Open Source Security Tools

Saurabh Barjatiya

2012-01-21 Sat

Contents

Note: Commands and configuration files mentioned below are with respect to Cent-OS / Fedora / Red-hat based systems. Same content should apply to other distributions with minor changes in command options, location of configuration files, etc.

1 Introduction

Open-source security tools can be categorized based on their functionality into following broad categories:

- Tools that help in securing system from its users
- Tools that help in securing system from network (LAN / Internet)
- Tools that help in securing Internet usage from given system

This document has been divided into sections with respect to categories of tools as described above. Discussion for each tool is kept brief due to time constraints. Information on resources through which tools can be learned in-depth are provided at many places, to facilitate further learning of mentioned tools.

2 Securing own system

This section contains information on tools and techniques that can help in securing system from its own users. It has been further divided into various sub-sections to categorize large number of tools that help in securing systems.

2.1 Preventing changes

One of the objectives of securing is to prevent unauthorized changes. In this sub-section tools that help in preventing unauthorized changes are discussed. Information on `sudo` which helps in providing granular access to users is also mentioned.

2.1.1 Permissions and Ownership

One of the simple ways of preventing changes in Discretionary Access Control (DAC) systems such as Linux is by creating users and groups and setting proper file permissions. In Linux each file or process is owned by a user and a group, and permissions on file can be configured to specifically allow / deny access to it by user owner, group owner or others.

There are three types of permissions for each file / directory:

- Read (`r`, `4`)
- Write (`w`, `2`)
- Execute (`x`, `1`)

Each of these three permissions are applied to three different categories of users:

- User owner (`u`)
- Group owner (`g`)
- Others (`o`)

Various commands that can be used in manipulating permissions and ownership are:

- `ls -l` can be used to check present ownership and permissions on various files and directories
- `chmod` can be used to change permissions. `chmod` command is very flexible as supports permissions supplied in numeric as well as alphabetic format.
- `chown` can be used to change current user owner and group owner of a file or directory.
- `chgrp` can be used to change just group owner without modifying user owner
- `usermod` can be used to modify primary and set of secondary groups of a given user

Linux man pages can be used to learn about each of above mentioned commands. Detailed description of Linux file permissions is available at

- http://www.zzee.com/solutions/linux-permissions.shtml
- http://www.tuxfiles.org/linuxhelp/filepermissions.html
- http://www.tuxfiles.org/linuxhelp/fileowner.html

2.1.2 Sudo access

Many times there are more than one users and administrators of a system. Keeping track of user and administrator activities in multi-user systems is not easy. To complicate things further sometimes users need to run commands like mount, umount, ping (!), which require root permissions for them to work. In most cases these problems are resolved by making root owner of given program and giving sticky execute permissions to given file. But sometimes it is desired that only specific user can run given command and not all the users of the system. When such fine-grained access control requirements are present, `sudo` can be used to get desired results.

Most Linux distributions come with `sudo` installed. Administrators can use `visudo` command to edit `/etc/sudoers` file (which should not be edited using editors directly) and give permissions to specific users to run specific commands. Users who are granted permission for specific command can login as themselves and use `sudo` command to get work done.

Since `sudo` allows administrators to give permissions to normal users to run command as root. It can be (*and is often*) used as root-kit to gain root access by malicious users on compromised systems. Hence even if `sudo` is not used, it is important to understand how it works and monitor `/etc/sudoers` file for changes, to prevent unwanted access grants to users, enabling them to preform operations as `root`.

Detailed configuration of sudo can be learned from

- http://ubuntuforums.org/showthread.php?t=1132821

2.1.3 SELinux

- Introduction to SELinux
  SELinux provides Mandatory Access Control (MAC) based security to Linux systems. SELinux assigns a security context to each file and process. Activities performed by a process are allowed or denied based on configured policy and assigned context. SELinux also supports concept of **sensitivity** and **category**.

  - Example 1 - Apache with SELinux
    For example apache executable (`/usr/sbin/httpd`) on most systems is assigned SELinux context with type `httpd_exec_t`. Due to this whenever apache binary file is executed running process gets context with type `httpd_t`. Now default SELinux policy in most cases has list of ports (`http_port_t`) that a `httpd_t` process is allowed to listen to (TCP 80, 443, 488, 8008, 8009, 8443). If we change apache configuration in `/etc/httpd/conf/httpd.conf` file and change listening port from 80 (`Listen 80`) to 90 (`Listen 90`) and try to start apache as root user then it will fail. Note that in DAC root user never gets 'access denied' or 'permission denied' messages and can do virtually whatever he / she desired. But with SELinux root user cannot start apache on port 90 because of various SELinux contexts and policies.

    Now root user can always modify SELinux contexts and policies to get work done. For example root user can add port 90 to list of ports (`http_port_t`) that apache is allowed to listen to. But even

such changes can be prohibited by policy to not take effect until system reboot. Thus unwanted changes in policy would require system reboot which will get noticed and may even require manual intervention to enter passwords in case encrypted file-systems are in use, making system very secure.

