-r--r--. apache apache unconfined_u:object_r:httpd_sys_content_t:s0 wp-mail.php
-rw-r--r--. apache apache unconfined_u:object_r:httpd_sys_content_t:s0 wp-settings.php
-rw-r--r--. apache apache unconfined_u:object_r:httpd_sys_content_t:s0 wp-signup.php
-rw-r--r--. apache apache unconfined_u:object_r:httpd_sys_content_t:s0 wp-trackback.php
-rw-r--r--. apache apache unconfined_u:object_r:httpd_sys_content_t:s0 xmlrpc.php
```

Wordpress ရဲ့ File လေးတွေမှာ ကပ်ပါနေတဲ့ File Context လေးတွေမှာ `unconfined_u:object_r:httpd_sys_content_t:s0` ဆိုတဲ့ Security Context Label တွေနဲ့ `unconfined_u:object_r:httpd_sys_rw_content_t:s0` ဆိုတာ ၂ ခုတွေ့ပါလိမ့်မယ်။ သေချာကြည့်ပါ အဲဒီ ၂ ခုမှာ "rw" ပါတာနဲ့ မပါတာနဲ့ ကွာပါတယ်။ သေချာကြည့်ရင် မတူတဲ့ ၂ မျိုးဟာ Type နေရာမှာ `httpd_sys_content_t` ရယ် `httpd_sys_rw_content_t` ရယ်ပါ။
ပြောရရင် ရှေ့ကပ်ထားတဲ့ httpd_ ဆိုတဲ့နေရာလေးက Apache HTTP Server ရဲ့ Process မှာလည်းတွေ့ရသလို ဒီ Web Content တွေဖြစ်တဲ့ Wordpress File လေးတွေမှာလည်း တွေ့ရမှာဖြစ်ပါတယ်။ ထပ်ပြီး နားလည်လွယ်အောင်ပြောရရင် Apache HTTP Server ဟာ "Subject" (လုပ်တဲ့ကောင်) ဖြစ်ပြီး ဟောဒီ File လေးတွေဟာ "Object" (ခံရမယ့်ကောင်) လေးတွေဖြစ်ပါတယ်။ Apache HTTP Server ကနေ ဟောဒီ File လေးတွေကို "Read" "Write" "Execute" တွေ လုပ်နိုင်ပါတယ်။ လုပ်ခွင့်ပေးမှာလားမပေးဘူးလားဆိုတာကိုတော့ DAC Permission နဲ့တကွ Apache HTTP Server မှာကော File လေးတွေမှာကော ရှိတဲ့ Security Context (Label) လေးတွေကို နှိုင်းယှဉ်ပြီး SELinux ရဲ့ Targeted Security Policy နဲ့အညီ အောက်ပါအတိုင်း ထိန်းချုပ်မှာပါ။ ဘာမှထပ်ပြီး မလုပ်ရင်တောင် အခုအတိုင်းတော်တော်လေးလုံခြုံနေပါပြီ။

ဒါကြောင့် Apache HTTP Server ဟာ သူနဲ့သက်ဆိုင်ရာ Label များကို Read သို့မဟုတ် Read & Write Access ရရှိမှာဖြစ်ပါတယ်။

ဒါဆိုရင် ဘာဆက်လုပ်လို့ရမလဲ။ လုံခြုံရေးပိုမိုကောင်းမွန်အောင် အလွယ်ဆုံးနည်းနဲ့ ဘယ်လိုမျိုးဆက်ချုပ်ကိုင်မလဲ?

အလွယ်ကူဆုံးဆိုရင်တော့ Wordpress တစ်ခု စ တင် Deploy လုပ်တဲ့အခါ Theme တွေ Module တွေကို Installation ပြုလုပ်မှာ သေချာပါတယ်။ ဒါပေမယ့် အားလုံး ကြိုက်ပြီ၊ ထပ်ထည့်စရာ ဖြုတ်စရာမရှိတော့ဘူးဆိုတဲ့အခါ wp-content ဆိုတဲ့ Folder အောက်က "themes" နဲ့ "plugins" ဆိုတဲ့ Folder တွေရဲ့ Label တွေကို အောက်မှာပြထားသလို `httpd_sys_content_t` ပြောင်းလိုက်ရုံပါပဲ။ အကျိုးဆက်ကတော့ Apache Web Server အနေနဲ့ အဲဒီ သက်ဆိုင်ရာ File/Folder တွေကို ဖတ်လို့ပဲရတော့မှာပါ။ အကြောင်းအမျိုးမျိုးနဲ့ Wite လုပ်လို့ရမှာ မဟုတ်တော့ပါဘူး။

