Global Information Assurance Certification Paper

Introduction

In the ever-changing world of global data communications, inexpensive Internet connections, and fast-paced software development, security is becoming more and more of an issue. Security is now a basic requirement because global computing, enabled by the Internet, is inherently insecure. As your data goes from point A to point B on the Internet, for example, it may pass through several other points along the way, giving other users the opportunity to intercept, and even, alter it. Even other users on your system may maliciously transform your data into something you did not intend. Unauthorized access to your system may be obtained by intruders, also known as "crackers", who then use advanced knowledge to impersonate you, steal information from you, or even deny you access to your own resources.

These security issues arise naturally when a system is connected to a public network, where network access is available to everyone. It becomes especially important when it is connected to the Internet. To protect against possible threats of these connections, security measures are provided at different levels. Network-level security measures filter unwanted traffic on the network. Host-level security measures provide protection mechanisms for the host itself. The protection for the host depends on the operating system.

One important operating system that has gained widespread these days is Linux. For this reason, Protecting Linux has been the concern of many Internet security professionals, whose efforts resulted in "Securing Linux Step-By-Step" guidebook by the SANS Institute. The SANS (System Administration, Networking, and Security).

The book is very practical and useful in increasing the security of a Linux host. It is organized as a guidebook that tells its reader what to do with each service

without going into theoretical backgrounds. It gives pointers to other material for more information. Its organization is very well suited to direct application of the steps. Still, a reader of this guide needs to have a good knowledge how to deal with Linux, which is not a trivial task.

This is where my contribution comes in. I have programmed these steps and made it as a middle layer between the user and Linux. I have done the programming incrementally, and then finally i made the integration of all the resulting code. I believe my program will facilitate and simplify a great deal of the securing process for the user. It might minimize human error also.

I will address my accomplished work. For that purpose, i have tried to format the chapter to be well structured in a unified manner. Each network service presents its own security holes, i mention them for each network service and how the guidebook deals with each, then i list a small program in pseudo-code that facilitates this task, and finally, i show how i integrate it all.

I will present each network service in a section, with three subsections: security issue, solution, and scripting. For each security issue i give some descriptions followed by the solution as provide by the guidebook. The solution is intended to increase the security of the service. After a solution is provided, i list my contribution by the title "scripting". I give in these sections the high level steps necessary to write the script, so it is more readable. Appendix A list all source code of my work written in Korn shell language.

1.1 Securing inetd Services

The Internet daemon (**inetd)** starts at boot time from an initialization file. When inetd starts, it reads its configuration from the `/etc/inetd.conf` file. This file contains the names of the services that inetd listens for and starts. You can add or delete services by making changes to the `inetd` to the `inetd.conf` file.

1.1.1 Disable Internet Daemon Services

1.1.1.1 Security Issue

The Internet daemon "`inetd`" controls access to network services that only start up when needed, such as telnet and FTP. A typical workstation has no need to provide the services in the Internet daemon configuration file, `/etc/inetd.conf`.

In case of a server, we check what services that are not needed then they should be disabled.

1.1.1.2 Solution

To disable services in `/etc/inetd.conf`, edit the file and put the comment character "`#`" in at the begging of each line the corresponding to these service, Save the edited file and restart the "inetd" process.

1.1.1.3 Scripting

Below is a high level algorithm to implement the solution. For detailed Korn shell script, please **see Appendix A.3.1** & **A.3.2**

```
edit /etc/inetd.conf
for each service that is not commented(i.e. do not start with
"#" )
     ask the user if he wants to disable
     if yes
         comment this service
save the changes
```

1.1.2 Turn off inetd

1.1.2.1 Security Issue

A typical workstation has no need to provide the services in the Internet daemon configuration file, `/etc/inetd.conf`. Therefore, we can just turn off inetd completely.

1.1.2.2 Solution

If no services are allowed in `/etc/inetd.conf`, we can just turn it off by stopping the service and removing it from the SYSV init startup sequence.

1.1.2.3 Scripting

```
if no services are allowed in /etc/inetd.conf then
 stop inetd daemon and remove it from startup
```

1.1.3 TCP Wrapper For Enabled inetd Services

The TCP wrapper daemon, `/usr/sbin/tcpd`, is a small program that is invoked by inetd instead of the normal daemon program. It checks the source address of the request against its access control lists in `/etc/hosts.allow` and `/etc/hosts.deny`. If the lists say that the requester is allowed to connect, the wrapper daemon passes the connection to the appropriate service and everything proceeds as usual. If the lists say to deny access to the service, the connection is dropped. Either way, the wrapper daemon puts an entry in the system logs to document the event.

1.1.3.1 Security Issue

The TCP wrapper was adopted as standard access control mechanism very early on in Linux. All modern Linux distributions are shipped with inetd services wrapped by `/usr/sbin/tcpd`, however, by default, `/etc/hosts.allow` and `/etc/hosts.deny` are empty, so access is allowed from any host by default. Access control routines check `/etc/hosts.allow` for a match first. If found access is allowed. Then it check `/etc/hosts.deny` for a match. If found, access denied. If no match is found in either file, access is allowed.

1.1.3.2 Solution

1. Best practice is to deny all access by default, then selectively add hosts that are allowed access to services. In `/etc/hosts.deny` specify "`ALL: ALL`". The wildcards ALL stand for all services and all hosts.

2. Give access only to specific hosts for specific services. The basic format is: `daemon: host` daemon is the name of the daemon program defined in `/etc/inetd.conf`, host can be a symbolic hostname, IP address or network/netmask.

1.1.3.3 Scripting

Below is a high level algorithm to implement the solution. For detailed Korn shell script, please **see Appendix A.3.3 & A.3.4**

```
for deny all
     edit /etc/hosts.deny
     insert ALL: ALL
     insert ypserv: ALL
     save the changes
for allow only specific hosts
     edit /etc/hosts.allow
     for each service that is enabled by inetd
          ask the user if he wants to allow all hosts or
          specific hosts
          if he wants to allow all hosts then
               insert <service-name>: ALL

          otherwise,
               insert <service-name>: host
save the changes
```

1.2 Domain Name System and BIND

Machines that are connected in a network must be uniquely identified. This is done by assigning them a standard and unique address; the IP address. When a host needs to communicate with another host, the IP address of the destination host must be provided. This is not a very convenient job for humans, since they are dealing with numbers. People find it easier to memorize names, hence names are assigned to hosts and some program does the necessary mapping between the host names and their IP numbers, thus relieving users from this task.

In the past, networks were not as complex and large as they are now, therefore, the mechanism that was used was based on a centralized database file that contained all the host names and their IP numbers. However, as the Internet got larger, this file got very big and unmanageable. As a result, The Internet Community has figured out another way to solve the problem. This solution was the DNS system. Without going into further details, the principle that underlies the operation of DNS is simple; distributing the data among different DNS servers, in a structured manner, such that each server is responsible for part of the whole name space.

This way, the pieces of the data are smaller and manageable. BIND stands for Berkeley Internet Name Domain. It is the implementation of DNS under UNIX.

1.2.1 Security Issues with Zone Transfers

A zone can be considered as a piece of the name space for which there is one master and one ore more slave name servers. A master name server holds locally the information about the zone it serves. A "slave" server gets a copy of the zones from its master server, for fault tolerance reasons.

The operation by which the slave gets the copy is termed "zone transfer". If not restricted to slave servers only, zone transfers can be a security risk, as they potentially give outsiders a complete list of all of an organization's computers connected to the internal network. All the hostnames and their IP numbers will be exposed. Many sites choose to allow UDP DNS packets through their routers and firewalls, but explicitly block DNS zone transfers originating at external sites. This design is a compromise between safety and usability: it allows outsiders to determine the IP address of each internal computer, but only if the computer's name is already known.

1.2.1.1 Solution

There is a firewall-based solution, and a host-based solution. Since in my project i focus on host security, i will adopt the host-based solution. This is enabled by an option that can be set in BIND. Using this option, the system administrator can tell BIND to restrict zone transfers to specified hosts only, normally, the secondary DNS servers. To be more specific, under "named.conf", the zones for which "named" is told to be master, add the line *"allow-transfer {IP number list of slave servers};"*, and for those zones that it is told to be slave, add *"allowtransfer {none;}"* Here is an example:

```
zone "notransfer.com" {
     type master;
     file "db.notransfer.com";
     allow-transfer { 192.168.1.1; 192.168.1.2;};
};
```

1.2.1.2 Scripting

Below is a high level algorithm to implement the solution. For detailed Korn shell script, please **see Appendix A.5.4**

```
For each domain of type master in named.conf
    check if "allow-transfer" option exists; if not  ask the
user to supply the IP's of slaves, then    insert "allow-
transfer {IP's of slaves} in the  domain"
For each domain of type slave in named.conf
    check if "allow-transfer{None;}" exists; if not  insert
the line "allow-transfer{None;}" in the   domain
End of script
```

1.2.2 Security Issues with Root Access

With BIND version 8.2, named has the option of running as a user other than the superuser. This can limit the damage done if some other vulnerability is discovered that allows the attacker to get a shell. However, if someone can get in as a regular user, there is a good chance that some local exploit will give him root access.

1.2.2.1 Solution

The DNS daemon, *named*, can be made to run in its own chroot directory tree. That way, even if an attacker gets a shell, the amount of damage he can do is limited to the chroot directory. He can destroy or tamper with the name server, but that is all. Setting up a proper chroot jail must be done with care. Not only do the zone files and other configuration files need to be in the chroot directory tree, but libraries, devices, and executables necessary for the operation of *named* have to be there, too.

1.2.2.2 Scripting

Below is a high level algorithm to implement the solution. For detailed Korn shell script, please **see Appendix A.5.5**

```
create the new dns user and group
prepare the chroot directory
```

```
copy configuration files and programs
copy shared libraries
set syslogd to listen to named logging
edit the named init script to start in chroot jail
```

1.3 Securing Electronic Mail

An electronic mail message begins its life as a file on your computer's disk, created by a Mail User Agent (MUA). After you compose the letter, the MUA gives it to a Mail Transport Agent (MTA) like “sendmail”. The message traverses one or more hosts (MTAs) in different possibly networks, and is given to a final delivery agent, which appends it to the recipient's mailbox, another disk file. Each of these terms is explained in detail later in this section.

1.3.1 Turn of sendmail daemon mode

1.3.1.1 Security Issue

In Workstation does not need to run a Mail Transfer Agent (MTA), like sendmail, in daemon mode. Most Internet service providers, university, government offices, and commercial organizations provide mail relay severs.

1.3.1.2 Solution

Edit `/etc/sysconfig/sendmail` and insert this two line

```
DAEMON=no
QUEUE=15m
```

After saving the file, restart the sendmail daemon.