A bigger advantage of SELinux is that even if apache is compromised and due to some vulnerability attacker is able to get shell access through apache. Attacker will not be able to use privilege escalation and get root privileges as attackers role will remain to be `httpd_t` and will not change to something else like `unconfined_t` which can be used by root users.

- Example 2 - Categories and Sensitivity
  Consider example of small organization with accountants and security guards as two different categories of people. If accountants and security guards are both provided login accounts on same system then it cannot be guaranteed that accountants cannot share information with security guards and vice-versa in DAC based systems. In DAC user owner can always make any owned resource world readable and world writable for everyone to read and write.

  With SELinux we can assign login roles to all accountants and security guards such that their files automatically get categorized. Then policy can be setup so that accountants cannot access files owned / created by security guards with category of security guards and vice-versa. This rule will get enforced even if someone uses `su -` and tries to convert current login to root login.

  In the same organization there can be security guards at different level of security clearance. That is there can be guards at lower level, than manager level and then some top level manager guards. We may not want information entered by top level manager guards to be accessible by security guards at lower level. At the same time we would want all the information accessible to lower level guard to be accessible by manager level guards and top level guards. This can again be achieved using SELinux by assigning proper **sensitivity** to each document and user role.

- SELinux modes
  Learning and using SELinux was fairly complex (and probably to some extent still is), and hence most distributions give ways of running SELinux which are comfortable for users and system administrators.

Few common ways of running SELinux are:

- Permissive mode
  In permissive mode SELinux allows all actions if they are acceptable as per DAC rules but logs or alerts in case an action is supposed to be denied as per configured policy. Thus all the commands and programs work as they would work without having SELinux with minor difference of logs / alerts being generated on SELinux policy violations.
- Targeted mode
  In Targeted mode only specific programs usually server processes like `apache`, `ssh`, etc. which accept network connections or serve users are run with SELinux security and policy constraints. All the other programs like GUI (X11 Server), gnome-terminal or text editors run without any SELinux context. Thus servers are secured with SELinux and normal programs, editors and tools are not affected by SELinux in any way.
- Disabled mode
  SELinux can also be completely disabled. To disable SELinux edit `/etc/sysconfig/selinux` file and change `SELINUX=enforcing` to `SELINUX=disabled`. SELinux can be disabled temporarily (provided configured SELinux policy permits so) by using `setenforce 0` command. It can again be enabled later on without requiring reboot using `setenforce 1`. At any time we can use `getenforce` command to check current SELinux user.

  As a rule of thumb if you ever get access denied messages when doing something as root, it is always worth it to check whether the error messages are due to SELinux. A quick (but risky) way of checking it is by disabling SELinux temporarily and try the action again. If action succeeds after disabling SELinux and again fails after enabling SELinux then we can be sure that SELinux is causing things to fail and fix contexts and policies to make things work.

- Conclusion
  This was just basic introduction to SELinux. It may not be beneficial for small or medium sized organizations to spend efforts in learning SELinux as it is fairly complex and documentation on SELinux is severely limited. But it can definitely provide great security to large organizations with many users working with different sensitivity of data

in different branches. It is also definitely useful for defense sectors like police, army etc.

To learn SELinux (not for mere mortals) one can use following links:

- http://www.fosteringlinux.com/category/articles/selinux/
- http://www.vantosh.com/publications/LK2010-SELinux-Tutorial.pdf
- http://www.securitytube.net/video/991

2.1.4 Auto-logout

If system is used remotely and it is desired that idle users get logged out automatically then we can create script `/etc/profile.d/autologout.sh` with following contents:

```
#!/bin/bash

TMOUT=900
readonly TMOUT
export TMOUT
```

Here value 900 is in seconds (equivalent to 15 minutes)

2.2 Keeping file safe

2.2.1 Versioning

- Subversion
  Sample SVN commands:

  - To create new SVN repository

  ```
  svnadmin create <project_name>
  ```

  - To checkout repository in current folder for working

  ```
  svn checkout file:///<full_path_to_project_folder>
  ```

  - To add files to repository

  ```
  svn add <files>
  ```

  - To commit changes done to files which are part of repository

```
svn commit -m "<message>"
```

  - To check status of files with respect to last update or commit

```
svn status
```

  - To see commit logs of particular file

```
svn log <filename>
```

  - To update current working copy with latest revision of files from repository

```
svn update
```

  - To get help on svn

```
svn help [<topic>]
```

To learn SVN properly in detail use http://maverick.inria.fr/~Xavier.Decoret/resources/svn/index.h

- git
  Sample git commands:

  - To set name and email address in global git config file

```
git config --global user.name "Saurabh Barjatiya"
git config --global user.email "saurabh@sbarjatiya.in"
```

  - To initialize a new folder as git repository

```
git init
```

  - To specify which modified files should be part of next index. This would add specified names to index which would be referred during next commit.

```
git add
```

  - To commit current changes to repository

```
git commit -m ""
```

- To check status of repository. It lists files which are part of index and would be updated on next commit, files which are modified since last commit or checkout but not added to index and files which are not being tracked. It also lists which branch we are working on and unresolved conflicts.

```
git status
```

- To list difference in file with respect to given commit or tag

```
git diff
```