```
[root@lab wp-content]# pwd
/var/www/html/wordpress/wp-content
[root@lab wp-content]# ls -lZ
-rw-r--r--. apache apache unconfined_u:object_r:httpd_sys_rw_content_t:s0 index.php
drwxr-xr-x. apache apache unconfined_u:object_r:httpd_sys_rw_content_t:s0 plugins
drwxr-xr-x. apache apache unconfined_u:object_r:httpd_sys_rw_content_t:s0 themes
drwxr-xr-x. apache apache system_u:object_r:httpd_sys_rw_content_t:s0 uploads
[root@lab wp-content]# chcon -R -t httpd_sys_content_t plugins/ themes/ index.php
[root@lab wp-content]# ls -lZ
-rw-r--r--. apache apache unconfined_u:object_r:httpd_sys_content_t:s0 index.php
drwxr-xr-x. apache apache unconfined_u:object_r:httpd_sys_content_t:s0 plugins
drwxr-xr-x. apache apache unconfined_u:object_r:httpd_sys_content_t:s0 themes
drwxr-xr-x. apache apache system_u:object_r:httpd_sys_rw_content_t:s0 uploads
```

မှတ်ချက်။ ။ အကယ်၍များ စာဖတ်သူတို့အသုံးပြုနေတဲ့ Environment မှာ vHost တွေ တခြား Configuration တွေရှိရင်လည်း File Context တွေနဲ့ပတ်သက်တဲ့ အကြောင်းအရာရဲ့ အဓိက အချက်က ဒီဟာကို သိထားရင်ကို ရပါပြီ။

Protected Impacts should a breach occur

ဒီလောက်လေးလုပ်ရုံနဲ့တင် တခြား Security တွေထဲက အလိုလျောက် ဖြေရှင်းပြီးဖြစ်ပြီးသားတွေ ရှိပါတယ်။ အဲဒါတွေကတော့

- Backdoor ကို ဘယ်လိုမှ ထုတ်လို့မရတော့ပါဘူး။ ဘာလို့လဲဆိုတော့ အကြောင်းအမျိုးမျိုးနဲ့ တစ်စုံတစ်ယောက်ကနေ ဖေါက်ထွင်းဝင်ရောက်တဲ့အခါ သူရောက်ရှိ/ရရှိမှာကတော့ Web Server ရဲ့ Privilege တွေကိုပဲရရှိမှာပါ။ သူလုပ်သမျှဟာ Web Server က လုပ်တဲ့အတိုင်းပဲဖြစ်မှာပါ။ ဒါကြောင့် Web Server က အသုံးပြုလို့ရတဲ့ Port နံပါတ်တွေကိုပဲ အသုံးပြုလို့ရမှာပါ။ အခြား Port နံပါတ်တွေကို အသုံးပြုပြီး Backdoor/BackConnect လုံးဝလုပ်လို့ရတော့မှာမဟုတ်ပါဘူး။
- ဖေါက်ထွင်းဝင်ရောက်သူအနေနဲ့ Home Directories ထဲက Data တွေကို လုံးဝဖတ်လို့မရတော့ပါဘူး။ ဘာလို့လဲဆိုတော့ ဖေါက်ထွင်းဝင်ရောက်နိုင်တဲ့သူဟာ သူလုပ်သမျှသည် `httpd_t` အနေနဲ့လုပ်ရမှာသာ ဖြစ်တဲ့အတွက် `user_home_t` ဆိုတဲ့ Label ရှိတဲ့ File/Folder တွေကို ဝင်ရောက်မကြည့်ရှုနိုင်တော့ပါ။
- Rooting လို့ခေါ်တဲ့ apache User ကနေ root User ရဲ့ Privilege တွေရရှိအောင် ပြုလုပ်ခြင်းဟာလည်း အလုပ်လုပ်တော့မှာမဟုတ်ပါဘူး။ `httpd_t` Label ရရှိထားတဲ့ ဖေါက်ထွင်းဝင်ရောက်နိုင်တဲ့သူဟာ အချို့သော system libraries တွေကို ခေါ်ယူအသုံးပြုနိုင်မှာမဟုတ်သလို Compiler တွေက လိုအပ်တဲ့

Library တွေကိုလည်း ယူမသုံးနိုင်တဲ့အတွက် Executable Program တစ်ခုခုကို အသုံးပြုပြီး Priviledge Escalation ပြုလုပ်ဖို့တော့ လုံးဝမဖြစ်နိုင်တော့တဲ့ အခြေအနေပါပဲ။

Practical Security* #2 - *Generating Custom Security Policy

SELinux ရဲ့ Security Policy တွေထဲမှာ မပါဝင်တဲ့ Server တွေကို ထိုင်ကြတဲ့အခါမှာ SELinux က နားမလည်တဲ့ (သို့မဟုတ်) သူ့ Security Policy ထဲမှာ မပါဝင်တဲ့အချက်အလက်တွေ ရှိလာတဲ့အခါ လိုအပ်တာတွေကို Generate လုပ်ပြီး ထပ်ထည့်လို့ရပါတယ်။ ဒီသဘောသဘာဝကို အကောင်းဆုံးပြသဖို့အတွက် Zabbix Server ထိုင်တဲ့အခါ ကြုံတွေ့ရတဲ့ အခက်အခဲလေးကို ထည့်သွင်းဖေါ်ပြလိုက်ပါတယ်။

Pre-Configurations or Environment

ဆက်လက်မပြုလုပ်မီ ကျွန်တော်ပြသပေးမယ့် System နဲ့ပတ်သက်တာတွေကို စတင်ပြောပေးပါမယ်။

- Operating System: CentOS 7.5.1804 (Core)
- Web Server: Apache HTTP Server 2.4.6
- PHP Engine: PHP 5.4.16
- MariaDB: 5.5.60
- SELinux Targeted Policy: selinux-policy-targeted version 3.13.1
- SELinux Mode/Type: Enforcing/Targeted
- PHP Web Shell (Demo as Hacker): Madspot Shell Version 1.0
- Zabbix Server: 4.0.3 (with MySQL Support)

Seeking the Problem

Zabbix Server ကို Install ပြုလုပ်ပြီးတဲ့အခါမှာတော့ Zabbix Web Panel ပေါ်မှာ "Zabbix Server is running" ဆိုတဲ့နေရာမှာ "no" လို့ပြနေပါတယ်။ အဓိကကတော့ အကြောင်းအမျိုးမျိုးကြောင့် Zabbix Server (Engine) နဲ့ Web Server နဲ့ မချိတ်မိတာကြောင့် တက်နေတဲ့ Error ဖြစ်ပါတယ်။ ပထမအဆင့်အနေနဲ့ `setenforce 0` လို့ ရိုက်ထည့်ကြည့်ပြီး `systemctl restart zabbix-server` လို့ရိုက်လိုက်ရင် ချိတ်မိသွားတာတွေ့ရမှာပါ။ ဒါပေမယ့် `setenforce 1` နဲ့ SELinux ကို Enforcing Mode ပေးလိုက်တဲ့အခါ Error ပြန်တက်သွားတာတွေ့ရပါလိမ့်မယ်။ ဒါဟာ Zabbix Server နဲ့ Web Server ကြားက Permission တစ်ခုခုကို SELinux က ပိတ်ပင်ထားတာကြောင့်ဆိုတာ အလွယ်သိရှိနိုင်မှာပါ။ ဆက်ပြီး SELinux ရဲ့ Audit Log တွေကို သွားကြည့်ရအောင်။ Audit Log ထဲမှာတော့ အောက်က Error ၃ ကြောင်းနဲ့ ဆင်ဆင်တူတာတွေ တက်နေမှာပါ။