1.3.1.3 Scripting

Below is a high level algorithm to implement the solution. For detailed Korn shell script, please **see Appendix A.6.1**

```
if user wants to turn off sendmail
    edit file /etc/sysconfig/sendmail and insert this two
    line
        DAEMON=no
```

```
        QUEUE=15m
save the changes
restart the sendmail daemon
```

1.3.2 Define SMTP Server

SMTP is Internet's standard host-to-host mail transport protocol and traditionally operates over TCP, port 25. SMTP is protocol used in sending and receiving e-mail.

1.3.2.1 Security Issue

There are many choices for Mail User Agents that can be selected by Linux users. The character-mode MUAs, e.g. mail, elm, mutt and pine, all call sendmail directly to deliver mail. Note that this is not the sendmail daemon used to receive mail that we disabled in the previous section. In addition, pine, nmh, exmh, Netscape mail and Email from KDE can be set to use a remote SMTP server.

1.3.2.2 Solution

There are two ways to do this, using the m4 macros or editing /etc/sendmail.cf directly. You direct sendmail to relay all out-bound mail to the SMTP server with the SMART_HOST feature. Edit `/etc/sendmail.cf`. look for a line start with DS, DR, DH and DM. Edit the lines to read as:

`DSsmtp.example.org` where smtp.example.org is the fully qualified domain name of the SMTP server.

`DRsmtp.example.org` where to forward mail with unqualified addresses (those without any @ domain portion).

`DHsmtp.example.org` where to send mail with local addresses (user@host without any other domain information).

`DMexample.org` The masquerade address (the domain that is used at the end of the sender's mail address).

Another way, these can be set using m4 macros using the following options:

```
SMTP_AS(`smtp.server')
UNQUALIFIED_AS(`domain.name')
LOCAL_AS(`domain.name')
```

1.3.2.3 Scripting

Below is a high level algorithm to implement the solution. For detailed Korn shell script, please **see Appendix A.6.2 & A.6.3**

```
if user wants to define SMTP server
    set from the user the fully qualified domain name of the
    <SMTP.server> and his <domain.name>
     edit file /usr/lib/sendmail-cf/cf/redhat.mc
     add the required m4 macros features and options
     save the changes
     recompile redhat.mc file and produce sendmail.cf
     stop and restart sendmail
```

1.3.3 Restricting Electronic Mail

The following scripts do the restriction for the server if electronic mail necessary needed.

1.3.3.1 Turn off SMTP `vrfy` and `expn` commands

The SMTP `vrfy` command allows a remote user to verify the E-mail address of a local user on the server. The SMTP `expn` command expands aliases and mailing list.

1.3.3.1.1 Security Issue

Local addresses and aliases should be considered confidential, or at least sensitive, information. The vrfy command will not only verify the local address but it will also not protect the privacy and confidentiality of the people. The expn command will not only expand mail aliases but it will also dump the addresses in :include: style mailing lists. So, must be protect the privacy and confidentiality of the mail by turning off the vrfy and expn commands.

1.3.3.1.2 Solution

To turn off the commands, edit `/etc/sendmail.cf` and search for the line with PrivacyOptions. Change it to read:

```
O PrivacyOptions=goaway
```

The goaway option is short for

Authwarnings	Enable X-Authentication-Warning: headers
Noexpn	Disallow all SMTP EXPN commands
novrty	Disallow all SMTP VRFY commands
needmailhelo	Require HELO before MAIL
needexpnhelo	Require HELO before EXPN
nedvrfyhelo	Require HELO before VRFY

To set this option with m4 macros, add the following to build macro (for Red Hat Linux, this is file `/usr/lib/sendmail-cf/cf/redhat.mc`) and rebuild `/etc/sendmail.cf`:

```
define(`confPRIVACY_FLAGS', `goaway')
```

then complete the procedure for m4 macro.

1.3.3.1.3 Scripting

Below is a high level algorithm to implement the solution. For detailed Korn shell script, please **see Appendix A.6.5**

```
If user wants to turn off SMTP vrfy and expn commands
     edit file /usr/lib/sendmail-cf/cf/redhat.mc
         add the required m4 macros line
         define(`confPRIVACY_FLAGS', `goaway')
     save the changes
     recompile redhat.mc file and produce sendmail.cf
     stop and restart sendmail
```

1.3.3.2 Control Mail Rely Services

1.3.3.2.1 Security Issue

Although mail relay should not be allowed from any random computer on the Internet, it may be needed for a central mail server to allow some host to rely mail through it. For Red Hat Linux, you can specify the hosts that are allowed to rely mail in the /etc/mail/access database.

1.3.3.2.2 Solution

1. To find if sendmail has the ability to use the access database check for the name of the database in `/etc/sendmail.cf`

```
Kaccess hash -o /etc/mail/access
```

If the line isn't found, the only easy way to add this feature is with the m4 macro `FEATURE(`access_db')`, then complete the procedure of m4 macro.

2. The access database has a simple "key value" format. The key is a hostname, domain or network number. The value is an action or an arbitrary message. The actions are:

REJECT	refuse connections from this host
DISCARD	accept the message but silently discard it
OK	allow access (overriding other built-in checks)
RELAY	allow access including relaying SMTP

Only hosts or domains with the RELAY action are allowed to use the mail relay. edit `/etc/mail/access` to read:

```
host or domain        RELAY
```

then restart the sendmail daemon after modifying the database.

1.3.3.2.3 Scripting

Below is a high level algorithm to implement the solution. For detailed Korn shell script, please **see Appendix A.6.6 & A.6.7**

```
if the access database is not active
     edit file /usr/lib/sendmail-cf/cf/redhat.mc
        add the required m4 macros line FEATURE(`access_db')
     save the changes
    recompile redhat.mc file and produce sendmail.cf
     stop and restart sendmail

to set rely access for domain
     edit file /etc/mail/access
     for each hosts is this file with rely enable
```

```
        if user does not wants to relay for this host
             remove this host
    get from user new hosts or domains to add to the rely
    access
        insert these hosts or domains to the file
     save the changes
     stop and restart sendmail
```

1.3.3.3 Set Domain Name Masquerading

Masquerading is the process of transforming the local hostname in addresses into that of another host. This results in the mail message appearing to come from that other host rather than the local host. Masquerading is most often used in domains where email is addressed to the domain rather than to individual hosts inside the domain.

1.3.3.3.1 Security Issue

Some e-mail clients can be set to send mail that appears to come from the domain mail server, but configuring even a small LAN of workstation to do this is tedious. The mail server can be configured to rewrite the headers of all outbound mail so that they masquerade as the central mail server. Then the e-mail addresses for all users would appear, for example as `name@example.org`

1.3.3.3.2 Solution

To define the masquerade address, use the m4 macros and add the following:

```
MASQUERADE_AS(`example.org')
FEATURE(masquerade_entire_domain)
FEATURE(allmasquerade)
FEATURE(masquerade_envelope)
```

1.3.3.3.3 Scripting

Below is a high level algorithm to implement the solution. For detailed Korn shell script, please **see Appendix A.6.8**

```
if user wants to set masquerading
```

```
set from the user the <domain.name> of the SMTP server
 edit file /usr/lib/sendmail-cf/cf/redhat.mc
 add the required m4 macros
     MASQUERADE_AS(`domain.name')
     FEATURE(masquerade_entire_domain)
     FEATURE(allmasquerade)
     FEATURE(masquerade_envelope)
 save the changes
recompile the redhat.mc file and produce sendmail.cf file
stop and restart sendmail
```

1.3.3.4 Secure The POP and IMAP Daemons

The Post Office Protocol (POP) and the Internet Message Access Protocol (IMAP) are used by client workstations to retrieve e-mail stored on the central mail server. Therefore, the mail server will have to run either the POP or IMAP daemon or both. While sendmail has gotten its act together over the last few years, there have been several nasty remote root exploits for POP and IMAP daemons in the recent past.

1.3.3.4.1 Security Issue

POP is the older simpler of the two protocols, providing basic commands for authentication, retrieval and deletion of mail messages from the mail server. IMAP is more flexible and support creating, deleting and renaming mail folders (mailboxes), searching, selective retrieval of message attributes and more.

POP/IMAP is traditionally run out of inetd, so access control through TCP wrappers is easy and very important. Limit access to only those hosts that have a legitimate need for the server.

1.3.3.4.2 Solution

Edit `/etc/hosts.allow` to allow access to the daemons by only the machines within that LAN and no others. To read:

```
ipop3d:   domain.network
imapd:    domain.network
```

1.3.3.4.3 Scripting

Below is a high level algorithm to implement the solution. For detailed Korn shell script, please **see Appendix A.6.9**

```
Edit file /etc/hosts.allow
     for each host allowed to access ipop3d and imapd
          check if user dose not wants to allow this  host
               remove this host
     get from user new hosts that are allowed to access POP3
          add them to the access list of POP3
     get from user new hosts that are allowed to access IMAP
          add them to the access list of IMAP
  save the changes
```

1.4 Securing NFS

Network File System (NFS) is a good way to distribute file systems, where it permits a client to work with files on a remote server. So NFS makes the sharing of files over a network is possible. NFS related directly with Remote Procedure Call (RPC) which construct together network protocol that allow a network of systems to operate as they a single machine.

1.4.1 NFS for Workstations

A typical workstation does not need to NFS service. However, if needed the procedure for securing NFS can be followed.

1.4.1.1 Security Issue

NFS introduces security issues by allowing other systems to access the local file system.

1.4.1.2 Solution

NFS directories that are exported should be turned off. Furthermore, to increase the security, NFS daemon must be removed.

1.4.1.3 Scripting

Below is a high level algorithm to implement the solution. For detailed Korn shell script, please **see Appendix A.7.1**

```
Turn off NFS.
Remove NFS software completely.
End the script.
```

1.4.2 NFS for Server

For the server; in the most we can not spare sharing files system among the network; i.e. we need NFS.

If we can not remove the service then we can at least restrict it by limiting the use of NFS by specific users.

1.4.2.1 Security Issue

NFS has some security problems because it is based on the Remote Procedure Call (RPC) protocol, which has a number of security holes. The RPC authentication is based on IP numbers of the clients and the UID/GID of users, all of which are easily to be spoofed.

1.4.2.2 Solution

For security matter, we should limit the accessing to NFS service. This control access is done by portmapper daemon. The portmapper is like TCP wrappers, which uses two files for allow users or deny them. These files are: `/etc/hosts.allow` and `/etc/hosts.deny`. User who are allowed to use NFS service should be listed in `hosts.allow` file and the users who are denied to use NFS should be listed in `hosts.deny` file. Those users must be specified by their IP address.

1.4.2.3 Scripting

Below is a high level algorithm to implement the solution. For detailed Korn shell script, please **see Appendix A.7.2**

```
Get from user IP addresses of machines  that they would
  be allowed use NFS.
Add these IP addresses to /etc/hosts.allow file.
End the script.
```

1.5 Securing Apache HTTP Server

The Hyper Text Transfer Protocol is the protocol that is used to request and receive documents from servers on the World Wide Web (WWW). Access to the World Wide Web is one of the driving forces behind the growth of the Internet.

