```
Linux Access Control and Exploits
Tue, Sep. 16, 2008

_____________________________________________________________________
Access control in Linux

This lecture involves details of access control in Linux.

See Pollock article and Hacking Linux Exposed for more explanation
and details not covered here.

_____________________________________________________________________
IDs

* Each user has a user id (uid) and belongs to (possibly several) groups
  each of which has a gid.

* The uid and default gid are stored in /etc/passwd. E.g, on my laptop

    root:x:0:0:root:/root:/bin/bash
    bin:x:1:1:bin:/bin:/sbin/nologin
    daemon:x:2:2:daemon:/sbin:/sbin/nologin
    adm:x:3:4:adm:/var/adm:/sbin/nologin
    lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
    sync:x:5:0:sync:/sbin:/bin/sync
    shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
    halt:x:7:0:halt:/sbin:/sbin/halt
    mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
    news:x:9:13:news:/etc/news:
    uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
    ...
    lynux:x:500:500:Franklyn Turbak:/home/lynux:/bin/bash
    gdome:x:501:501:Georgia Dome:/home/gdome:/bin/bash
    cs342:x:502:502:CS342 Account:/home/cs342:/bin/bash

* /etc/group defines groups and lists which users belong to them.

    root:x:0:root
    bin:x:1:root,bin,daemon
    daemon:x:2:root,bin,daemon
    sys:x:3:root,bin,adm
    adm:x:4:root,adm,daemon
    tty:x:5:
    disk:x:6:root
    lp:x:7:daemon,lp
    mem:x:8:
    kmem:x:9:
    wheel:x:10:root
    ...
    lynux:x:500:
    gdome:x:501:
    cs342:x:502:
    cs342sta:x:503:cs342,lynux
    cs342stu:x:504:cs342,gdome,lynux

  Sysadmins can define new groups, e.g. on puma

     faculty (all cs faculty)
     cs111sta (members of the CS111 teaching staff)
```

```
     cs251stu (cs251 students)

* Our networked dept. machines now use LDAP, a database system for users/groups
  that does not use /etc/passwd and /etc/group directly. But you will be
  using /etc/passwd and /etc/group on the machine you administer in E125.

  On puma, you can use the getent command to extract info from the LDAP database
:

    [fturbak@puma ~] getent passwd fturbak
    fturbak:x:708:708:Franklyn Turbak:/home/fturbak:/bin/bash

    [fturbak@puma ~] getent passwd adaigle
    adaigle:x:4281:4282:Amanda Daigle class of 2012:/students/adaigle:/usr/local
/bin/scponly
    # /usr/local/bin/scponly only allows SCP, not login access. For 110/111 stud
ents

    [fturbak@puma ~] getent group cs235stu
    cs235stu:*:4012:cs235,fturbak,rshull,gdome,lbenson,pboettne,ydai,mdobbs,cgre
vet,chamada,
    shamilto,mkierste,slafranc,clopez,mmoore2,sshiplet,asolomon,atang,ktaylor

------------------------------------------------------------------------
Checking/Changing who you are:

* whoami: name associated with current uid
* groups: groups of which current uid is a member
* su <username>: "become" <username>
* su - <usernam>: "become" <username>, using initialization files
* su: "become" root (su = superuser)

------------------------------------------------------------------------
File Permissions in Linux

    [lynux@localhost cs342]$ ls -al handouts
    total 68
    drwxrwx--- 4 lynux lynux  4096 2008-09-12 07:36 .
    drwxrwxr-x 6 lynux lynux  4096 2008-09-02 03:08 ..
    drwxrwx--- 2 lynux lynux  4096 2008-09-02 03:15 course-info
    -rw-rw---- 1 lynux lynux   638 2008-09-09 08:59 linux-commands.txt
    -rw-rw---- 1 lynux lynux 12335 2008-09-12 07:33 os-security.txt
    -rw-rw---- 1 lynux lynux  3073 2008-09-11 21:27 os-security.txt~
    drwxrwx--- 2 lynux lynux  4096 2008-09-09 05:38 security
    [lynux@localhost cs342]$

* How do you read a permission string (e.g. drwxrwxr-x, -rw-rw----)?