- To lists all the commits done so far along with commit messages.

```
git log
```

- To lists all the branches available for the current project

```
git branch
```

- To creates a new branch with current state of repository

```
git branch <new branch name>
```

- To checkout repository files with commit name, branch name or tag

```
git checkout {<commit> | <branch> | <tag>}
```

- To tag current commit with name so that we can later use this for checkout

```
git tag
```

- To merge other branch with current branch. If merge results into conflict then we are notified about it. All conflicts have to be resolved before next commit.

```
git merge <branch>
```

To learn git properly and in-depth use http://book.git-scm.com/

2.2.2 Backup

- `rsync`
  Common rsync options are:
  - `-v` (verbose)
  - `-t` (preserve timestamps)
  - `-r` (recursive)
  - `-p` (preserve permissions)
  - `-a` (archive -rlptgoD)
  - `-z` (compress)
  - `--progress`
  - `--inplace`
  - `--delete`
  - `--partial`

  To understand various options of `rsync` refer to rsync man page (`man rsync`).

- `dd`
  Command dd options are:
  - `if=` (input file)
  - `of=` (output file)
  - `count=` (number of blocks to copy)
  - `bs=` (block size)
  - `skip=` (number of blocks to skip from beginning)

  To understand various options of `dd` refer to dd man page (`man dd`)

2.2.3 Raid

- Types of raid
  Link - http://en.wikipedia.org/wiki/RAID

- Software raid (`mdadm`)
  Basic mdadm commands:
  - Check status of current raid arrays

`more /proc/mdstat`

- Create a new raid raid 1 array with two members

`mdadm -C /dev/md3 -n2 -l1 /dev/xvdc1 /dev/xvdd1`

- Add given device to existing array

`mdadm -a /dev/md3 /dev/xxdd1`

- Stop given array (No reason to do this)

`mdadm -S /dev/md3`

- Store information of running auto-detected arrays in start-up file.

`mdadm --detail -sv > /etc/mdadm.conf`

- Auto detect raid arrays (Very useful when booting on raid system with Live CDs or rescue mode)

`mdadm --auto-detect`

There are myths that software RAID is very slow and unusable. I have been practically using software RAID on servers since more than two years now and have never faced performance and reliability problems. The only error possible is that sometimes GRUB is not installed on all disks which are part of RAID array and hence system may not boot when one of the disk fails. This can be easily resolved by booting into rescue mode and enabling bootable flag of all RAID partitions which are part of GRUB array. Then grub has to be re-installed on array using something like `grub-install /dev/md0`. Note that this happens when system with failed disk is rebooted. There is no effect on running system due to failed disk. Hence failure of disk will never lead to data loss or unannounced down time. Once we are aware that disk is failed we can announce downtime of few minutes and reboot the system to replace the failed disk.

- Hardware raid
  Whenever available can be preferred over software array. Note that some motherboards display RAID options even when there is no hardware RAID. In such systems same Linux style software raid is loaded on motherboard ROM and part of installed RAM is used for RAID. Such fake hardware arrays are very slow and do not provide flexibility and power of actual software array implemented in Linux. Hence care should be taken to ensure that actual hardware RAID controller is present when using hardware RAID.

2.2.4 Encryption

- Encrypting files (`gpg`, `kgpg`)

  - To generate new keys

```
gpg --gen-key
```

  - To encrypt a file using public key (Note: To keep files safe encrypt using your own public key)

```
gpg -r <name> -e <file_to_encrypt>
```

  - To decrypt file encrypted with our public key using our private key

```
gpg --output <file_name> -d <encrypted_file>
```

  - To edit key, for example to change pass-phrase

```
gpg --edit-key <name>
```

  This takes to a edit-key menu where one of the options (`passwd`) is for changing key pass-phrase. One can use `help` command to see list of all possible commands.

  - To encrypt using symmetric key (using password)

```
gpg -c <file_to_encrypt>
```

  - To decrypt file encrypted using symmetric key (password)

`gpg --output <file_name> -d <encrypted_file>`

  - To sign a file

`gpg -b -s <file_to_sign>`

  - To verify signature on a file

`gpg --verify <signature> <file that was signed>`

  - To list all keys

`gpg --list-keys`

  - To list all public keys

`gpg --list-public-keys`

  - To list all secret keys

`gpg --list-secret-keys`

Note that all the above operations can be done very easily using `kgpg` GUI too. The commands are listed so that operations can be scripted / automated.

- Encrypted file systems (`encfs`)
  `encfs` can be installed using `yum installed fuse-encfs` provide rpmforge, epel etc. repositories are properly configured. Then we can use

  `encfs <full_path_of_encrypted_dir> <full_path_of_work_dir>`

  When the command is used for first time we can configure password and choose how to encrypt. Default security options for encryption are good enough. All future mounts of encrypted dir would require same password. There is a `.encfs<n>` file in each encrypted dir that is used to store information on how to decrypt.

  To change password used for storing `.encfs<n>` file we can use

  `encfsctl <raw_directory>`

  Note that by default only root user is allowed to access contents of work dir when a encrypted file system is mounted. To allow non-root user to access work dir we must use `--public` option while mounting encrypted raw dir to work dir.