One of the reasons for the success of HTTP is its simplicity. When a client contacts a WWW server, the client asks for a filename; the server responds with a document formatted in either plain ASCII or HTML (Hyper Text Markup Language). The document is then displayed. HTML documents can have embedded tags for images (which are separately retrieved) and for hypertext links to other documents. Another interesting use of the Web today involves putting programs behind web pages. Programs are created with a protocol called the Common Gateway Interface (CGI). CGI scripts can be quite simple-for example, a counter that increments every time a person looks at the page. Or they might be quite sophisticated. For example, the FedEx package-delivery service allows its customers to use the company's WWW server to trace packages.

The servers are configured so that a specified directory on the system (for example, /usr/local/etc/httpd/htdocs) corresponds with the root directory of the WWW client (for example, http://www.ora.com/).

1.5.1 Security Issue

The World Wide Web is one of the most interesting uses of the Internet. But it also poses profound security challenges. In order of importance, these challenges are:

1. An attacker may take advantage of bugs in your web server or in cgi scripts to gain unauthorized access to other files on your system, or even to seize control of the entire computer.

2. Confidential information that is on your web server may be distributed to unauthorized individuals.
3. Confidential information transmitted between the web server and the browser can be intercepted.
4. Because of the existence of standards and patented technologies, many organizations have found it necessary to purchase specially licensed software. This licensed software, in turn, can create its own unique vulnerability.

1.5.1.1 Solution

Apache HTTP server is designed for flexibility and has a wealth of features. Most security related settings are in the main configuration files found in `/etc/httpd/conf` in Red Hat Linux. The file `httpd.conf` sets up basic operating parameters for the HTTP daemon. `access.conf` sets up basic access rules, and `srm.conf` sets up basic server configuration parameters like the document root, aliased directories, CGI script directories, and icons for indexes. The approach we use is to start with a very restrictive policy, then the user selectively opens access to HTTP server directories and options. This follows the safe policy, "what is not explicitly allowed is prohibited". The following steps accomplish this policy.

1.5.1.1.1 Set Basic Access to Default Deny

In `access.conf`, set the root directory access and options to:

```
<Directory />
Options None
AllowOverride None
order deny,allow
deny from all
</Directory>
```

So, by default, access to all directories and files on the server is denied.

1.5.1.1.2 Selectively Open Access to Specific Directories

In the remainder of `access.conf`, specify only the access and options absolutely necessary. For an Intranet server, access should be allowed only to IP addresses on the internal network:

```
<Directory /home/httpd/html>
Options None
AllowOverride None
order deny, allow
deny from all
allow from 192.168
</Directory>
```

1.5.1.1.3 Selectively Allow Options on Specific Directories

Normally, for the operation of any web server, a number of cgi scripts run on the server. This is a security risk that can be minimized by limiting the capabilities of these scripts. Moreover, symbolic links can facilitate the reading of html pages, still, they can be set to access sensitive data.

A security conscious guy might legitimately think that cgi scripts pose a major security risk and should be disabled altogether. However, any practical web server now must run some cgi scripts, i.e., it's a functionality requirement. Therefore, we would rather guide the web server administrator to make informed decisions to restrict cgi scripts, rather than enabling them unknowingly. Here are the Options that we deal with in my script. They can be set in the `<Directory>` directive:

`ExecCGI`	should only be allowed for CGI directories
`FollowSymLinks`	If users have write access to the HTML directories, they can set symbolic links to areas that contain sensitive data
`Includes`	Server side includes can be used to bypass default file access restrictions
`IncludesNoexec`	Safer version of Includes that disables the `#exec` statement and `#include` of CGI scripts
`Indexes`	the daemon will print a directory listing for any directory without an index file (index.html). This may expose the names of data files ordinarily hidden

1.5.1.2 Scripting

Below is a high level algorithm to implement the solution. For detailed Korn shell script, please **see Appendix A.8.3 & A.8.4**

```
Find the root directory directive in access.conf
delete any options already there first, then add
    these lines:
      "Options None"
      "AllowOverride None"
      "order deny,allow"
      "deny from all"

Now, for each directory-except root-ask user to supply host or
network IP numbers allowed to access the directory. Insert in
the directory these lines:
              "Options None"
              "AllowOverride None"
              "order deny,allow"
              "deny from all"
              "allow from IP number list"

Finally, for each directory -except root- ask user if he wants
to set any of these options:
 {ExecCGI, FollowSymLinks, Includes, IncludesNoexec, Indexes}.
 Take the options he chose and append them in the directory.
 End of script
```


1.6 Securing FTP

File Transfer Protocol (FTP) is the proto col that performs transferring files through the network from machine to another.

The most pop ular application of FTP is downloading files across the Internet.

1.6.1 Securing FTP for Workstation

A typical workstation should disable FTP service. However, if needed, then we should be disabling the anonymous option.

1.6.1.1 Security Issues

Transferring files over a network by FTP protocol is insecure because FTP does not apply any encryption on the information that makes attacking and sniffing are possible.

1.6.1.2 Solution

For workstation case, anonymous FTP is not needed and nor recommended. Therefor, we can disable the anonymous FTP and then remove it.

The disabling is done by removing the permission to use FTP service by guest and anonymous user. This is done by replacing the following line in `/etc/ftpaccess` file:

```
class    all       real,guest,anonymous *
```

by this line:

```
class    all       real *
```

Since anonymous FTP is not going to be used, then remove the package that manages the anonymous FTP, i.e. "`anonftp`". A `anonftp` sets up the anonymous FTP home directory and correct permissions.

1.6.1.3 Scripting

Below is a high level algorithm to implement the solution. For detailed Korn shell script, please **see Appendix A.9.1**

```
Edit   /etc/ftpaccess

Replace:    class    all  real,guest,anonymous   *
By:         class    all  real
```

```
Remove anonftp: rpm –e anonftp.
End the script.
```

1.6.2 Securing FTP for Server

For servers who provide FTP services, some access control should be provided. This is done in order decrease the security holes in the system caused by weak authentication used by FTP. For anonymous FTP, we can remove if there is no need.

Generally, SSH is the modern application that replaces FTP, it is more secure and has strong authentication.

1.6.2.1 Security Issue

FTP application permits the users to exchange their data in text form without strong authentication. This is subject to attacks, e.g., sniffing userids and passwords or important information easily.

In the anonymous FTP option, like Internet FTP server, the problem is bigger where everyone uses this server and read its directories.

1.6.2.2 Solution

There is three areas work on to secure FTP server:

1. Limit access using TCP wrappers.
2. Limit permitted operations.
3. Protect incoming "directory".

Limit access with TCP wrappers:

The FTP daemon is invoked by `inetd` and protected by tcp wrapper. Thus, we can control the access across control files `/etc/hosts.allow` and `/etc/hosts.deny`.

We put the local domains of users or network who specified by server in `/etc/hosts.allow` in the format:

```
in.ftpd:  domain1, domain2, …
```

For anonymous, we put "`ALL`" as the users.

Limit permitted operations:

We control some operations that related with FTP services, like defining classes of users, limiting the number of simultaneous users, and limiting operations allowed by classes of users. This controlling is done by the `/etc/ftpaccess` file

By this changing, we little the operations allowed to the user like `delete`, `rename`, and `chmod`.

Protect incoming "directory":

For better security, remove incoming directory from the anonymous FTP service, so there is no directory with write . The location of this directory is `/home/ftp/incoming`

1.6.2.3 Scripting

Below is a high level algorithm to implement the solution. For detailed Korn shell script, please **see Appendix A.9.2 & A.9.3 & A.9.4**

```
delete the incoming directory.
limit the allowed operations in /etc/ftpaccess for guest
and anonymous users. This is done by deny the use of the
following commands in /etc/ftpaccess file. That is by
make option "no" as following configuration in the file:

        chmod       no   guest,anonymous
        delete      no   guest,anonymous
        overwrite   no   guest,anonymous
        rename      no   guest,anonymous

Get from user who can use FTP service, specific hosts or
all. Add then to the hosts.allow
End the script.
```

1.7 SSH

Secure SHell (SSH) is introduced as a solution for secure Internet communication. It is a replacement for telnet, `rlogin`, `rcp`, and `rsh` that have a number of security weaknesses. SSH provides strong authentication using RSA public/private key cryptographic algorithms and transparent encryption of network communications for login and file coping operations.

1.7.1 Start SSH the daemon

SSH has its own daemon that can be run *"stand alone"*server.

1.7.1.1 Security Issue

To enable the user using this service, we should start SSH as daemon at boot time in order to make the service remain running as long as machine is working.

1.7.1.2 Solution

The SSH daemon should be started at boot time and run until shutdown. It should not be started from `/etc/inetd.conf` because the daemon has to compute a host key every time it starts up and this causes a very high overhead. We start SSH daemon using `/etc/init.d/sshd`.

1.7.1.3 Scripting

Below is a high level algorithm to implement the solution. For detailed Korn shell script, please **see Appendix A.10.1**

```
Check if the sshd is already installed or not.
    If not:
         prompt the user to install it manually.
    If exist; then
Let the SSH daemon starts  at boot time.
End the script.
```

1.7.2 Replace "r" programs with SSH

When SSH is installed and working, there is no need now to have the "r" programs (rlogin,rsh, and rcp). So it is better to remove these programs and its daemons (in.rlogind, in.rsh, and in.rexecd).

1.7.2.1 Security Issue

The "r" programs have security weaknesses. It transfers data without any encryption or authentication that make the information incur to attacking.

1.7.2.2 Solution

Because the weakness of authentication for "r" programs, it is recommended to removing these programs and replaces them by SSH programs.

1.7.2.3 Scripting

Below is a high level algorithm to implement the solution. For detailed Korn shell script, please **see Appendix A.10.2**

```
Remove "r" programs.
Replace SSH programs with "r" programs. So S SH programs
become instead of "r" programs.
End the script.
```

1.8 Printing Services

When we need printing in the network, we should take some restriction for server. In general, the Berkely lpd (Line Printer Daemon) is shipped with Red Hat Linux. We can do some procedures to make this daemon more securing, but better than that is to replace Berkeley lpr/lpd with LPRng (LPR Next Generation).

1.8.1.1 Security Issue

The security bug happens when the system uses Red Hat lpr command. This command may make it possible to print files which the user does not have read access.

1.8.1.2 Solution

If we do not replace `lpr`/`lpd` with LPRng, we can make `lpd` more securing when specifying hosts who allowed submitting print job. This is done by putting the list in `/etc.hosts.lpd`. We can get the same effect by listing them in `/etc/hosts.equiv`, but this file is also used by the dreaded "r" programs (`rsh`, `rlogin`, etc) to determine which hosts are allowed remote shell and login access.

1.8.1.3 Scripting

Below is a high level algorithm to implement the solution. For detailed Korn shell script, please **see Appendix A.11.1**

```
Get from user IP address of machines that they would be
    allowed use printer.
Add these IP address to /etc/hosts.lpd file.
End the script.
```

1.9 Samba (SMB) Server

The SMB protocol is the core of the common Internet file system developed by Microsoft for file and printer sharing. The idea behind Samba is to make a UNIX server look exactly like any NT box to its clients in the network neighborhood. Setting up the Samba software itself is relatively simple, but there are a number of nuances to successfully integrating it into the office environment.