```
[root@lab ~]# tail -f /var/log/audit/audit.log | grep denied
type=AVC msg=audit(1546241364.121:614): avc:  denied  { name_connect } for
pid=2894 comm="httpd" dest=10051 scontext=system_u:system_r:httpd_t:s0
tcontext=system_u:object_r:zabbix_port_t:s0 tclass=tcp_socket
type=AVC msg=audit(1546241368.076:618): avc:  denied  { unlink } for  pid=6290
comm="zabbix_server" name="zabbix_server_preprocessing.sock" dev="tmpfs" ino=21616
scontext=system_u:system_r:zabbix_t:s0
tcontext=system_u:object_r:zabbix_var_run_t:s0 tclass=sock_file
type=AVC msg=audit(1546241379.009:626): avc:  denied  { unlink } for  pid=6357
comm="zabbix_server" name="zabbix_server_alerter.sock" dev="tmpfs" ino=21606
scontext=system_u:system_r:zabbix_t:s0
tcontext=system_u:object_r:zabbix_var_run_t:s0 tclass=sock_file
```

- name_connect နဲ့ comm=httpd နဲ့ tclass=tcp_socket ပါတဲ့ Error Log ကတော့ အကြမ်းအားဖြင့် httpd_t Label ရှိတဲ့ Web Server ကနေ zabbix_port_t Label ရှိတဲ့ TCP Socket ကို ချိတ်မရတဲ့ Error Log ပါပဲ။ သေချာပြောရရင် Web Server ကနေ Zabbix ကို ချိတ်မရတာပါ။
- unlink ရယ် comm=zabbix_server ရယ် ပါဝင်တဲ့ ကျန်တဲ့ ၂ ခုကတော့ Zabbix Server ကနေ သက်ဆိုင်ရာ Sock File တွေကို unlink() System Call ပြုလုပ်တာမှာ မရပဲဖြစ်နေတာပါ။

Troubleshooting

ပထမတစ်ခုကတော့ Boolean လောက်နဲ့ ရှင်းရတာ လွယ်ကူနိုင်ပါတယ်။ ဒါကြောင့် ဆက်လက်ပြီး Boolean တွေကို သွားကြည့်ပါမယ်။