2.3 Monitoring what is going on

2.3.1 Resource usage

- RAM (`free`)
- Hard-disk (`df -h`, `quota`)
- CPU (`top`, `htop`, `uptime`)
- Processes (`ps`)
- Open files (`lsof`)
- Network
  - Open sockets (`netstat`)
  - Running TCP connections (`tcptrack`)
  - Protocol wise or filtered statistics (`iptraf`)
- Users logged in (`w`, `who`)
- Hardware connected (`lspci`, `lsusb`)
- Services (`chkconfig`, `service`)

2.3.2 Remote monitoring

- SNMP
  Very basic SNMP server configuration can be learned from http://www.cyberciti.biz/nixcraft/linux
- MRTG
  Basic MRTG configuration can be learned from http://www.cyberciti.biz/nixcraft/linux/docs/unic
  Note that MRTG is not only for monitoring end hosts it can be used for monitoring network devices like firewalls, switch, IPS, routers etc. most of which support SNMP based monitoring.
- Nagios
  Nagios quick-start configuration can be learned from http://nagios.sourceforge.net/docs/3_0/quic
  fedora.html After doing quick install one can either do manual configuration of servers and hosts or use GUI front-ends like NagiosQL (http://exchange.nagios.org/directory/Addons/Configuration/NagiosQL/details) or NConf (http://exchange.nagios.org/directory/Addons/Configuration/NConf/details) to configure nagios installation. If users are interested in configuring service monitoring using config files instead of using GUI then

they can refer to http://nagios.sourceforge.net/docs/3_0/monitoring-publicservices.html to learn how to monitor public services using nagios.

2.4 Logging actions

2.4.1 Auditing

A detailed report on auditing is hosted at http://www.sbarjatiya.in/website/courses/2010/monsoon/pro
Interested users can download the report and use it to learn how to use Linux auditing.

2.4.2 Command history

We can create file named `/etc/profile.d/history.sh` with following contents:

```
#!/bin/sh

HISTTIMEFORMAT="%y %m %d %T "
HISTSIZE=100000
HISTFILESIZE=100000
export HISTTIMEFORMAT HISTSIZE HISTFILESIZE
```

to ensure that all commands entered by all users get logged with date and time. This also ensure that last 100000 (practically all) commands get logged and not just last 1000. If root access is not available then the same configuration can be done in `.bash_rc` file of user account so that at least history of given user commands is stored with date and time information

2.4.3 Capturing packets

- `tcpdump`
  `tcpdump` allows one to capture packets received on a interface. It can print analysis of packet on command line or can save packet information in .pcap file for later analysis. Since tcpdump is command line based, its use can be automated using scripts. For more complex packet capture requirements one can explore use of `libpcap` directly. tcpdump also uses libpcap for packet capture and filter operations.

  Useful command line options are:

Option	Description
-c	Stop capturing after count number of packets have been captured
-C	Do not store more than specified MB in single capture file
-D	List interfaces along with description
-i	Interface to capture on
-n	Do not convert IPs to host-names
-nn	Do not convert port numbers to service names
-p	**Do not** put interface in promiscuous mode
-r	Read packets from given file and not from live interface
-s	Capture only first specified number of bytes of each packet
-v	Verbose output
-q	Quieter output
-w	Write output to file
-A	Print information contained in packet in ASCII format
-x	Print information contained in packet in hexadecimal format

tcpdump supports filter formats which get converted to kernel level packet capture filters and thus are more efficient then user level packet capture filters. Various expressions that can be given to tcpdump as filter argument and their meaning are described in below table:

Expression	Meaning
host <ip_address>	Only capture packets to/from given IP address
src host <ip_address>	Only capture packets where source IP matches given IP address
dst host <ip_address>	Only capture packets where destination IP matches given IP address.
port <number>	Only capture packets with given source or destination port number
<protocol>	Only capture packets of specified protocol
and, or, not	Multiple expressions can be combined with and, or, not

Refer to `man tcpdump` for more detailed information on tcpdump command line options and filter expressions.

- `wireshark`
  Wireshark provides GUI interface to capture and analyze packets.

Wireshark is more powerful than tcpdump as it understands various protocols and their header formats in great detail. It can also be used to rebuild captured streams. Wireshark also supports display filters along with capture filters, where display filters are more fine-grained than capture filters. Wireshark is available for all platforms and works equally well in all of them.

2.4.4 Remote logging

- `rsyslog`
  One of the very important tools for reliable information after system is compromised is logs. But if system is compromised we can't depend on log files as they could have been altered or deleted. Hence it is important to store log of all important servers on some different (probably central) log server which is extremely hardened and very tightly controlled. This log server can then be used to analyze causes and sources of break-in.

  To send logs to remote machine or to receive logs from remote machine we can use rsyslog. rsyslog configuration can be learned from http://rsyslog.com/doc/rsyslog_conf.html One can also try to use from many samples provided at http://wiki.rsyslog.com/index.php/Configuration_Samples after browsing through basic rsyslog configuration documentation.