  + Leftmost chars:

    - normal file
    d directory
    l link
    s socket

  + Other 9 chars: read (r), write (w), execute (x) permissions for 3 entities:
    3 chars for owner (u=user);
    3 for group (g);
    3 for everyone else (o=other)
```

```
_____________________________________________________________________
What Do Permissions Mean?

  + On file:
    r: can read from file
    w: can write to file
    x: can execute file as a program

  + On directory:
    r: can access directory contents
    w: can add new file and delete existing file
        (even if don't have any permissions on file!)
    x: can list the files in the directory.

_____________________________________________________________________
Interpreting Permissions as Bits

There are actually *12* (not *9*) permission bits in Linux.
From most to least significant:

   * setuid bit
     - on executable program: change effective user id of program (more later)
     - this bit changes "x" to "s" and no "x" to "S"

   * setgid bit
     - on executable program: change effective group id of program (more later)
     - on directory: files/subdirectories inherit group and its permissions from
 directory
     - this bit changes "x" to "s" and no "x" to "S"

   * sticky bit
     - on directory: only owner can delete files in the directory (used in /tmp)
     - this bit changes "x" to "t" and no "x" to "T"

   * user read bit
   * user write bit
   * user execute bit

   * group read bit
   * group write bit
   * group execute bit

   * other read bit
   * other write bit
   * other execute bit

_____________________________________________________________________
Changing File Permissions in Linux

chown changes owner (-R flag performs recursively)
  + Only root can do this

chgrp changes group (-R flag performs recursively)
  + Owner can only change to group it belongs to;
    otherwise root needs to perform.

chown <owner>.<group> is equivalent to
```

```
  chown <owner>
  chgrp <group>

chmod changes permissions (-R flag performs recursively). E.g.:

    [lynux@localhost handouts]$ ls -al os-security.txt
    -rw-rw---- 1 lynux lynux 13290 2008-09-12 07:48 os-security.txt

    [lynux@localhost handouts]$ chmod o+rx os-security.txt
    [lynux@localhost handouts]$ ls -al os-security.txt
    -rw-rw-r-x 1 lynux lynux 13290 2008-09-12 07:48 os-security.txt

    [lynux@localhost handouts]$ chmod g-w os-security.txt
    [lynux@localhost handouts]$ ls -al os-security.txt
    -rw-r--r-x 1 lynux lynux 13290 2008-09-12 07:48 os-security.txt

    [lynux@localhost handouts]$ chmod u+x os-security.txt
    [lynux@localhost handouts]$ ls -al os-security.txt
    -rwxr--r-x 1 lynux lynux 13290 2008-09-12 07:48 os-security.txt

    [lynux@localhost handouts]$ chmod a-wx os-security.txt
    [lynux@localhost handouts]$ ls -al os-security.txt
    -r--r--r-- 1 lynux lynux 13290 2008-09-12 07:48 os-security.txt

    [lynux@localhost handouts]$ chmod 754 os-security.txt
    [lynux@localhost handouts]$ ls -al os-security.txt
    -rwxr-xr-- 1 lynux lynux 13290 2008-09-12 07:48 os-security.txt

    [lynux@localhost handouts]$ chmod u+s os-security.txt
    [lynux@localhost handouts]$ ls -al os-security.txt
    -rwsr-xr-- 1 lynux lynux 13290 2008-09-12 07:48 os-security.txt

    [lynux@localhost handouts]$ chmod g+s os-security.txt
    [lynux@localhost handouts]$ ls -al os-security.txt
    -rwsr-sr-- 1 lynux lynux 13290 2008-09-12 07:48 os-security.txt

    [lynux@localhost handouts]$ chmod 754 os-security.txt
    [lynux@localhost handouts]$ ls -al os-security.txt
    -rwxr-xr-- 1 lynux lynux 13290 2008-09-12 07:48 os-security.txt

    [lynux@localhost handouts]$ chmod 4754 os-security.txt
    [lynux@localhost handouts]$ ls -al os-security.txt
    -rwsr-xr-- 1 lynux lynux 13290 2008-09-12 07:48 os-security.txt

    [lynux@localhost handouts]$ ls -al .
    total 72
    drwxrwxr-x 4 lynux lynux  4096 2008-09-12 08:27 .
    ...

    [lynux@localhost handouts]$ chmod +t .