```
[root@lab ~]# semanage boolean -l | grep zabbix
zabbix_can_network             (off  ,   off)  Allow zabbix to can network
httpd_can_connect_zabbix       (off  ,   off)  Allow httpd to can connect zabbix
```

တွေ့ပါပြီဗျာ။ `httpd_can_connect_zabbix` ဆိုတာဟာ Web Server ကနေ Zabbix ကို ချိတ်ဆက်ဖို့အတွက် ခွင့်ပြု/မပြု ကို သတ်မှတ်တဲ့ Boolean လေးပါ။ ဒါကြောင့် `setsebool -P httpd_can_connect_zabbix on` ဆိုတဲ့ Command လေးနဲ့ ဖွင့်ပေးလိုက်ရင်တော့ Log ထဲမှာ unlink Error ၂ ခုပဲ ပတ်ချာလည်တက်နေတာကို တွေ့ရမှာပါ။ အဲဒီ Error ၂ ခုကိုတော့ Custom Policy ထုတ်ပြီး ဆက် ရှင်းကြည့်ပါမယ်။
ဒါကြောင့် မရှင်းရသေးတဲ့ Error Log လေးနှစ်ခုကို zabbixerror.log ဆိုတဲ့ ဖိုင်လေးထဲကို copy/paste အရင်ထည့်ပါမယ်။ ပြီးရင်တော့ audit2allow ဆိုတဲ့ Command လေးကို ဆက်သုံးကြည့်ပါမယ်။

```
[root@lab ~]# audit2allow -w -i zabbixerror.log
type=AVC msg=audit(1546241368.076:618): avc:  denied  { unlink } for  pid=6290
comm="zabbix_server" name="zabbix_server_preprocessing.sock" dev="tmpfs" ino=21616
scontext=system_u:system_r:zabbix_t:s0
tcontext=system_u:object_r:zabbix_var_run_t:s0 tclass=sock_file

        Was caused by:
                Missing type enforcement (TE) allow rule.