2.5 Analyzing logs

2.5.1 Periodic log reports (`logwatch`)

Script `/etc/crond.daily/0logwatch` runs daily due to cron and sends email report of log analysis. This report contains important information like

- Access denied or File not found error messages sent by web server
- Hacking attempts detected on web server
- Disk space analysis
- Commands run as sudo
- Mail server statistics if mail service is configured

Since these reports are sent by email proper configuration of local SMTP server (like `sendmail`) is required so that this emails reach administrator without any problem. If email system or aliases are not configured properly

all these reports go to root@localhost and are rarely read. Both syslog and logwatch are highly configurable systems and can be used to log and report interested events periodically. Hence proper use of logwatch can help in detecting unwanted changes without relying on administrators to manually go through log files regularly to detect anomalies.

2.6 Detecting unwanted changes

It is important to detect unwanted changes to file-system. Auditd is one of the tools which can help in monitoring changes. But sometimes we may want changes to be compared with previous system state to get idea of what got modified since last check. File based intrusion detection systems provide this functionality. With file based IDS we can keep information about system state at any given time and then compare it with system state at a later stage. Interesting differences (where we specify what is interesting) are shown. Such tools help in detecting unauthorized modifications of systems files. This also helps in detection of installation of back-doors and root-kits.

2.6.1 File based Intrusion Detection Systems (FIDS)

Two very famous file based intrusion detection system are discussed below. Basic difference between tripwire and AIDE is that tripwire supports cryptographic security by providing options for signing database generated while examining system state. Hence it is not possible for someone to update both system and tripwire database without knowing tripwire key passwords. But in case of AIDE an intelligent attacker who detects presence of such systems can update AIDE database to avoid detection. Off-course such attack vectors can be reduced by keeping AIDE database backups on other machines or by signing AIDE database and verifying the signature later. But managing systems where databases are stored remotely or where signatures have to generated and verified manually, is fairly complex.

- Tripwire
  Tripwire installation and configuration can be learned from following links:

  - http://www.linuxjournal.com/article/8758?page=0,2
  - http://www.akadia.com/services/tripwire.html
  - http://centos.org/docs/2/rhl-rg-en-7.2/ch-tripwire.html

  Considerable information on how to use tripwire is present in following man pages:

  - `man twintro`
  - `man tripwire`
  - `man twconfig`
  - `man twpolicy`
  - `man twadmin`

- AIDE
  AIDE is very simple and easy to use (at least in comparison to tripwire). To learn AIDE one can start with below configuration file:

```
@@define AIDEDIR /home/saurabh/Desktop/aide
database=file://@@{AIDEDIR}/database/current_database.db.gz
database_out=file://@@{AIDEDIR}/database/new_database.db.gz
verbose=20
report_url=file://@@{AIDEDIR}/log/aide_report.txt
report_url=stdout
gzip_dbout=yes
warn_dead_symlinks=yes
config_version=1.0

/home/saurabh/Desktop/aide/test p+u+g+S
```

  Replace =/home/saurabh/Dekstop= in above configuration appropriately.

  Once configuration file is setup as mentioned above one can learn AIDE by performing operations in test folder and checking for reports using AIDE.

  Useful AIDE commands are:

  - Checking configuration

```
aide -c <configuration file> --config-check
```

  - Initializing database

```
aide -c <configuration_file> -i
```

  - Check system against current database

```
aide -c <configuration_file> -C
```

- Check system and also update database

```
aide -c <configuration_file> -u
```

We can run AIDE periodically using cron. Sample script which can be used for such checks is

```
#!/bin/bash
/usr/sbin/aide --update -V 20 | /bin/mail \
      -s "Weekly Aide Data" barjatiya.saurabh@gmail.com
cp <new_database> <current_database>
```

Note that AIDE does not provides cryptographic security when used as mentioned above. When using AIDE as mentioned above onus of protecting database from attacker is on administrator.

3 Security from network

3.1 Firewalls

3.1.1 `iptables`

`iptables` is very powerful and flexible host firewall available on all major Linux distributions by default. A very brief introduction to iptables is hosted at http://www.sbarjatiya.in/website/tutorials/iptables/iptables.pdf A detailed project report on iptables is hosted at http://www.sbarjatiya.in/website/courses/2011/spring/pro iptables.pdf

3.1.2 `ebtables`

Although iptables is very powerful and flexible, it cannot be used for firewalling purposes between VMs located on same base host when VMs communicate with each other using software bridges. In such cases `ebtables` comes to rescue. We can apply `ebtables` rules on bridge interfaces even when some of them do not have IP addresses assigned to them, making `iptables` unusable for such interfaces. Syntax, configuration files and man page of `ebtables` are very similar to `iptables` making it very easy to learn, once someone is comfortable with iptables.

3.1.3 `arptables`

If ARP protocol filtering is required then `arptables` firewall has more features and options then what are provided by `iptables` for ARP can be used.

3.2 Secure Shell

One among major advantages of using Linux Operating System is having SSH server installed and available on most distributions by default. If used properly SSH helps in securing remote access to machines (both command line and GUI) and also allows secure way of transferring or synchronizing files between two machines. SSH can also help in tunneling with support for Local, Remote and Dynamic port forwarding. Thus it is important to understand and use SSH properly so that it can aid as great security tool.