1.9.1 Get the latest version of Samba

1.9.1.1 Issue

The are some security problems with older version of Samba that have been announced and fixed in newer releases.

1.9.1.2 Solution

Updated packages for Red hat version 6.0 have been issued for Samba to correct security problems. Check the update for distribution, and make sure that installed Samba version 2.0.5a or later.

1.9.1.3 Scripting

Below is a high level algorithm to implement the solution. For detailed Korn shell script, please **see Appendix A.12.1**

```
edit file /etc/smb.conf and check for Samba version
     if the version is less 2.0.5
          give user massage to update Samba manually
     otherwise, continue to secure Samba
```

1.9.2 Limit Access to Specific Hosts

1.9.2.1 Security Issue

A directory exported for SMB file access is called a “share”. If a share is set up wrong you could allow anyone fore the Internet with a lowly Windows box to read and write files on the Linux Samba server.

1.9.2.2 Solution

To hedge against this possibility, restrict the hosts that have access to only those within the administrative domain of the server.

The “hosts allow” option takes a space or comma delimited list of hostnames and/or network numbers ala TCP wrappers. Localhost is included so that local users can change with Samba password with smbpasswd.

1.9.2.3 Scripting

```
edit file /etc/smb.conf in the [global] section
     get from user hosts that are allowed to access smb
          add them to the access list of smb
     interfaces to listen on
```

1.9.3 Remove “guest” shares

1.9.3.1 Security Issue

The default `/etc/smb.conf` that comes with Red Hat 6.0 only enables user-level shares of the home directories for each user on the local host. Other distributions may enable other publicly-readable guest shares.

1.9.3.2 Solution

Before running Samba, carefully inspect the shares defined in `/etc/smb.conf` and disable any that are not absolutely necessary. For the remaining shares allow write access only when absolutely necessary. Consider setting write permissions for only those users that need the permission, not for any user connected to the service.

1.9.3.3 Scripting

```
if find file /etc/smb.conf
     give user massage to change permission for these file
     manually
     otherwise, continue to secure Samba
```

1.9.4 Set default file creation masks

1.9.4.1 Security Issue

The default file creation mask leaves files world-readable on the Linux server.

1.9.4.2 Solution

If your local policy is to keep local users from reading each others file, change the mask so that none of the "other" bits are allowed.

1.9.4.3 Scripting

```
edit file /etc/smb.conf
     change the masks to be as
         create mask = 0770
         directory mask = 0750
```

1.10 Central Logging

A centralized logging host is important for organizations with multiple machines. The central log server or "loghost" provides an additional line of defense and more flexibility in monitoring the logging in the whole network. Loghost is useful tool for analyzing a system that have been compromised.

1.10.1 Configure `syslogd` to accept remote log message

To make the central log server more efficient, we try to increase the circle of machines that the server can register any logging to the system..

1.10.1.1 Security Issue

The default behavior of the `syslog` daemon in Red Hat Linux is not to accept remote log messages. This is contrary to the behavior of most `syslog` daemons. Therefore, any attack or error message produced from a remote machine; the server can not be known the server.

1.10.1.2 Solution

We should make the necessary changing in the `syslog` daemon to accept remote log messages. This is done by adding "-r" that allow the server to accept.

1.10.1.3 Scripting

Below is a high level algorithm to implement the solution. For detailed Korn shell script, please **see Appendix A.13.1**

```
Edit the syslog daemon startup file;
/etc/rc.d/init.d/syslog .
Add the option of accepting remote messages; " -r" to
become as follow:
         Start)    …
             Daemon syslogd -r
Restart the syslog daemon to operate in the last
configuration.
End the script.
```

1.10.2 Configure Log Rotation

Red Hat Linux, uses `logrotate` tool to keep log files to manageable size and retain these files and all log message for a specified period of time. Logrotation program is also designed to rotate, preserve, and delete log files after certain period of time, or when files reach a certain size.

1.10.2.1 Security Issue

For better security, we do not need to receive or train any message from log host that may be causes problem to the system. So, the logrotation for the loghost should be turned off.

1.10.2.2 Solution

To prevent any message from loghost, we delete the file that intended with like messages. This file is `/etc/logrotate.conf`. Deleting this file will turn off the log rotation of loghost.

1.10.2.3 Scripting

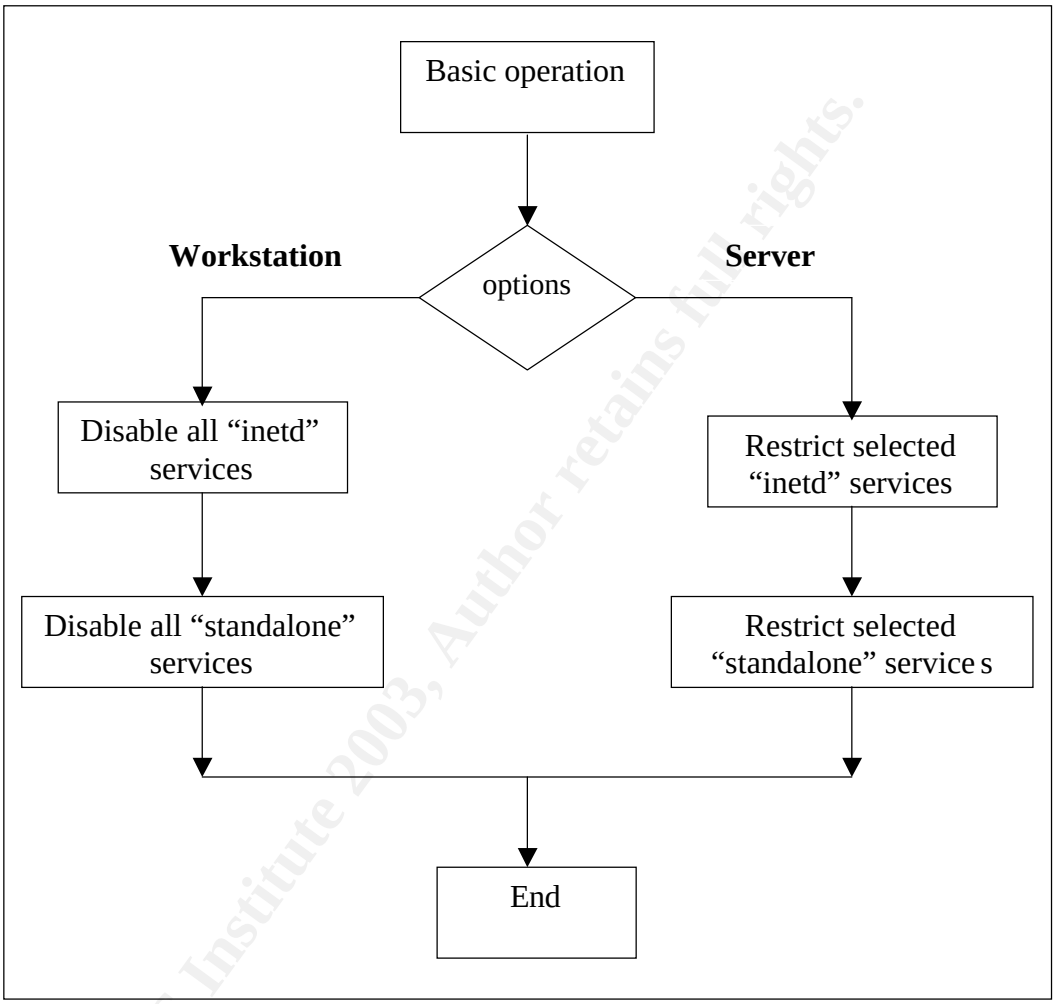
Below is a high level algorithm to implement the solution. For detailed Korn shell script, please **see Appendix A.13.2**

```
Eliminate the /etc/logrotate.conf; i.e. delete the file
End the script.
```

1.11 Final Phase: Integration

To accomplish the work, i have to integrate all the scripts above. This is the main program that calls other scripts.

The following flow chart describes the "general" structure of the final program:



So, i can divide my work into three parts: basic security operation, workstation processing, and server processing.

1.11.1 Basic Security Operation

In this section, i treat the machine regardless of the purpose of it, workstation or server. So i apply basic operation for more securing like disable rebooting from the console and password protect LILO boots.

1.11.2 Workstation Processing

If the user selects his machine to works as workstation, then there is no needed to many services. Therefore i disable all unnecessary services either if the service is inetd service or standalone.

1.11.3 Server Processing

For machines as servers, i can not disable every thing. Instead i restrict the using of services that user select.

1.11.4 Working Environment

My program interacts with user to select his options by using `dialog` program. This program is full-screen interactive, where the user choose and move between available options using keyboard buttons (Tab, Space, Enter, arrows…). The dialog available in various boxes for doing different tasks, like yes/no box, menu box, get box, text box, and other boxes.

The following steps overview the flow of my program, and also state examples for using dialog boxes:

1. In the beginning of apply the program, i prompt the user to select the kind of the machine that will be secured.
   I use dialog menu box to list the available kinds (workstation or server). When user select the kind, the program will be executed corresponding to the selection. See Figure 1.

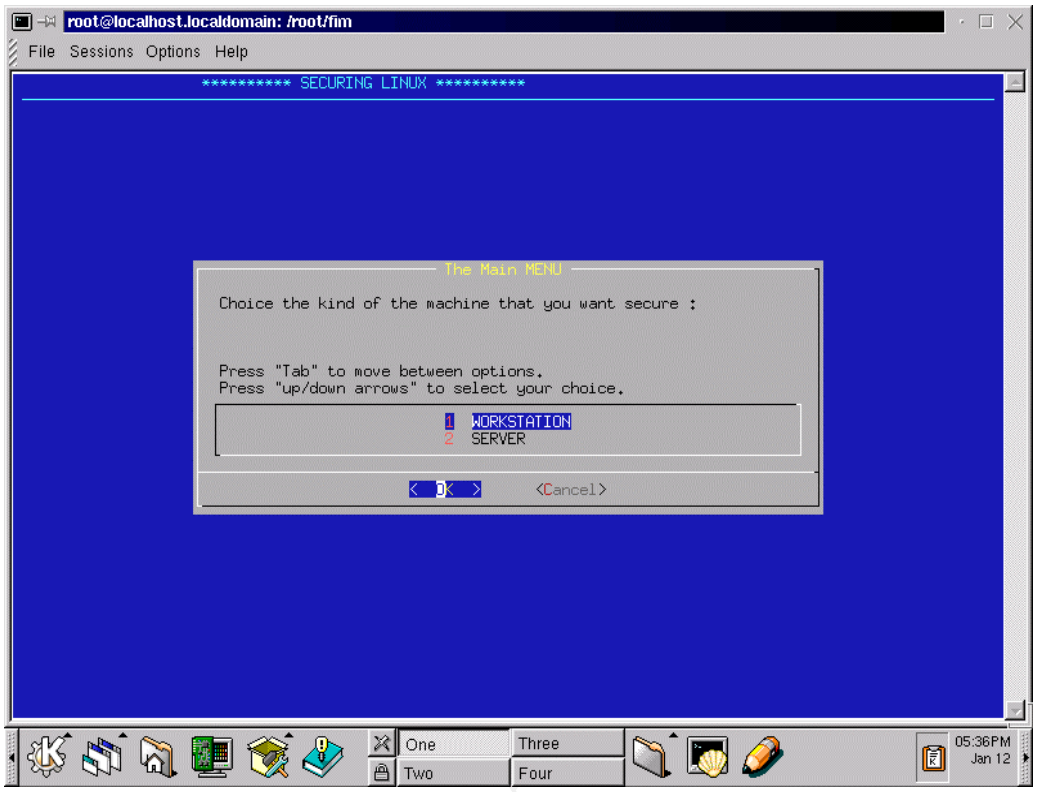


Fig.1 menu dialog at main script

2. If the selection is workstation, all services will be disabled, but if user choose server, i should do the restriction procedure. This is done in two main processing, inetd restriction and standalone resurrection. For both processing, i use dialog checklist box that enable user (by check using space button) to select the service(s) that he want to be available on the machine. Then, suitable scripts will secure the selected services. See figure 2 that show how list the inetd services in checklist box, and figure 3 for stand-alone daemons.