                You can use audit2allow to generate a loadable module to allow
this access.

type=AVC msg=audit(1546241379.009:626): avc:  denied  { unlink } for  pid=6357
comm="zabbix_server" name="zabbix_server_alerter.sock" dev="tmpfs" ino=21606
scontext=system_u:system_r:zabbix_t:s0
tcontext=system_u:object_r:zabbix_var_run_t:s0 tclass=sock_file

        Was caused by:
                Missing type enforcement (TE) allow rule.

                You can use audit2allow to generate a loadable module to allow
this access.
```

`audit2allow -w -i zabbixerror.log` ဆိုတဲ့ Command လေးနဲ့ ကြည့်လိုက်တာပါ။ Tag လေးတွေကို ရှင်းပြရရင်တော့ -w ဆိုတာ Error တွေရဲ့ အကြောင်းအရင်းကို ရှင်းပြခိုင်းတာဖြစ်ပြီး `-i zabbixerror.log` ဆိုတာတော့ ခုနက မှတ်ထားတဲ့ Error တွေထည့်ထားတဲ့ File လေးကို ရည်ညွှန်းလိုက်တာပါ။ အချုပ်အနေနဲ့ ပြန်ပြောရရင်တော့ zabbixerror.log ဆိုတဲ့ ဖိုင်ထဲမှာ ရှိတဲ့ Error log တွေဟာ ဘာကြောင့် ဖြစ်ရတာလဲဆိုတာကို ရှင်းပြခိုင်းတာပါ။

Output မှာတော့ Error တစ်ကြောင်းချင်းစီရဲ့ အောက်မှာ "Was caused by" ဆိုတာလေးနဲ့ ဖြစ်ကြောင်း အကျဉ်းကို ဖော်ပြပေးထားပါတယ်။ အဓိကကတော့ TE ရဲ့ ခွင့်ပြုတဲ့ စည်းကမ်းချက် မရှိတာပါပဲ။ ဒါကြောင့် ဆက်ပြီး Custom Policy ကို ထုတ်ပါမယ်။

```
[root@lab ~]# audit2allow -i zabbixerror.log -M zabbixerror
******************** IMPORTANT ***********************
To make this policy package active, execute:

semodule -i zabbixerror.pp

[root@lab ~]# ls -l
total 16
-rw-------. 1 root root 1555 Jun 21  2018 anaconda-ks.cfg
-rw-r--r--. 1 root root  524 Dec 31 14:19 zabbixerror.log
-rw-r--r--. 1 root root  961 Dec 31 14:25 zabbixerror.pp
-rw-r--r--. 1 root root  193 Dec 31 14:25 zabbixerror.te
[root@lab ~]# cat zabbixerror.te

module zabbixerror 1.0;

require {
        type zabbix_var_run_t;
        type zabbix_t;
        class sock_file unlink;
}