3.2.1 Scripting password based ssh login (`ssh-pass`, `expect`)

Many times users face problems in scripting when scripts require to run ssh / scp commands which would prompt for passwords. To supply passwords through scripts one can use `ssh-pass` hosted at http://linux.softpedia.com/get/Security/Sshpass-8693.shtml If more complex scripting requirements are present then one can learn `expect` scripting language.

Sample expect script that establishes SSH connection and runs `ls -l` command is:

```
#!/usr/bin/expect -f
spawn ssh saurabh.barjatiya@mirage.iiit.ac.in
expect "password:"
send "<password>\r"
expect "saurabh.barjatiya@mirage"
send "ls -a\r"
set timeout 1
expect "Not there"
send_user "$expect_out(buffer)"
close

#send "exit\r"
#expect "Not there"
#send_user "$expect_out(buffer)"
```

Another sample expect script that establishes a SSH connection and gives control to user after login:

```
#!/usr/bin/expect -f
spawn ssh saurabh.barjatiya@mirage.iiit.ac.in
expect "password:"
send "<password>\r"
expect "saurabh.barjatiya@mirage"
interact
```

Lastly sample script that allows copying of large VM images from one system to another using expect

```
spawn rsync -vazS --progress --partial \
    root@10.3.3.48:/mnt/data1/saurabh_do_not_delete/ \
    /mnt/data1/saurabh_do_not_delete/
set timeout -1
sleep 3
expect {
                yes/no {
                send "yes\r"
                exp_continue
        }
        password: {
                send "<password>\r"
                exp_continue
        }
        denied {
                send "<password>\r"
                exp_continue
        }
        eof {
                exit 0
        }
}
wait
exit 0
```

3.2.2 Key based ssh login (`ssh-keygen`, `~/.ssh/authorized_keys`)

If expect and ssh-pass based methods are not acceptable as they require storing of plain-text password in script source codes or prompting for them as command line argument then we can establish trust / key based authentication between two systems. For this one system generate keys using

```
ssh-keygen
```

Then copy file `~/.ssh/id_rsa.pub` to other machine at location `~/.ssh/authorized_keys`. It is important that permissions on folder `~/.ssh` are 700 and that `authorized_keys` file has 600 permissions. After setting the keys properly if we SSH from one system another then we don't get prompted for password and login is allowed based on keys.

3.2.3 Keys with pass-phrase (`ssh-agent`, `add-key`)

Key based login is also not acceptable sometimes as if system is left unattended or keys get compromised then attacker will get full access to system which trusts compromised keys. Hence ssh keys are often protected with passwords (called pass-phrases in this case). To assign pass-phrase to existing keys or to change pass-phrase of existing keys we can use following command:

```
ssh-keygen -p
```

Now every time the given key is used for authentication then user will get prompted for password. This can be annoying if we are going to use many consecutive ssh commands which will depend on keys as we will have to supply pass-phrase once for each command. To solve this problem we can run ssh-agent using command:

```
exec $(which ssh-agent) $SHELL
```

and on the obtained shell (which replaced parent shell transparently) we can use command:

```
add-key
```

and supply password (pass-phrase) only once. This causes key to get stored with ssh-agent and we do not need to supply pass-phrase for this particular running instance of shell. As soon as we exit from shell ssh-agent also exits automatically making system very secure.

3.2.4 Blocking brute-force attempts (`denyhosts`)

Anyone who has administered Linux server with public IP and SSH enabled for more than a day can check logs to see how many people attack a new server using brute-force to login as root user in just one day. The number is

really surprising. Given that SSH as root will compromise system completely allowing attackers to continue trying different passwords on servers is not a good idea. Thus it is important to stop same person to try more than given number of password within an interval, especially for user accounts like root. Denyhosts script allows achieving this goal in very easy and efficient manner.

One should be able to install `denyhosts` on all major distributions using package managers like `yum`, `apt-get`, etc. Location of `denyhosts.cfg` file can be found using `locate` command (preceded by `updatedb`, if necessary). Lines that I personally modify in default `denyhosts.cfg` file are:

```
PURGE_DENY = 12w
PURGE_THRESHOLD = 2
BLOCK_SERVICE = ALL
DENY_THRESHOLD_ROOT = 10
ADMIN_EMAIL = barjatiya.saurabh@gmail.com
SYSLOG_REPORT = YES
AGE_RESET_VALID = 2d
AGE_RESET_ROOT = 2d
RESET_ON_SUCCESS = YES
DAEMON_SLEEP = 120S
(Uncomment) 'SYNC_SERVER = ...' line
SYNC_DOWNLOAD_THRESHOLD = 10
```

It is also recommended to install `denyhosts` as service and enable it on start-up so that it automatically runs on reboot. To prevent important trusted IPs from getting blocked they can be white-listed by creating file named `allowed-hosts` in denyhosts `data` directory. We can add one IP per line in this file and all the mentioned IPs will not get black-listed irrespective of how many wrong password attempts are made from these IPs.