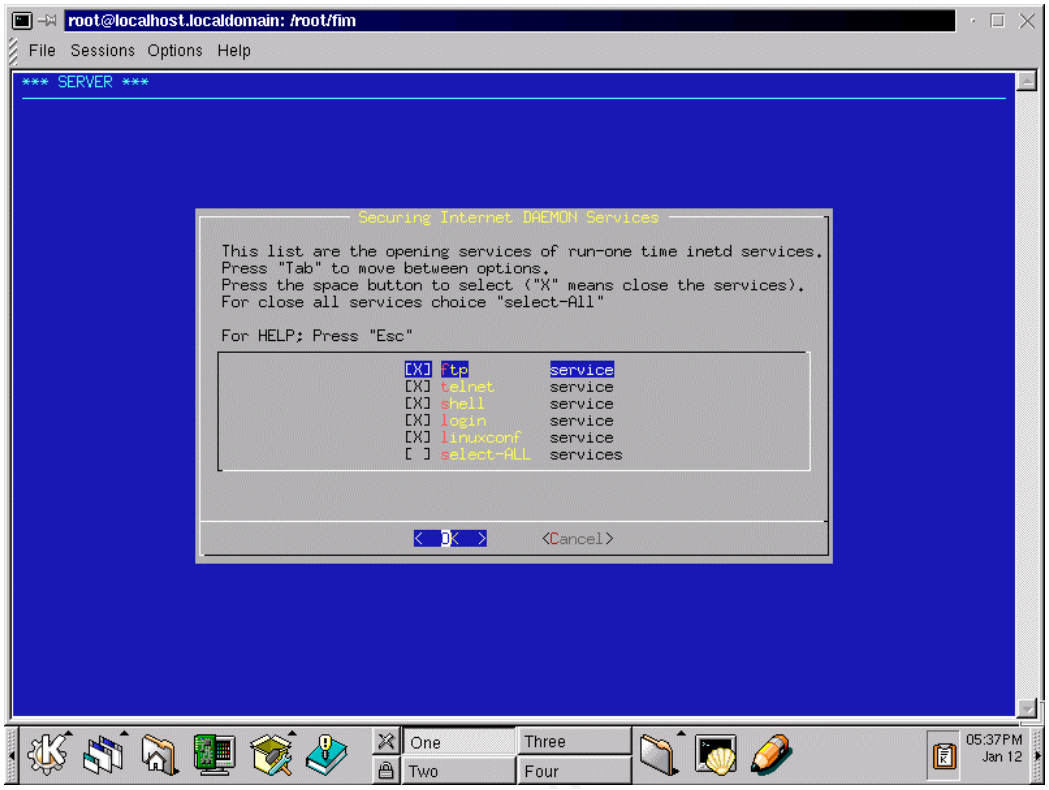


Fig. 2 Checklist dialog for selecting inetd services

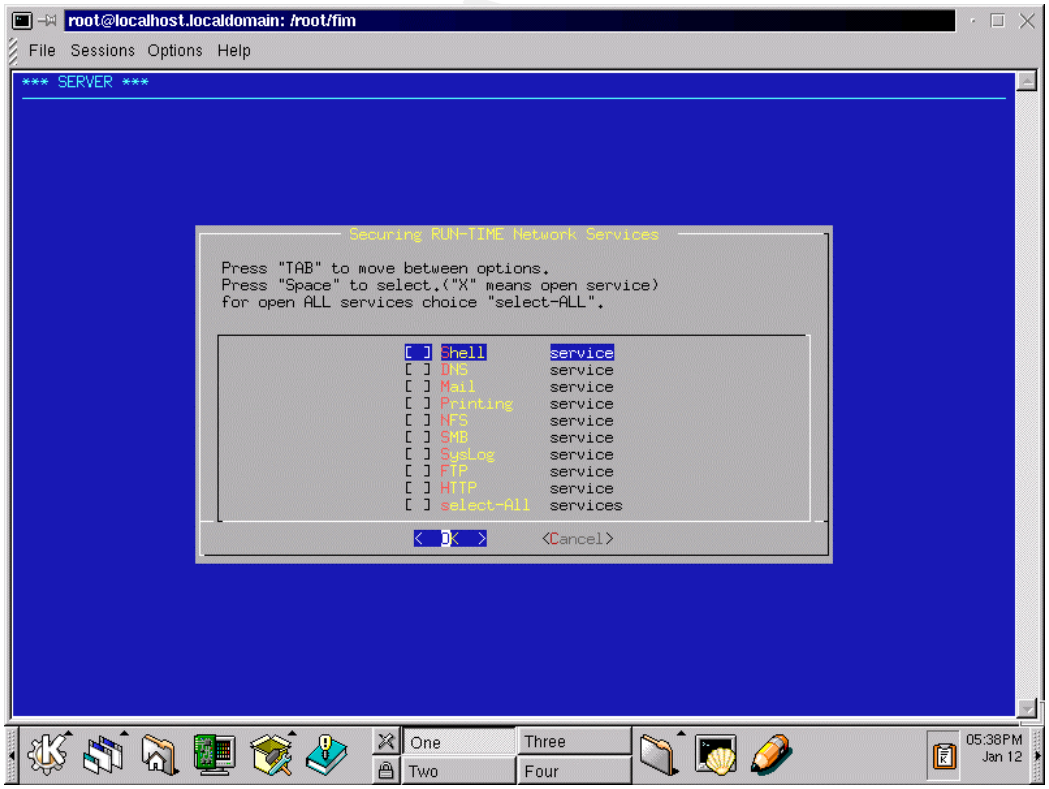


Fig.3 Checklist dialog for selecting stand-alone services

3. Another interesting and important dialog box is get-input. This box is repeated many times through the program, where it accepts the input from keyboard that make user able to enter domains or IP addresses in order to allow or deny entered IP(s) or domains from the service. See the template of the get-input box in figure 4.

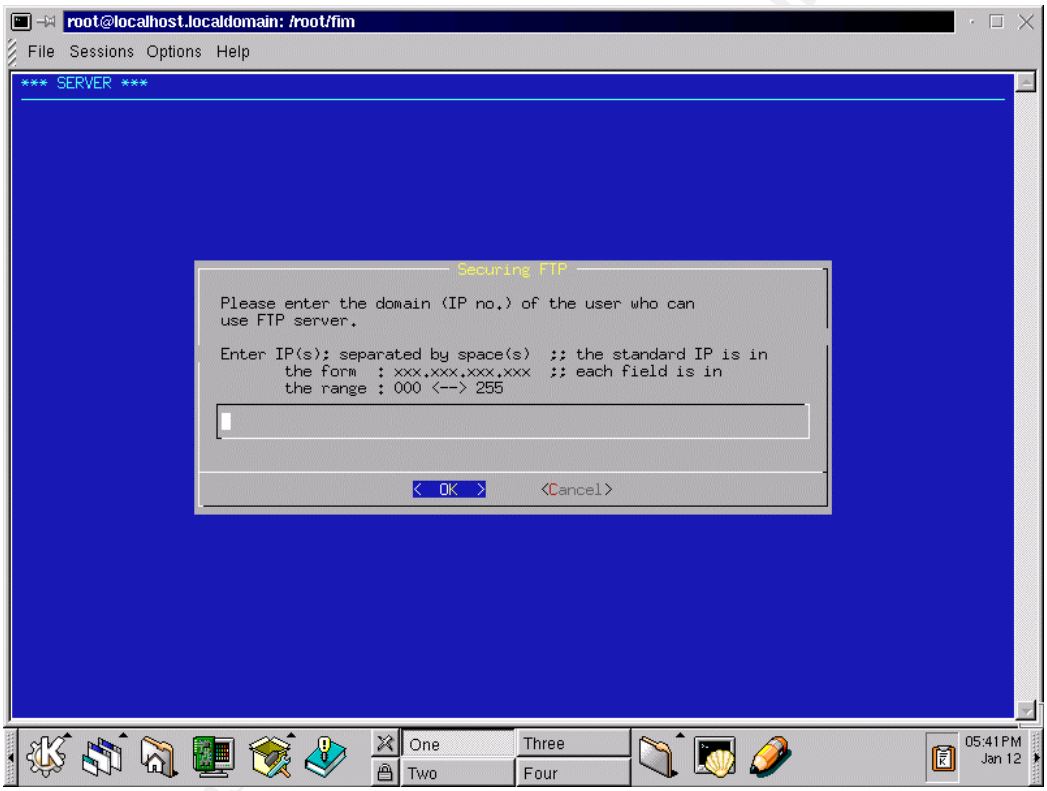


Fig. 4 get-input dialog

Conclusions

In the beginning of the project, i thought that automating the steps in the guidebook can serve all needs. Here are my conclusions about the effectiveness of automating the security steps:

- For a workstation, my program is really convenient and has one goal: disabling all services. In this mode, my program can replace manual configuration effectively. This is especially good for a novice user.
- For a server, my program can do a number of functions for each service. Its effectiveness in this mode depends on the requirements of the user. For example, i use a "default deny" policy for TCP Wrappers. If the user wants a different policy, he will have to do it manually. If he wants the same policy, my program can be effective for him.
- In conclusion, my program is effective and helpful in some circumstances.