    [lynux@localhost handouts]$ ls -al .
    total 72
    drwxrwxr-t 4 lynux lynux  4096 2008-09-12 08:27 .
    ...

-----------------------------------------------------------------
The SetUID (SUID) Bit

  Some programs need to use protected/private files --
```

```
  e.g., passwd stores encrypted passwords in /etc/shadow,
  which has permissions that are something like

     -rw------- 1 root root 1554 2008-09-15 05:57 /etc/shadow

  [This is a white lie, but believe it for now]

  How can a regular user change her own password? Because of setuid!

    [lynux@localhost setuid]$ which passwd
    /usr/bin/passwd
    [lynux@localhost setuid]$ ls -al /usr/bin/passwd
    -rwsr-xr-x 1 root root 4730 2008-09-13 08:08 /usr/bin/passwd

  The setUID bit says that while /usr/bin/passwd is running, it will
  have the owner's (in this case root's) UID as its effective UID.
  So it can write to /etc/shadow!

-------------------------------------------------------------------------
Playing with SetUID: A Squirrel program

  Let's create a "squirrel" program owned by lynux that appends into
  a file of lynux named "nest":

    [lynux@localhost setuid]$ ls -al nest
    -rw-rw-r-- 1 lynux lynux 0 2008-09-16 06:39 nest

  The "nest" file is readable by anyone, but only writable by lynux.
  It's initially empty.

  Here's a simple C program squirrel.c:

    ---------------------------------------------------
    /* Append the argument to the file named "nest" */

    /* Include standard library headers */
    #include <stdio.h>
    #include <stdlib.h>

    int main (int argc, char** args) {

      FILE *fp;
      fp=fopen("nest", "a");
      if (fp > 0) {
        if (argc >= 1) fprintf(fp, "%s\n", args[1]);
        fclose(fp);
      } else {
        printf("Unable to open file\n");
      }
    }
    ---------------------------------------------------

  Let's compile and run it.

    gcc -o squirrel squirrel.c
    [lynux@localhost setuid]$ ls -al squirrel
    -rwxrwxr-x 1 lynux lynux 5208 2008-09-16 06:45 squirrel

  Now lynux can add items to the nest ...
```

```
    [lynux@localhost setuid]$ cat nest
    [lynux@localhost setuid]$ squirrel aaa
    [lynux@localhost setuid]$ cat nest
    aaa
    [lynux@localhost setuid]$ squirrel bbb
    [lynux@localhost setuid]$ cat nest
    aaa
    bbb

  ... but poor gdome can't:

    [gdome@localhost setuid]$ ./squirrel ccc
    Unable to open file

  But if lynux makes the file suid ...

    [lynux@localhost setuid]$ chmod u+s squirrel
    [lynux@localhost setuid]$ ls -al squirrel
    -rwsrwxr-x 1 lynux lynux 5208 2008-09-16 06:45 squirrel

  ... then gdome can write to it via squirrel:

    [gdome@localhost setuid]$ cat nest
    aaa
    bbb
    ccc

--------------------------------------------------------------------------
Can We Make Squirrel a Script?

  Does lynux need to write squirrel in C? Why not just use the following
  bash script named "squirrel2"?

    ----------------------------------------
    #!/bin/bash -p

    # squirrel expressed as a bash script
    # The -p option says to pay attention to setuid and setgid bits

    if (($#>=1))
    then
      echo $1 >> nest
    fi
    ----------------------------------------

  Lynux makes this suid and takes it for a spin:

    [lynux@localhost setuid]$ chmod u+s squirrel2

    [lynux@localhost setuid]$ ls -al squirrel2
    -rwsrwxr-x 1 lynux lynux 161 2008-09-16 06:59 squirrel2

    [lynux@localhost setuid]$ squirrel2 ddd

    [lynux@localhost setuid]$ cat nest
    aaa
    bbb
    ccc
```

```
    ddd

  Sadly, gdome can't use it:

    [gdome@localhost setuid]$ ./squirrel2 eee
    ./squirrel2: line 8: nest: Permission denied

  Why? For safety reasons, our version of Linux does *not* allow
  shell scripts to be suid!

--------------------------------------------------------------------
Circumventing the Restriction

  Does this mean we have to write all suid programs in C rather than
  as bash scripts?