White-listing configuration must be done before denyhosts is used. If we try to white-list banned IP to remove ban, then it may not work without tinkering with other denyhosts data files.

3.2.5 login logs

Linux also supports various commands which provide important information about various successful and unsuccessful login attempts made on a system using ssh. Few of those commands are:

- **last :** `last` command can be used to check list of current month's successful login

- **lastb :** `lastb` can be used to check list of current month's bad login attempts
- **lastlog :** `lastlog` command can be used to check last successful login of each user

3.2.6 Disabling command execution over SSH

- Restricted SSH (`rssh`, `scponly`)
  Sometimes on shared systems users do not need full ssh or command line access and only need to transfer files (`scp`, `rsync`, `sftp`, etc.) to / from shared host. In such cases giving full ssh permissions is not safe. To restrict users so that they can only perform `scp`, `rsync` etc. and nothing extra one can install `rssh` or `scponly`. Installation and configuration of these two tools is very easy and hence is not described here.
- Restricting sub-system to sftp with chroot
  If one has SSH version 4.9+ (can be installed using source on Cent-OS) then we can use chroot and sftp force options provided by latest SSH server without requiring to install or setup `rssh` or `scponly`. Configuration lines that would force users belonging to group `sftponly` to use sftp are

  ```
  Match Group sftponly
     ChrootDirectory %h
     ForceCommand internal-sftp
     AllowTcpForwarding no
     X11Forwarding no
  ```

  Apart from forcing users to use `sftp`, the above lines would also chroot them to their home directory. Hence users wont be able to access anything outside their home-directory. The last two lines disable TCP and X11 forwarding so that only thing users can do with SSH is file transfer.

3.2.7 Preventing misuse of SSH

SSH supports port forwards in form of Local, Remote and Dynamic port forwards. SSH man pages and documentation on Internet can be read to

understand what each of these forwards provide and how to use them. Problem for administrators usually is that these features provide security loop holes as users can establish connections to/from host to which they SSH. Hence if SSH access to public server is provided then using Dynamic port forwarding one can turn server to a SOCKS proxy. To ensure that such security problems are not possible it is best to disable GUI login and TCP port forwarding for all non-root users.

Configuration lines that allow blocking of GUI login and port forwarding are:

```
AllowTcpForwarding no
X11Forwarding no
```

These can be preceded with filter `Match User !root` (Caution: experimental) so that these rules apply to all except root user.

3.3 Virtual Private Networks

3.3.1 OpenVPN

Many times SSH login to public servers are provided so that users can connect to a enterprise network from outside. But VPN is definitely a better and safer alternative in comparison to SSH for remote access. VPN is complex and hence cannot be discussed in this session. Interested administrators are encouraged to learn configuration of OpenVPN using How-To located at http://openvpn.net/index.php/open-source/documentation/howto.html

I have setup OpenVPN for IIIT Hyderabad and it has been in use by more than 1000 users (10-15 concurrent users) since many years and we do not face any problems in using OpenVPN. OpenVPN is ideal when users could connect from various operating systems like Windows, Mac OS X, Solaris, Linux etc. to OpenVPN server to use VPN services. OpenVPN works well for both 64-bit and 32-bit OS variants without any problem.

3.4 Port scanning

3.4.1 `nmap`

It is always good to verify after proper `iptables` configuration has been done that only desired services are accessible from outside users. An easy way to accomplish this is to port scan protected server from prospective client machines to check which ports are open. One very versatile port scanning

tool available for Linux (also works equally well with Windows) is `nmap`. `nmap` is well documented and we can use:

```
nmap
```

command without arguments to see various options supported by nmap and try each of them.

4 Securing Internet access within organization

Internet access is provided to employees in almost all organizations as access to email, on-line search engines, social networking websites, Wikipedia, etc. is required or at least improves productivity of employees. But if such access is provided without proper safe guards and without educating users about potential threats from Internet, then such access is very unsafe. Hence very basic Internet security related points are discussed below. Although these points do not come under 'open source security tools' which is main title or agenda for the lecture. But ignorance or mistakes with respect to mentioned points will render use of all sophisticated tools / techniques discussed so far useless, and expose users and in turn organization to variety of attacks.

4.1 Poor coding based attacks

Developers who code websites (especially active / dynamic websites) for company should be educated about various types of attacks that are tried on any dynamic website from malicious users. At least developers should ensure that the websites in use are safe from:

- SQL Injection attacks
- XSS attacks
- Brute force attacks

4.2 Firefox plugins for browsing security and privacy

Advertisements and social networking website APIs (facebook, twitter, etc.) allow tracking of user movement from one website to another. If privacy is a concern (which is usually the case) then it should not be possible for advertisement websites or social networking websites to know about all on-line activity of particular user. Hence use of following security plugins with Firefox browser is recommended for safe Internet use:

- Ghostery
- No Script
- Web of Trust (WOT)
- Ad-block plus
- Better privacy

4.3 Sensitive posting in social networking websites

It is important to ensure that users do not post anything sensitive (esp. photos and videos) on social networking and video sharing websites. Users should be explained that things once uploaded, **cannot be taken back / deleted**. Files which are uploaded will remain available to one or the other person in one form or another and it cannot be guaranteed that same file wont surface again somewhere else on Internet.