References

1. The SANS institute, "Securing Linux Step-By-Step", 1999-2000
2. Simson Garfinkel and Gene Spafford, "Practical Unix and Internet Security", O'Reilly & Associates Inc, 2nd Ed. 1996
3. http://www.ibiblio.org/mdw/HOWTO/
4. Gerhard Mourani, "Get Acquainted with Linux Security and Optimization System 1.1", Open Network Architecture, 1999
5. http://new.linuxnow.com/tutorial/intro/intro.html
6. http://www.linuxdoc.org/LDP/
7. http://www.infocom.cqu.edu.au/Units/aut98/85321/Stud y_Material/ Text_Book/chap7/html.html
8. http://www.linuxprinting.org/projects.html
9. http://www.interhack.net/pubs/network-security.html
10. Craig Hunt, "TCP/IP Network Administration", O'Reilly & Associates, Inc. 1st Ed. 1992
11. Simple Mail Transfer Protocol, John C. Klensin, Editor
12. Bryan Costales with Eric Allman, "Sendmail", O'Reilly & Associates, Inc. 2nd Ed. 1997
13. http//us1.samba.org/samba/docs/SambaIntro.html
14. Andrew Tanenbaum, "Computer Networks", Prentice-Hall, Inc 3rd Ed. 1996

Appendix A

Korn Shell Scripts of Securing Linux

In this appendix, i list the source code (Korn Shell script) that implements the security steps of Securing Linux.

2.1 Main of Securing Linux

```
#!/bin/ksh
#RUN_ME: Main of Securing Linux


bak="                    ********** SECURING LINUX **********"
tit="The Main MENU"
txt="
Choose the kind of the machine that you want to secure :



Press \"Tab\" to move between options.
Press \"up/down arrows\" to select your choice."

dialog --backtitle "$bak" --title "$tit" --menu "$txt" 15 70 2\
 1 "WORKSTATION" 2 "SERVER" 2> f
select=$?
x=`cat f`
if [ $select = 0 ]; then
      if [ $x = 1 ]; then
            ksh /root/fim/code/workstation/main.ws 2> /root/fim/RESULT
      else
            ksh /root/fim/code/server/main.sv  2> /root/fim/RESULT
      fi
fi
rm -f f
```

2.1.1 Main program for Workstation

```
#!/bin/ksh
#main.ws: Main program for Workstation

ksh /root/fim/code/workstation/s3.1.1
/etc/rc.d/init.d/inet stop
/sbin/chkconfig inet off

ksh /root/fim/code/workstation/ s3.2.1

ksh /root/fim/code/workstation/s3.3.1

ksh /root/fim/code/workstation/s3.5.1

ksh /root/fim/code/workstation/s3.6.1

ksh /root/fim/code/workstation/s3.7.1

ksh /root/fim/code/workstation/s3.8.1

ksh /root/fim/code/workstation/s3.9.1

ksh /root/fim/ code/workstation/s3.5.2.ws

ksh /root/fim/code/workstation/s3.6.2.ws
```

2.1.2 Main Program of Server

```
#!/bin/ksh
#main.sv: Main Program of Server

ksh /root/fim/code/server/inetd
ksh /root/fim/code/server/runtime
```

2.1.2.1 Main Program for inetd Services

```
#!/bin/ksh
#inetd: Main Program for inetd Services

# an argument list of the services to be closed must be passed in the
# command line.
# Enter: ksh S3.1.1 `grep -v "^#" inetd.conf|awk '{print $1}'`
# Side effect: if all services are already commented, the command will
# stop, waiting for arguments. Solutin: grep first then check the exit
# code. If 0, then run the command otherwise dont.

bak="*** SERVER ***"
tit="Securing Internet DAEMON Services"
txt="
This list shows the opened services of one -time inetd services.

Press \"Tab\" to move between options.
Press the space button to select (\"X\" : close the service).
For close all services choice \"Close-ALL\"

For HELP; Press \"Esc\""
```

```
xx=`grep -v "^#" /etc/inetd.conf|awk '{print $1}'`
cc=`echo $xx|wc -w`

((i=1))
for i in $xx; do
      var=$var"$i service on "
      var1=$var1"$i service "
      ((i++))
done

listing=$var"Closing-Internet daemon off"

((size=$cc+15+1))
((lst=$cc+1))
#############################################
stop=1
while [ $stop = 1 ]; do

dialog --backtitle "$bak" --title "$tit" --checklist "$txt" \
$size 70 $lst $listing 2>ffff
ddd=$?
        if [ $ddd = 0 ];then
            v=`grep -q "select" ffff`
            if [ $? = 0 ]; then
                  ksh /root/fim/code/workstation/s3.1.1
                  /etc/rc.d/init.d/inet stop
                  /sbin/chkconfig inet off
            else
                  cp -f /etc/inetd.conf file0
                  for i in `cat ffff`;do
                  j=`echo $i|sed s/\"//g`
                  sed s/^$j/#$j/ /etc/inetd.conf > file0
                  cp -f file0 /etc/inetd.conf

                  done

      fi
        stop=0
      fi

      if [ $ddd = 1 ]; then
            stop=0
      fi

        if [ $ddd = 255 ]; then
            stophelp=1
            while [ $stophelp = 1 ]; do
                  ((high=$cc+12))
msg="
Press \"Enter\" to enter the HELP window.

Choose \"Cancel\" to return to main menu"
dialog --backtitle "$bak" --title "$tit" --menu "$msg" \
$high 70 $cc $var1 2>fff
                  if [ $? = 0 ];then
                        x=`cat fff`
dialog --backtitle "$bak" --title "HELP" --textbox fff 15 70
                        stophelp=1
                  else
                        stophelp=0
                  fi
```

```
            done
          stop=1
        fi
done
########################################
rm -f fff ffff file0 file1

ksh /root/fim/code/workstation/s3.2.1
ksh /root/fim/code/workstation/s3.2.2
```

2.1.2.2 Main Program for Run-Time Network Services

```
#!/bin/ksh
#runtime: Main Program for Run-Time Network Services

###--> create file1; that will contain "ALL" service s that the user select
#     from its what the services he want open or not

echo "Shell DNS Mail Printing NFS SMB SysLog FTP HTTP select -All"> file1

bak="*** SERVER ***"
tit="Securing RUN-TIME Network Services "
txt="
The Internet daemon handels transient \"one-time\" network connections,
but there are other network daemons that start at boot-time and
normally run until system shutdown, as \"httpd\" for the Web,\"named\"
for DNS services, and \"smbd\" for Samba SMB networking services.
They are started at boot-time by the \"init\" process (process number 1).



Press Enter to continue ..."

txt1="
Press \"TAB\" to move between options.
Press \"Up/Down Arrows\" to move between services.
Press \"Space\" to select.(\"X\" means open service)
for open ALL services choice \"Open-ALL\".
"
list="SSH service off DNS service off Mail service off  \
Printing service off NFS service off SMB service off  \
SysLog service off FTP service off HTTP service off  \
Open-All services off"

dialog --backtitle "$bak" --title "$tit" \
--msgbox "$txt" 15 75

dialog --backtitle "$bak" --title "$tit" \
--checklist "$txt1" 20 70 10 $list 2>file2

###--> file2 contain services that user select; these services will be
#     restricted. The restriction mainly do by step4.

###--> Here; you select "ok". i.e. execute the script

dialexit=$?

if [ $dialexit = 0 ]; then
      x1=`grep "Open-All" file2`
      if [ $? = 0 ]; then
```

```
            ksh /root/fim/code/server/s4.2.2
            ksh /root/fim/code/server/s4.2.6
            ksh /root/fim/code/server/s4.3.1
            ksh /root/fim/code/server /s4.3.3
            ksh /root/fim/code/server/s4.4
            ksh /root/fim/code/server/s4.5.1
            ksh /root/fim/code/server/s4.6.1
            ksh /root/fim/code/server/s4.7.1
            ksh /root/fim/code/server/s4.8.1
            ksh /root/fim/code/server/s4.8.2
            ksh /root/fim/code/server/s4.9.1
            ksh /root/fim/code/server/s4.9.2
            ksh /root/fim/code/server/s4.9.3
            ksh /root/fim/code/server/s4.10.1
            ksh /root/fim/code/server/s4.10.2
      else

###---> create file3; this file will contain the services that will turned
#     off

      echo >file3
      for i in `cat file1`; do
            x=`grep -v $i file2`
            if [ $? = 0 ];then echo $i >> file3;fi
      done


######################### Start the restriction #################

x=`grep "SSH" file2`
if [ $? = 0 ]; then
      ksh /root/fim/code/server/s4.2.2
      ksh /root/fim/code/server/s4.2.6
fi

x=`grep "DNS" file2`
if [ $? = 0 ]; then
      ksh /root/fim/code/server/s4.3.1
      #ksh /root/fim/code/server/s4.3.2
      ksh /root/fim/code/server/s4.3.3
fi

x=`grep "Mail" file2`
if [ $? = 0 ]; then
      ksh /root/fim/code/server/s4.4
fi

x=`grep "Printing" file2`
if [ $? = 0 ]; then
      ksh /root/fim/code/server/s4.5.1
fi

x=`grep "NFS" file2`
if [ $? = 0 ]; then
      ksh /root/fim/code/server/s4.6.1
fi

x=`grep "SMB" file2`
if [ $? = 0 ]; then
      ksh /root/fim/code/server/s4.7.1
      #ksh /root/fim/code/server/s4.7.2
```

```
      #ksh /root/fim/code/server/s4.7.3
      #ksh /root/fim/code/server/s4.7.4
      #ksh /root/fim/code/server/s4.7.5
fi

x=`grep "Syslog" file2`
if [ $? = 0 ]; then
      ksh /root/fim/code/server/s4.8.1
      ksh /root/fim/code/server/s4.8.2
fi

x=`grep "FTP" file2`
if [ $? = 0 ]; then
      ksh /root/fim/code/server/s4.9.1
      ksh /root/fim/code/server/s4.9.2
      ksh /root/fim/code/server/s4.9.3
fi

x=`grep "HTTP" file2`
if [ $? = 0 ]; then
      echo;
      ksh /root/fim/code/server/s4.10.1
      ksh /root/fim/code/server/s4.10.2
fi
#################################################################

x=`grep "SSH" file3`
if [ $? = 0 ]; then
      /etc/rc.d/init.d/sshd stop >> /root/fim/result
      /sbin/chkconfig sshd off >> /root/fim/result
fi

x=`grep "DNS" file3`
if [ $? = 0 ]; then
      ksh /root/fim/code/ workstation/s3.5.1
      ksh /root/fim/code/workstation/s3.5.2
fi

x=`grep "Mail" file3`
if [ $? = 0 ]; then
      #/etc/rc.d/init.d/sendmail stop >> /root/fim/result
      #/sbin/chkconfig sendmail off >> /root/fim/result
      ksh /root/fim/code/workstation/s3.6.1
      ksh /root/fim/code/workstation/s3.6.2
fi

x=`grep "Printing" file3`
if [ $? = 0 ]; then
      /etc/rc.d/init.d/lpd stop >> /root/fim/result
      /sbin/chkconfig lpd off >> /root/fim/result
fi

x=`grep "NFS" file3`
if [ $? = 0 ]; then
      ksh /root/fim/code/workstation/s3.7 .1
fi

x=`grep "SMB" file3`
if [ $? = 0 ]; then
      /etc/rc.d/init.d/smb stop >> /root/fim/result
      /sbin/chkconfig smb off >> /root/fim/result
```

```
fi

x=`grep "Syslog" file2`
if [ $? = 0 ]; then
      /etc/rc.d/init.d/rpc.mountd stop >> /root/fim/result
      /sbin/chkconfig rpc.mountd off >> /root/fim/result
      /etc/rc.d/init.d/rpc.rquotad stop >> /root/fim/result
      /sbin/chkconfig rpc.rquotad off >> /root/fim/result
      /etc/rc.d/init.d/rpc.statd stop >> /root/fim/result
      /sbin/chkconfig rpc.statd off >> /root/fim/result
fi

x=`grep "FTP" file3`
if [ $? = 0 ]; then
      ksh /root/fim/code/workstation/s3.9.1
fi

x=`grep "HTTP" file3`
if [ $? = 0 ]; then
      ksh /root/fim/code/workstation/s3.8.1
      #ksh /root/fim/code/workstation/s3.8.2
fi

############################################### ######################
###--> The following loop in order to close any rest services have not
#     closed yet

x=`lsof -i -Fc | grep '^c' |cut -b2-20 | sort -u`
x1=`echo $x|sed /"inet"/d > y`
#for i in `lsof -i -Fc | grep '^c' |cut -b2-20 | sort -u`; #do
for i in $y; do
      x=`grep -v $i file2`
      if [ $? = 0 ]; then

mesg="
Do you want to close $i services ?



For BETTER security; select YES ."
dialog --backtitle "$bak" --title "Securing SERVER"  --yesno "$mesg" 12 70

            if [ $? = 0 ]; then
                 /etc/rc.d/init.d/$i stop
                 /sbin/chkconfig $i off
            fi
      fi
done

fi
fi

if [ $dialexit = 1 ]; then echo ;fi
if [ $dialexit = 255 ]; then
      #echo "help"
      dialog --msgbox "HELP" 10 70
fi

rm -f result
rm -f file1
rm -f file2
```

```
rm -f file3
rm -f y
```