  Sort of ... but there's a trick to transform a bash script to
  a C program.

  Here's a C program named "squirrel3.c" that runs the script "squirrel2":

    ----------------------------------------
    int main (int argc, char* argv) {
      execv("squirrel2", argv);
    }
    ----------------------------------------

  Let's compile and run it:

    [lynux@localhost setuid]$ gcc -o squirrel3 squirrel3.c

    [lynux@localhost setuid]$ chmod u+s squirrel3

    [lynux@localhost setuid]$ ls -al squirrel3
    -rwsrwxr-x 1 lynux lynux 4820 2008-09-16 07:08 squirrel3

    [lynux@localhost setuid]$ squirrel3 eee

    [lynux@localhost setuid]$ cat nest
    aaa
    bbb
    ccc
    ddd
    eee

  Even gdome can use squirrel3!

    [gdome@localhost setuid]$ ./squirrel3 fff
    [gdome@localhost setuid]$ cat nest
    aaa
    bbb
    ccc
    ddd
    eee
    fff

  Moral: using C's execv, we can execute a bash script using suid!

--------------------------------------------------------------------
```

```
The Rootshell: A Dangerous SUID Program

  Here's an interesting C program, rootshell.c:

    ------------------------------------------------------------
    #include <stdio.h>

    int main (int argc, char** args) {
      /* rootshell <arg1> ... <argn> acts like /bin/bash -p  <arg1> ... <argn>
*/

      char* newargs[argc + 2];
      int i;
      newargs[0] = "/bin/bash";
      newargs[1] = "-p"; /* Essential for setuid root to work */
      for (i=1;i<argc;i++) {
        newargs[i+1] = args[i];
      }
      newargs[argc + 1] = NULL;  /* Array must be null-terminated */
      execv("/bin/bash", newargs);
    }
    ------------------------------------------------------------

  Suppose lynux compiles it and makes it SUID:

    [lynux@localhost setuid]$ gcc -o rootshell rootshell.c

    [lynux@localhost setuid]$ chmod u+s rootshell

    [lynux@localhost setuid]$ ls -al rootshell
    -rwsrwxr-x 1 lynux lynux 4968 2008-09-16 07:19 rootshell

  Now what if gdome uses it?

    [gdome@localhost setuid]$ ./rootshell
    bash-3.2$ whoami
    lynux
    bash-3.2$ ... go off and do anything as lynux ...

  Oops! This program allows anyone to *become* lynux in a shell!

  If the owner is root, this is called a "root shell".

  It's obviously very dangerous (and beloved by hackers) because
  it allows Elevation of Privilege.

  So dangerous in fact, that we better take away SUID:

    [lynux@localhost setuid]$ chmod u-s rootshell

    [lynux@localhost setuid]$ ls -al rootshell
    -rwxrwxr-x 1 lynux lynux 4968 2008-09-16 07:19 rootshell

  Now gdome can just create a new shell *owned by her*

    [gdome@localhost setuid]$ ./rootshell
    [gdome@localhost setuid]$ whoami # this is a new shell, not the original
    gdome
    [gdome@localhost setuid]$ exit  # exit the new shell
```

```
    exit
    [gdome@localhost setuid]$ # now back in original shell.

_____________________________________________________________________
The SetGID (SGID) Bit

* On executable files, the setGID bit can be used like setUID, except
  it changes the effective *group* ID of the user executing the file.

* On a directory, SGID causes new files/subdirectories to inherit
  permissions of the directory.

    [lynux@localhost download]$ mkdir shared
    [lynux@localhost download]$ chgrp cs342stu shared
    [lynux@localhost download]$ ls -al shared
    total 16
    drwxrwxr-x 2 lynux cs342stu 4096 2008-09-16 08:36 .
    drwxrwxr-x 4 lynux lynux    4096 2008-09-16 08:36 .


    [gdome@localhost ~]$ cd ~lynux/cs342/download/shared
    [gdome@localhost shared]$ ls -al
    total 16
    drwxrwxr-x 2 lynux cs342stu 4096 2008-09-16 08:36 .
    drwxrwxr-x 4 lynux lynux    4096 2008-09-16 08:36 ..