4.4 Saving passwords in browser

Users very often save passwords in browsers for convenience. It should be ensured that all users who save passwords also setup master password to ensure safety of saved passwords. It should be noted that master passwords can only be set in browsers like Opera, Firefox etc. Many famous browsers like Internet Explorer (at least as per my knowledge at time of writing) do not support configuration of master password. Hence use of Internet Explorer to save passwords is extremely bad idea. If password is stored by Internet explorer on system without encryption (in plain-text) then it can be read by other applications without any problem. Note that for login systems to work browsers cannot store hash of passwords, they are required to store complete password for auto password filling to work.

4.5 Saving passwords in chat client

Again as per my knowledge at time of writing there is no chat client including `pidgin`, Gtalk, Yahoo messenger to name a few which supports configuration of master password. This makes sense as usually users need to provide only one password in chat clients to login, so having another single password to protect this password does not makes much sense. But storing passwords in such chat clients is very unsafe, same as storing passwords in browsers without use of master password.

4.6 HTTP vs HTTPS

Often difference between HTTP and HTTPS and how these protocols are different in poorly understood. Very rarely users are aware of role of certificates, certificate authorities, etc. in making HTTPS secure. Hence these concepts should be explained to all users so that their Internet browsing, especially authentication and banking are secure.

Topics that can be discussed under this heading are:

- ARP spoofing
- Port mirroring
- Untrusted ISP
- CA certificates
- squid HTTPS port and use of stunnel

4.7 IPv6

IPv6 support is now present **and enabled** by default on all operating systems. Thus failure to secure system on IPv6 addresses can lead to severe security problems. General awareness on IPv6 severely lacks understanding of even basics of IPv6 like addressing, routing, etc. Hence IPv6 provides a significant attack surface area in absence of tools and mechanisms that protect IPv6 based network attacks. Thus it is important to setup proper rules to prevent IPv6 based connections / attacks unless IPv6 concepts are properly understood and implemented by administrators and users.

For example IPv6 supports tunneling using 2002::/16 on IPv6 side and protocol 41 on IPv4 side. If firewall rules are not setup to prevent IPv4 to IPv6 tunnels and vice-versa then such tunnels can be used to establish connections to any remote machine without getting affected by devices / rules which do not take IPv6 into account. Even if ISP does not provides IPv6 services and addresses, free tunnel broker websites around the world allow users to connect IPv6 networks using tunnels without requiring any formal verified registration or paying any fee.

Most Linux services like SSH, HTTP etc. are configured such that they listen on both IPv4 and IPv6 addresses. Hence configuring firewalls like `iptables` without doing corresponding configuration in `ip6tables` would leave serious security blind-spots.

4.8 Anonymous browsing

Anonymous browsing is both security tool and potential security problem. With anonymous browsing one can open websites without being tracked. Note that anonymous browsing to avoid tracking would be useless without plugins like Ghostery, BetterPrivacy etc. as advertisements, cookies, etc. would allow tracking of user without requiring him / her to make requests from same IP address.

Anonymous browsing is security problem as all the websites blocked due to company policy can be browsed via anonymous browsing websites. Blocking anonymous websites themselves is not easy as they keep changing IP addresses and keep coming up with ways to avoid blocking by all famous security software. Also failing to block even one such website can lead to access to all websites.

It should be noted that skilled developers / programmers can program such websites on their own in very little time and host them on cloud, personal servers outside organization boundary. Later they can use their own websites for visiting prohibited websites. This cannot be avoided as it is impossible to know purpose, owner etc. of each and every website visited by user.

Famous anonymous websites or tools that help in anonymous browsing are:

- http://www.anonymouse.org
- Freedom
- Tor
- JAP

4.9 Anti-virus

Availability of anti-viruses for Linux operating system have caused severe confusion that Linux operating systems are equally vulnerable to viruses as other operating systems like Windows, which is not necessarily true. Linux is very secure in its default configuration of firewalls, services, etc. at time of install. It is easy to have complete security breach if user accounts setup are not secure (weak passwords, etc.) as one can use any user account to SSH / sftp to server and run commands or transfer files unless such access is specifically blocked for particular user. Also once shell access is present, almost all operations including installation from source files can be performed by

normal user without root privileges. Hence installing of back-doors, spying software etc. by attacker on systems that are not configured properly is very easy, as number of powerful tools like shell, compilers (gcc), languages(shell script, python, perl, php) etc. are available on most systems.

But all this requires severe mistakes from administrators side. If administrators keep strong passwords and user accounts (specifically SSH) in control then system can be considered to very safe. Specially installation of ad-ware, spyware due to clicking some bad link on browser / email client etc. is not possible. All downloaded files do not have execute permissions by default and users have to manually assign execute permissions to execute downloaded files. Hence possibility of having virus attacks accidentally is very rare.

One of the free anti-viruses available for Linux is Clamav. Anti-virus software are provided in Linux OS for scanning email attachments for viruses or ensuring virus files are not uploaded on common file shares, as these services can be used from other more vulnerable operating systems which require considerable care to protect them against viruses.