2.1.3 inetd Services

2.1.3.1 Disable Internet Daemon Services of workstation

```
#!/bin/ksh
#s3.1.1: Disable Internet Daemon Services of workstation
# an argument list of the services to be closed must be passed in the
# command line.
# Enter: ksh S3.1.1 `grep -v "^#" inetd.conf|awk '{print$1}'`
# Side effect: if all services are already commented, the command will
# stop, waiting for arguments. Solutin: grep first then check the exit
# code. If 0, then run the command otherwise dont.

j=0;k=1
cp -f /etc/inetd.conf file0
for i in `grep -v "^#" file0|awk '{print $1}'`;do
   sed s/^$i/#$i/ file$j > file$k
   ((t=$j))
   ((j=$k))
   ((k=$t))
done
cp -f file$j /etc/inetd.conf
rm -f file0 file1
```

2.1.3.2 Disable Internet Daemon Services of Server

```
#!/bin/ksh
#inetd-services.cod: Disable Internet Daemon Services  of Server

# an argument list of the services to be closed must be passed in the
# command line.
# Enter: ksh S3.1.1 `grep -v "^#" inetd.conf|awk '{print$1}'`
# Side effect: if all services are already commented, the command will
# stop, waiting for arguments. Solutin: grep first then check the exit
# code. If 0, then run the command otherwise dont.

i=1
while [ $i -le $# ]; do
  var=$var"\$$i ''  on "
  var1=$var1"\$$i '' "
  ((i++))
done

listing=$var"select All ''"
((size=$#+10+1))
y=0
((lst=$#+1))
while [ $y = 0 ]; do

cmmnd="dialog --backtitle $baktit --title $tit --separate-output
--checklist
$txt $size 70 $lst $listing 2>ffff"

eval $cmmnd
      y=$?
        if [ $y = 0 ];then
```

```
            v=`grep -q "select" ffff`
            if [ $? = 0 ]; then
            /etc/rc.d/init.d/inet st op
            /sbin/chkconfig inet off
            fi
            y=1
            j=0;k=1
            cp -f /etc/inetd.conf file0
            for i in `cat ffff`;do
                    sed s/^$i/#$i/ file$j > file$k
                ((t=$j))
                ((j=$k))
                ((k=$t))
            done
            cp -f file$j /etc/inetd.conf
        elif [ $y = 1 ];then
            y=99
        else
            z=0
            while [[ $z = 0 ]]; do
                cmmnd1="dialog --menu "HELP" $size 35 $# $var1
2>fff"
                eval $cmmnd1
                if [ $? = 0 ];then
                    x=`cat fff`
                    dialog --textbox $x.fim.h 10 30
                    z1=0
                else
                    z1=1
                fi
               z=$z1
            y=0
            done
      fi
done
rm -f fff ffff file0 file1
ksh /root/fim/code/workstation/s3.2.1
ksh /root/fim/code/workstation/s3.2.2
```

2.1.3.3 Set the default access rule to deny all

```
#!/bin/ksh
#s3.2.1: Set the default access rule to deny all

grep -q "ALL: ALL" /etc/hosts.deny
if [ $? != 0 ];then
echo "ALL: ALL" >> /etc/hosts.deny
echo "ypserv: ALL" >> /etc/hosts.deny
fi
```

2.1.3.4 Allow access to only specific hosts for specific services

```
#!/bin/ksh
#s3.2.2: Allow access to only specific hosts for specific s ervices

bak="*** SERVER ***"
tit="Allow access to specific services"
x1="
We are now going to allow specific hosts for open
```

```
services (ftp,telnet,shell,....)

Press Enter to continue ..."
grep -q -v "#" /etc/inetd.conf
if [[ $? = 0 ]]
then dialog --backtitle "$bak" --title "$tit" --msgbox "$x1" 12 70
fi
for i in `grep -v "#" /etc/inetd.conf | awk '{print $1}'` ; do

x2="
Do you want to open {$i} for :
"
dialog --backtitle "$bak" --title "$tit" --menu "$x2" 12 70 2\
 1 "Specific hosts" 2 "ALL hosts" 2>f

x=`cat f`
if [ $? = 0 ] ; then
      if [ $x = 2 ];then
            echo "$i: ALL" >> /etc/hosts.allow
      else

      export IPbak=$bak
      export IPtit=$tit
      export IPmesg="
      please enter allowed hosts for \"$i\"
      "
      ksh /root/fim/code/get-IP/INPUT-IP
      x4=`cat /root/fim/code/get-IP/IPnumber`
      echo "$i: $x4" >> /etc/hosts.allow
fi
fi
done
```

2.1.4 Stand-alone Services

2.1.4.1 Disable Run-Time Network Services

```
#!/bin/ksh
#s3.3.1: Disable Run-Time Network Services

# you must execute this step in root.
# for the W.S, no need for options, hence the commented i/o and if.

x="This time, we will shutdown all stand-alone network services..."

for i in ` lsof -i -Fc|grep '^c'|cut -b2-20|sort -u  `;do
/etc/rc.d/init.d/$i stop
/sbin/chkconfig $i off
done
```

2.1.5 DNS Service

2.1.5.1 Disable and remove DNS server

```
#!/bin/ksh
#s3.5.1: Disable and remove DNS server

x=$(lsof -i -Fc|grep '^c'|cut -b2-20|sort -u|grep named|wc -w)
```

```
if [ x -gt 0 ]
      then /etc/rc.d/init.d/named stop
fi

if [ `rpm -qa|grep caching-nameserver-[0-9]|wc -w` -gt 0 ]
      then  rpm -e caching-nameserver
      else
            x="{caching-nameserver} is already uninstalled"
fi
y=$(rpm -qa|grep ^bind-[0-9]|wc -w)
if [ $(rpm -qa|grep ^bind-[0-9]|wc -w) -gt 0 ]
      then  rpm -e bind
      else    echo "{bind} is already uninstalled"
fi
```

2.1.5.2 Set Primary and secondary name server

```
#!/bin/ksh
#s3.5.2: Set Primary and secondary name server

bak="*** SERVER ***"
tit="Identifying DNS server"
txt1="

Do you want to set up a Primary DNS server for your workstation?"
dialog --backtitle "$bak" --title "$tit" --yesno "$txt1" 10 70

if [ $? = 0 ]; then
msg="
Enter the name & IP for the primary name server : "
dialog --backtitle "$bak" --title "$tit" --inputbox "$msg" 10 70 2>file
x2=$(cat file)
echo "$x2">>/etc/resolv.conf
fi

txt2="

Do you want set a Secondary DNS server ?"
dialog --backtitle "$bak" --title "$tit" --yesno "$txt2" 10 70

if [ $? = 0 ]; then
msg="
Enter the name & IP for the secondary name server : "
dialog --backtitle "$bak" --title "$tit" --inputbox "$msg" 10 70 2>file
x2=$(cat file)
echo "$x2">>/etc/resolv.conf
fi

#more /etc/resolv.conf
rm -f file
```

2.1.5.3 Set Primary and Secondary name server for Workstation

```
#!/bin/ksh
#s3.5.2: Set Primary and Secondary name server for Workstation

bak="*** WORKSTATION ***"
tit="DNS Configuration"
```

```
txt1="

Do you want to set up a Primary DNS server for your workstation?"
dialog --backtitle "$bak" --title "$tit" --yesno "$txt1" 10 70

if [ $? = 0 ]; then
msg="
Enter the name & IP number for the primary name server : "
dialog --backtitle "$bak" --title "$tit" --inputbox "$msg" 10 70 2>file
x2=$(cat file)
echo "$x2">>/etc/resolv.conf
fi

txt2="

Do you want set up a Secondary DNS server ?"

dialog --backtitle "$bak" --title "$tit" --yesno "$txt2" 10 70

if [ $? = 0 ]; then
msg="
Enter the name & IP number for the secondary name  server : "
dialog --backtitle "$bak" --title "$tit" --inputbox "$msg" 10 70 2>file
x2=$(cat file)
echo "$x2">>/etc/resolv.conf
fi

rm -f file
```

