DANGEROUS FILES - LINUX SYSTEM

I like Linux, I really do, but I also know that it needs to be setup right to help secure it (see here). Now part of that securing process is making sure that malicious users cannot easily elevate their access rights on the system. We will be going through some of the more common ways attackers try this while writing a script to help us check it in an easier manner.

SUID Files
These are files where the UID bit has been set. This means that this program will execute with the same rights as the stated owner of the program. So for example..

```
-rwsr-xr-x  1 root root 31308 Apr  7  2005 /bin/ping6
```

The *ping6* program will execute with the rights of the *root* user no matter who executes it. Now if an attacker can exploit a vulnerability in one of these files, they could possibly get it to execute a whole bunch of other commands as well. Let's also start with our shell script (I am assuming you have a folder called */admin/bin*, otherwise chose another one)..

```
#touch /admin/bin/chk-files
#chmod 700 /admin/bin/chk-files
```

Now that we have our file ready lets add in the first function, Cut and paste the following into your *chk-files* file..

```
suid () {
 echo "CHECKING FOR SUID FILES"
 echo "~~~~~~~~~~~~~~~~~~~~~~~"
 echo ""
 find $SRCH \( -perm -4000 \) \
   -type f \
   -exec file {} \; \
   -exec ls -la {} \; \
   -exec echo "" \;
 read -p "HIT ANY KEY TO CONTINUE.."
}
```

This is a function, by itself it will not do anything, but when we are finished with the script it will be vital.

SGID Files
These files are exactly the same as the SUID files, except that the GID has been set, not the UID. This means that the program will execute with the rights of the stated group. For example..

```
-rwxr-sr-x  1 root root 11373 May 25  2005 /sbin/netreport
```

The *netreport* file will execute as the group root. The risks of these files are the same as for SUID files. Lets add our next function to our script file, cut and paste the following..

```
sgid () {
 echo "CHECKING FOR SGID FILES"
 echo "~~~~~~~~~~~~~~~~~~~~~~~"
 echo ""
 find $SRCH \( -perm -2000 \) \
   -type f \
   -exec file {} \; \
   -exec ls -la {} \; \
   -exec echo "" \;
 read -p "HIT ANY KEY TO CONTINUE.."
}
```

World Writable Directories
Both SUID and SGID files are generally system files with preset permissions, but your users can also give an attacker a foothold. The first way that can happen is with a directory that anyone on the system can write to. This means if an attacker can find sensitive documents or scripts in such a folder then they can tamper with these

files. Lets add a function to check for such directories to our script, cut and paste the following..

```
wwd () {
 echo "CHECKING FOR WORLD WRITABLE DIRECTORIES"
 echo "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"
 echo ""
 find $SRCH \( -perm -o+w \) \
   -type d \
   -exec ls -lad {} \;
 read -p "HIT ANY KEY TO CONTINUE.."
}
```

World Writable Files
Much like the above problem, except that world writable files are also dangerous as attackers can still tamper with them to perhaps get them to execute commands to help elevate the access rights of the attacker. So lets also check for these files, cut and paste the following..

```
wwf () {
 echo "CHECKING FOR WORLD WRITABLE FILES"
 echo "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"
 echo ""
 find $SRCH \( -perm -o+w \) \
   -type f \
   -exec ls -la {} \;
 read -p "HIT ANY KEY TO CONTINUE.."
}
```

Full Permission Files
The last type of badly setup file is a file that has given everyone full rights, meaning that the *rwx* settings have been enabled for the user, the group and for everyone else. Again, an attacker could use such a file to perhaps escalate their rights on the system or to cause damage. There is no reason for such files. Lets add a function to check for these, cut and paste the following..

```
fpf () {
 echo "CHECKING FOR FILES WITH FULL PERMISSIONS"
 echo "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"
 echo ""
 find $SRCH \( -perm -u+rwx -perm -g+rwx -perm -o+rwx \) \
    -type f \
    -exec ls -la {} \;
 read -p "HIT ANY KEY TO CONTINUE.."
}
```

Fixing it
All of the above files and directories can pose a threat to a system from a wily attacker. If you really want to lock down your system, you should remove or modify these rights. The general linux user will never run *ping* or *netreport*, there is no reason for world writable files or directories and if general access to a command is really needed, then use the *sudo* command to securely allow such access (see here). The first step in being able to give your files the right access levels is to know which files pose a risk. This is where the last part of our script files comes in, cut and paste the following..

```
clear
read -p "WHERE DO YOU WANT TO TEST? : "
SRCH=$REPLY
echo ""

while [ 0 ]
 do
  clear
  echo "CHECKING $SRCH FOR.."
  echo ""
  echo "1- CHECK FOR SUID FILES"
  echo "2- CHECK FOR SGID FILES"
  echo "3- CHECK FOR WORLD WRITABLE DIRECTORIES"
  echo "4- CHECK FOR WORLD WRITABLE FILES"
  echo "5- CHECK FOR FULL PERMISSION FILES"
  echo "x- EXIT"
```

```
  echo ""
  read -p "CHOOSE ONE : "
  CHECK=$REPLY
  echo ""

  case $CHECK in
   1) suid;;
   2) sgid;;
   3) wwd;;
   4) wwf;;
   5) fpf;;
   x|X)clear
       exit;;
   *) echo "bad choice, try again";;
  esac
done
```

What that last bit does is create a menu system which calls the previous functions you put in. Each function is designed to test for a certain type of bad file we have discussed in a folder you specify and all it's subfolders. As always, my shell scripts work for me, please fell free to use them or change them as you see fit.

Final Words
These checks are very simple, but that can really help increase your systems overall security. Also be aware to run these checks on a regular basis as you add new software or as your users add new files. Stopping these easily attacked files and folders will make an attacker work a little harder and thats always a good thing. As always, have fun and learn.