#============= zabbix_t ==============
allow zabbix_t zabbix_var_run_t:sock_file unlink;
```

`audit2allow -i zabbixerror.log -M zabbixerror` ဆိုတဲ့ Command မှာ -M ဆိုတာကတော့ Security Module တစ်ခု ကို Generate ထုတ်ခိုင်းတာပဲဖြစ်ပါတယ်။ အဲဒီ Command ရိုက်လိုက်တဲ့အခါ zabbixerror.pp နဲ့ zabbixerror.te ဆိုတဲ့ ဖိုင် ၂ ခု ထွက်လာတာ မြင်ရပါလိမ့်မယ်။ zabbixerror.pp ကိုတော့ Policy Profile File လို့ ခေါ်ပါတယ်။ သူ့ကိုတော့ဖွင့်ကြည့်လို့မရပေမယ့် zabbixerror.te ဆိုတဲ့ Type Enforcement Rule ရေးထားတဲ့ ဖိုင်လေးကိုတော့ cat နဲ့ ထုတ်ကြည့်နိုင်ပါတယ်။ ပါဝင်တဲ့ အကြောင်းအရာတွေကို သေချာ စစ်စေချင်ပါတယ်။ မလိုအပ်တာတွေ ပါမသွားအောင်ဖြစ်ပါတယ်။ ပြီးရင်တော့ `semodule -i zabbixerror.pp` ဆိုတဲ့ Command နဲ့ SELinux ရဲ့ Security Policy တွေထဲကို Module တစ်ခုအနေနဲ့ ရေးထည့်ပါမယ်။ ပြီးရင်တော့ တကယ် ဝင်သွားလားဆိုတာ List ထုတ်ကြည့်ပါမယ်။

```
[root@lab ~]# semodule -l | grep zabbixerror
zabbixerror     1.0
```

ဒါဆိုရင်တော့ ဝင်သွားပါပြီ။ ဒါကြောင့် `systemctl restart zabbix-server` ဆိုတဲ့ Command နဲ့ Zabbix Server ကို Restart ချကြည့်ပါမယ်။ ဒါပေမယ့် ရဦးမှာမဟုတ်ပါဘူး။ အခြား Error ထပ်တက်နေပါလိမ့်မယ်။ နောက်ထပ်တစ်ကြိမ် အောက်က Error အတိုင်း ထပ်မြင်ရပါလိမ့်မယ်။

```
type=AVC msg=audit(1546243011.968:1429): avc:  denied  { create } for  pid=16442
comm="zabbix_server" name="zabbix_server_preprocessing.sock"
scontext=system_u:system_r:zabbix_t:s0
tcontext=system_u:object_r:zabbix_var_run_t:s0 tclass=sock_file
```

ဒါကြောင့် နောက်တစ်ကြိမ် Security Policy ထပ်ထုတ်ပါမယ်။ ဒီတစ်ခါမှာတော့ အဲဒီ Log ကို zabbixerror2.log ထဲကိုထည့်ပြီး ဆက်လုပ်မှာပါ။

```
[root@lab ~]# audit2allow -i zabbixerror2.log -M zabbixerror2
[root@lab ~]# semodule -i zabbixerror2.pp
[root@lab ~]# systemctl restart zabbix-server
```

ဒီတစ်ကြိမ်မှာတော့ Error Log တွေ ပြောင်းသွားပြီး Zabbix Server ကနေ Connection တစ်ခုခု စလုပ်တိုင်း Error တက်မှာဖြစ်တဲ့အတွက် Error တွေ အများကြီး Screen ပေါ်မှာ ပြေးနေပါလိမ့်မယ်။ အဓိကကတော့ အောက်က Error ၂ မျိုးထဲ Log ထဲမှာ ပတ်ချာရမ်းနေတာပါ။