    [gdome@localhost shared]$ touch one

    [gdome@localhost shared]$ ls -al
    total 20
    drwxrwxr-x 2 lynux cs342stu 4096 2008-09-16 08:43 .
    drwxrwxr-x 4 lynux lynux    4096 2008-09-16 08:36 ..
    -rw-rw-r-- 1 gdome gdome       0 2008-09-16 08:43 one

    [gdome@localhost shared]$ chmod g+s .
    chmod: changing permissions of '.': Operation not permitted

    [lynux@localhost download]$ chmod g+s shared
    [lynux@localhost download]$ ls -al shared
    drwxrwsr-x 2 lynux cs342stu 4096 2008-09-16 08:43 shared


    [gdome@localhost shared]$ touch two

    [gdome@localhost shared]$ mkdir sub

    [gdome@localhost shared]$ ls -al
    total 32
    drwxrwsr-x 3 lynux cs342stu 4096 2008-09-16 08:44 .
    drwxrwxr-x 4 lynux lynux    4096 2008-09-16 08:36 ..
    -rw-rw-r-- 1 gdome gdome       0 2008-09-16 08:43 one
    drwxrwsr-x 2 gdome cs342stu 4096 2008-09-16 08:44 sub
    -rw-rw-r-- 1 gdome cs342stu    0 2008-09-16 08:44 two

_____________________________________________________________________
SGID in Practice: CS111 Drop Folders

  /home/cs111/drop:
  used 188 available 5687440
```

```
  drwxrwx--- 2 cs111 cs111 4096 Sep  8 11:17 .
  drwxr-xr-x 6 cs111 cs111 4096 Jul 22 15:03 ..
  lrwxrwxrwx 1 root  root    29 Sep  8 11:17 acroteau -> /students/acroteau/cs11
1/drop
  lrwxrwxrwx 1 root  root    28 Sep  2 20:39 adaigle -> /students/adaigle/cs111/
drop
  ...

  /home/cs111/drop/adaigle:
  used 104 available 11822176
  drwxr-s--- 13 adaigle cs111 4096 Sep  2 20:39 .
  drwxr-x---  4 adaigle cs111 4096 Sep  3 14:39 ..
  drwxr-s---  3 adaigle cs111 4096 Sep  6 01:15 ps01
  drwxr-s---  3 adaigle cs111 4096 Sep 15 23:35 ps02
  ...

  /home/cs111/drop/adaigle/ps01:
  used 24 available 11822176
  drwxr-s---  3 adaigle cs111 4096 Sep  6 01:15 .
  drwxr-s--- 13 adaigle cs111 4096 Sep  2 20:39 ..
  drwxr-sr-x  2 adaigle cs111 4096 Sep  6 01:16 Amanda_ps01

  /home/cs111/drop/adaigle/ps01/Amanda_ps01:
  used 292 available 11822176
  drwxr-sr-x 2 adaigle cs111  4096 Sep  6 01:16 .
  drwxr-s--- 3 adaigle cs111  4096 Sep  6 01:15 ..
  -rw-r--r-- 1 adaigle cs111  9044 Sep  6 01:15 Buggle.class
  -rw-r--r-- 1 adaigle cs111   336 Sep  6 01:15 BuggleException.class
  ...
  -rw-r--r-- 1 adaigle cs111  1599 Sep  6 01:15 Writing.class
  -rw-r--r-- 1 adaigle cs111   157 Sep  6 01:16 Writing.html
  -rw-r--r-- 1 adaigle cs111  2678 Sep  6 01:15 Writing.java

_____________________________________________________________________
The Sticky Bit

* Problem: Any user with write access to directory can delete a file
    from directory, regardless of owner:

    [lynux@localhost shared]$ touch important

    [lynux@localhost shared]$ chmod 700 important

    [lynux@localhost shared]$ ls -al
    total 36
    drwxrwsr-x 3 lynux cs342stu 4096 2008-09-16 09:06 .
    drwxrwxr-x 4 lynux lynux    4096 2008-09-16 08:36 ..