```
type=AVC msg=audit(1546243209.950:7283): avc:  denied  { connectto } for
pid=17252 comm="zabbix_server" path="/run/zabbix/zabbix_server_preprocessing.sock"
scontext=system_u:system_r:zabbix_t:s0 tcontext=system_u:system_r:zabbix_t:s0
tclass=unix_stream_socket
type=AVC msg=audit(1546243209.961:7284): avc:  denied  { connectto } for
pid=17221 comm="zabbix_server" path="/run/zabbix/zabbix_server_alerter.sock"
scontext=system_u:system_r:zabbix_t:s0 tcontext=system_u:system_r:zabbix_t:s0
tclass=unix_stream_socket
```

ဒါကြောင့် အဲဒီ ၂ ကြောင်းကို zabbixerror3.log ထဲပစ်ထည့်ပြီး `audit2allow -i zabbixerror3.log -M zabbixerror3` ဆိုတဲ့ Command နဲ့ နောက်တစ်ကြိမ် Security Policy ထပ်ထုတ်ပြီး `semodule -i zabbixerror3.pp` နဲ့ Module ကိုထပ်ထည့်ပါမယ်။ ပြီးရင်တော့ Zabbix Server ကို Restart ချတဲ့အခါ အဆင်ပြေပြေ Error ကင်းကင်းနဲ့ Run သွားတာ တွေ့ရပါလိမ့်မယ်။

Summary

ဒါကြောင့် SELinux Custom Policy ကို Generate ထုတ်မယ်ဆိုရင်တော့ တစ်ခါထဲနဲ့ မပြီးတာ တွေ့ရမှာပါ တစ်ဆင့်ပြီးတစ်ဆင့်၊ Error တစ်မျိုးပြီးတစ်မျိုး ဆင့်ကာ ဆင့်ကာ ရှင်းရမှာပဲဖြစ်ပါတယ်။

Overall Conclusion

နိဂုံးအနေနဲ့ ပြောရရင်တော့ အခုထိ ကျွန်တော် ရေးသားဖေါ်ပြထားတာတွေဟာ SELinux ရဲ့ အခြေခံ တွေနဲ့ အလွယ်ကူဆုံး "Targeted Environment" (သို့မဟုတ်) "Type Enforcement" Security Policy တွေနဲ့ ပတ်သက်တာတွေချည့်ပဲဖြစ်ပါတယ်။ ဒီအကြောင်းအရာတွေကို ဒီလိုမျိုး ကောင်းကောင်း အသုံးပြုနိုင်ပြီး ကြေညက်ပြီဆိုပါက ဆက်လက်လေ့လာစရာရှိပါသေးတယ်။ ဆက်လက်လေ့လာသင့်တဲ့ အကြောင်းအရာကတော့ Military Grade အဆင့်ရှိတဲ့ Multi-Level Security/Multi-Category Security ပဲဖြစ်ပါတယ်။ အပေါ်မှာ ဖေါ်ပြခဲ့သလိုပဲ MLS/MCS ဟာ Label တွေထဲက Sensitivity Level နဲ့ Category Level ကို အဓိက အသုံးပြုပြီး အရမ်းကောင်းတဲ့ Bella-Padula Model နဲ့လည်း ကိုက်ညီတဲ့ Security Model တစ်ခုဖြစ်ပါတယ်။ MLS/MCS ကို နားလည်လာပြီဆိုရင်တော့ Containerization/Virtualization/IaaS Cloud Computing တွေမှာ သုံးမယ့် Security model ဖြစ်တဲ့ sVirt ကို ဆက်လက် လေ့လာနိုင်ပြီပဲ ဖြစ်ပါတယ်။
ဒီလောက် အသေးစိတ် အနုစိတ် ထိန်းချုပ်လို့ရတဲ့ Security Model ကြီးတစ်ခုလုံးဖြစ်တဲ့ "SELinux" ဟာ Free and Open Source အနေနဲ့ ရရှိနေတာပါ။ ထို့အပြင် "RedHat", "CentOS", "Fedora" စတဲ့ Linux Distribution တွေမှာ SELinux ဟာ TE Type ကို "Enforcing" Mode အနေနဲ့ ပါလာပါတယ်။ ဒါတောင် ကျွန်တော်တို့ဟာ ဘာလို့များ အသုံးပြုပါက ကောင်းကျိုးအများကြီးရှိတဲ့ SELinux ကိုဆက်လက် မလေ့လာပဲ Disable လုပ်ပစ်ကြမှာပါလဲ။

SELinux နဲ့ပတ်သက်ပြီး ကျွန်တော့်အနေနဲ့ ခုလောက်ထိ ရေးသားဖေါ်ပြချက်တွေအပေါ်မှာ ကျေနပ်အားရကြလိမ့်မယ်လို့လည်း ယူဆပါတယ်။