    -rwx------ 1 lynux cs342stu    0 2008-09-16 09:06 important
    -rw-rw-r-- 1 gdome gdome       0 2008-09-16 08:43 one
    drwxrwsr-x 2 gdome cs342stu 4096 2008-09-16 08:44 sub
    -rw-rw-r-- 1 gdome cs342stu    0 2008-09-16 08:44 two

    [gdome@localhost shared]$ rm important
    rm: remove write-protected regular empty file ‘important’? y

    [gdome@localhost shared]$ ls -al
    total 32
    drwxrwsr-x 3 lynux cs342stu 4096 2008-09-16 09:06 .
    drwxrwxr-x 4 lynux lynux    4096 2008-09-16 08:36 ..
```

```
    -rw-rw-r-- 1 gdome gdome       0 2008-09-16 08:43 one
    drwxrwsr-x 2 gdome cs342stu 4096 2008-09-16 08:44 sub
    -rw-rw-r-- 1 gdome cs342stu    0 2008-09-16 08:44 two

* The "sticky bit" on a directory allows users to delete only
  those files owned by them:

    [lynux@localhost shared]$ chmod +t .

    [lynux@localhost shared]$ touch important2

    [lynux@localhost shared]$ chmod 700 important2

    [lynux@localhost shared]$ ls -al
    total 36
    drwxrwsr-t 3 lynux cs342stu 4096 2008-09-16 09:09 .
    drwxrwxr-x 4 lynux lynux    4096 2008-09-16 08:36 ..
    -rwx------ 1 lynux cs342stu    0 2008-09-16 09:09 important2
    -rw-rw-r-- 1 gdome gdome       0 2008-09-16 08:43 one
    drwxrwsr-x 2 gdome cs342stu 4096 2008-09-16 08:44 sub
    -rw-rw-r-- 1 gdome cs342stu    0 2008-09-16 08:44 two

    [gdome@localhost shared]$ rm important2
    rm: remove write-protected regular empty file ‘important2’? y
    rm: cannot remove ‘important2’: Operation not permitted

* This feature is used to protect files in share directory /tmp

-------------------------------------------------------------------------
Password-protecting Web Pages

* Can require an http password on a directory by putting an .htaccess
  file in it. E.g:

    # begin file .htaccess
    AuthUserFile /var/www/htpasswd
    AuthGroupFile /dev/null
    AuthName ByPassword
    AuthType Basic

    <Limit GET>
    require user gdome
    </Limit>
    # end file .htaccess

[show gdome example]

* Some other options are helpful

     allow from .wellesley.edu # allow those from wellesley domain
     allow from 149.130. # allow those from wellesley machines
     require valid-user # allow any user with http password, not just gdome

[show with gdome]

* Can set an http password for gdome by executing the following as root:

    # The first times (-c creates the password file; -m uses MD5 hashing)
    htpasswd -c -m /var/www/htpasswd gdome
```

```
    # Subsequent times
    htpasswd -m /var/www/htpasswd gdome

* Warning: http passwords are sent in the clear! (We'll see this later in semest
er.)

------------------------------------------------------------------------
Real-Life Access-Control Design

  Want to have http-password-protected course pset solutions
  that are viewable by staff but not students.

  How to achieve this goal?

------------------------------------------------------------------------
Exploits: What do Hackers Want?

* Your bandwidth: stepping stone to other attachs, hide tracks in your machine,
  part of botnet

* Your CPU (e.g. crack passwords)

* Your disk: pirated software (warez), porn, ...

* Your data: credit card number, SSN, bank records

So hackers want to own / root your machine.

------------------------------------------------------------------------
Password Attacks

- use passwords from keystroke logger
- find passwords "sent in the clear" on network (especially wireless)
- find passwords stored unprotected on computer
   (perhaps permissions on .bash_history or some other file are not set correctl
y).
- crack passwords (e.g. John the ripper) -- must be able to read password file
- social engineering: get people to divulge passwords, look at postits on comput
er
- dumpster diving

------------------------------------------------------------------------
Password Protection

- use long passwords, not in dictionary
- use password cracking programs to discover weak passwords
- hide encrypted passwords in /etc/shadow
- password expiration

------------------------------------------------------------------------
Elevation of Privilege Attacks:

* If can't take over root directly, try to break in as unprivilged user
  and elevate privilege.

  mycat with suid
  rootshell with suid
  rootshell with sgid
```

```
* path attacks

* code injection attacks

* symbolic link attack
```