2.1.5.4 Restrict zone transfers

```
#!/bin/ksh
#s4.3.1: Restrict zone transfers

# This line is a stub to fetch and check t he sec. DNS ip's from the user.
# assume it stores the valid IPs in variable "sec_dns_ips"

# here is a test value for the variable. remove it when done with the stub
# also, remove test lines( like echo $j) when step is complete
export IPbak="*** SERVER * **"
export IPtit="Securing DNS"
export IPmesg="
Please Eneter IP for DNS server :

"
ksh /root/fim/code/get-IP/INPUT-IP
sec_dns_ips=`cat /root/fim/code/get-IP/IPnumber`
#sec_dns_ips=" 192.168.2.3; 134.4.2.5; 56.4.2.5; 75;2.76.9; "
name="/etc/named.conf"
cp -f $name file1

a=1;b=2;displacement=1
for j in `grep -n -E "^[[:space:]]+type[[:space:]]+master" $name| \
cut -f1 -d:`
do
   #echo $j;#echo $sec_dns_ips
   ((k=$j + $displacement))
   xpr1="'$k a\'"
   xpr2="' \ \ \ \ \ \ allow-transfer { $sec_dns_ips };'"
   eval sed -e $xpr1 -e $xpr2 file$a > file$b
```

```
   ((displacement+=1))
   t=$a
   a=$b
   b=$t
done

# This time, the loop disables zone transfers for " type slave " NS

displacement=1
for j in `grep -n -E "^[[:space:]]+type[[:space:]]+slave" file$a |\
cut -f1 -d:`
do
   ((k=$j + $displacement))
   xpr1="'$k a\'"
   xpr2="' \ \ \ \ \ \ allow-transfer { none; };'"
   eval sed -e $xpr1 -e $xpr2 file$a > file$b

   ((displacement+=1))
   t=$a
   a=$b
   b=$t
done
cp -f file$a $name
rm -f file$a file$b
```

2.1.5.5 Run named in a chroot jail

```
#!/bin/ksh
#s4.3.3: Run named in a chroot jail

# Remember, if the user does change the default UID/GID in the inputbox
# , to check again that the new values are not used in passwd.

bak="*** SERVER ***"
tit="Securing DNS"
txt="
For better security, the service can be set up to run in its
own chroot directory tree. Setting up a proper chroot jail must
be done with care.

You can choose the following default chroot directory tree :

"

current=`pwd`
uid=`awk -F: 'END {print $3}' /etc/passwd`
echo $uid > ftest
list=`cut -f3 -d: /etc/passwd`
ok=0
echo $list >> ftest
while [ $ok -eq 0 ]; do
  echo $list | grep -q $uid
  ok=$?
  ((uid+=1))
done
gid=$uid

dialog --backtitle "$bak" --title "$tit" --inputbox "$txt" 15 70 "dns;$uid;
$gid; /home/dns" 2>dns_settings
```

```
uname=`cut -f1 -d";" dns_settings`
uid=`cut -f2 -d";" dns_settings`
gid=`cut -f3 -d";" dns_settings`
jail=`cut -f4 -d";" dns_settings`

groupadd -g $gid $uname
useradd -u $uid -g $gid -M $uname

mkdir -m 0700 $jail
cd $jail
mkdir -p etc lib dev usr/sbin var/named var/run
mknod -m 666 dev/null c 1 3

cp -f /etc/named.conf etc
cp -R -f /var/named/* var/named
chown -R dns.dns var/named var/run
cp -f /usr/sbin/{named,named-xfer} usr/sbin

cp -f /lib/libc.so.6 lib
cp -f /lib/ld-linux.so.2 lib

syslogfile=/etc/rc.d/init.d/syslog
l=`grep -E -n "daemon[[:space:]]+syslog" $syslogfile|cut  -f1 -d:`
xpr1="'$l c\'"
xpr2="'\ \ \ \ \ \ \ daemon syslogd -a /home/dns/dev/log'"
eval sed -e $xpr1 -e $xpr2 $syslogfile>fimtemp; cp  -f fimtemp $syslogfile

chmod 755 /etc/rc.d/init.d/syslog
/etc/rc.d/init.d/syslog restart

namedfile=/etc/rc.d/init.d/named
l=`grep -E -n "daemon[[:space:]]+named" $namedfile|cut  -f1 -d:`
xpr1="'$l c\'"
xpr2="'\ \ \ \ \ \ \ daemon named -u dns -g dns -t /home/dns'"
eval sed -e $xpr1 -e $xpr2 $namedfile>fimtemp; cp  -f fimtemp $namedfile

chmod 755 /etc/rc.d/init.d/named
/etc/rc.d/init.d/named restart
cd $current
rm -f ftest
rm  -f dns_settings
```

2.1.6 Electronic Mail Service

2.1.6.1 Turn off sendmail daemon mode

```
#!/bin/ksh
#s3.6.1: Turn off sendmail daemon mode

x="Do you want to turn off sendmail service ? "

if [ $? = 0 ] ; then
      x1=`grep "DAEMON" /etc/sysconfig/sendmail`
      x2="DAEMON=no"

      if [ `grep "DAEMON" /etc/sysconfig/sendmail` ];then
            sed s/"$x1"/"$x2"/ /etc/sysconfig/sendmail > file
      else
            echo "DAEMON=no" >> file
      fi
```

```
      cp -f file /etc/sysconfig/sendmail

      x1=`grep  "QUEUE" /etc/sysconfig/sendmail`
      x2="QUEUE=15m"
      if [ `grep "QUEUE" /etc/sysconfig/sendmail` ];then
            sed  s/"$x1"/"$x2"/ /etc/sysconfig/se ndmail > file
      else
            echo "QUEUE=15m" >> file
      fi
      cp -f file /etc/sysconfig/sendmail

      /etc/rc.d/init.d/sendmail restart
fi
rm -f file
```

2.1.6.2 Define SMTP Se rver

```
#!/bin/ksh
#s3.6.2: Define SMTP Server

bak="*** SERVER ***"
tit="Electronic Mail Configuratio n"
txt="

Do you want to set up SMTP server for your workstation ?
dialog --backtitle "$bak" --title "$tit" --yesno "$txt" 10 70
if [ $? = 0 ]; then
export IPbak=$bak
export IPtit=$tit
export IPmesg="
Enter your IP SMTP server : "
ksh /root/fim/code/get-IP/INPUT-IP
x2=`cat /root/fim/code/get-IP/IPnumber`
x3=`grep "^DS" /etc/sendmail.cf`
x4=`grep "^DR" /etc/sendmail.cf`
x5=`grep "^DH" /etc/sendmail.cf`
sed s/"$x3"/"DS$x2"/ /etc/sendmail.cf |sed s/"$x4"/"DR$x2"/ |
sed s/"$x5"/"DH$x2"/  >sendmail1.cf
ww=0
ww=`cut -f2-3 -d"." /root/fim/code/get-IP/IPnumber`
x6=`grep "^DM" /etc/sendmail.cf`
sed s/"$x6"/"DM$ww"/ sendmail1.cf >sendmail2.cf

mv -f sendmail2.cf /etc/sendmail.cf
rm -f sendmail1.cf
rm -f file
/etc/rc.d/init.d/sendmail restart
fi
```

2.1.6.3 Define SMTP Server for Workstation

```
#!/bin/ksh
#s3.6.2.ws: Define SMTP Server for Workstation

bak="*** WORKSTATION ***"
tit="Electronic Mail Configuration"
txt="
```

```
Do you want to set up SMTP server for your workstation ?"
dialog --backtitle "$bak" --title "$tit" --yesno "$txt" 10 70
if [ $? = 0 ]; then
export IPbak=$bak
export IPtit=$tit
export IPmesg="
Enter your IP SMTP server : "
ksh /root/fim/code/get-IP/INPUT-IP

x2=`cat /root/fim/code/get-IP/IPnumber`
x3=`grep "^DS" /etc/sendmail.cf`
x4=`grep "^DR" /etc/sendmail.cf`
x5=`grep "^DH" /etc/sendmail.cf`
sed s/"$x3"/"DS$x2"/ /etc/sendmail.cf |sed s/"$x4"/"DR$x2"/ |
sed s/"$x5"/"DH$x2"/  >sendmail1.cf
ww=0
ww=`cut -f2-3 -d"." /root/fim/code/get-IP/IPnumber`
x6=`grep "^DM" /etc/sendmail.cf`
sed s/"$x6"/"DM$ww"/ sendmail1.cf >sendmail2.cf

mv -f sendmail2.cf /etc/sendmail.cf
rm -f sendmail1.cf
rm -f file
/etc/rc.d/init.d/sendmail restart
fi
```

2.1.6.4 Restricting Electronic Mail

```
#!/bin/ksh
#s4.4: Restricting Electronic Mail

bak="*** SERVER ***"
tit="Securing Electronic MAIL"
msg="
WELCOME ... We will secure Electronic Mail ...

\"sendmail.cf\" is the file by which you can configure
basic information about your machine.
You specify the intended purpose of your
computer, mail wise : Is it workstation or mail server ?



Press Enter to continue ...

"

dialog --backtitle "$bak" --title "$tit" --msgbox "$msg"  15 70

cd /usr/lib/sendmail-cf/cf
ksh /root/fim/code/server/s4.4.1
ksh /root/fim/code/server/s4.4.2.1
ksh /root/fim/code/server/s4.4.2.2
ksh /root/fim/code/server/s4.4.3
ksh /root/fim/code/server/s4.4.4

m4 redhat.mc > /etc/sendmail.cf
/etc/rc.d/init.d/sendmail stop
/etc/rc.d/init.d/sendmail start
```

2.1.6.5 Turn off SMTP vrfy and expn commands

```
#!/bin/ksh
#s4.4.1: Turn off SMTP vrfy and expn commands

bak="*** SERVER ***"
tit="Securing Eelctronic MAIL"

text="

The \"vrfy\" SMTP command allows a remote usre to verify the E -mail
address of a local user on the server.
The \"expn\" SMTP command expands aliases and mailing list.


Do you want to Turn off SMTP vrfy and expn commands?

For better security, we strongly recommend YES"

dialog --backtitle "$bak" --title "$tit" --yesno "$text" 15 70

if [ $? = 0 ] ; then
   echo "define(\`confPRIVACY_FLAGS', \`goaway')" >> redhat.mc
fi
```

2.1.6.6 Conteol Mail Relay Services "Check that the access database is active"

```
#!\bin\ksh
#s4.4.2.1: Conteol Mail Relay Services
#           "Check that the access database is active"

bak="*** SERVER ****"
tit="Securing Electronic MAIL"
txt="
\"sendmail\" can use an access database to control mail clients who can
relay mail. To activate this feature, choose YES"
For better security, we recommend YES ."

msg="
The access database already active.



Press Enter to continue ..."

dialog --backtitle "$bak" --title "$tit" --yesno "$txt" 12 70
if [ $? = 0 ] ; then
   x=`grep Kaccess /etc/sendmail.cf |wc -w`
   if [ $x = 0 ] ; then

      echo "FEATURE(\`access_db')" >> redhat.mc
   else
      dialog --backtitle "$bak" --title "$tit" --msgbox "$msg" 10 70
   fi
fi
```

2.1.6.7 Control Mail Relay Server "Domains allowed to relay"

```
#!/bin/ksh
#s4.4.2.2: Control Mail Relay Server
#           "Domains allowed to relay"

bak="*** SERVER ***"
tit="Securing Electronic MAIL"
txt="
Now, we set access for domains allowed to relay.

In access database; only hosts or domains with the RELAY action
are allowed to use the mail server as a mail relay.


Press Enter to continue ..."

dialog --backtitle "$bak" --title "$tit" --msgbox "$txt" 12 70

for i in `awk '$2=="RELAY"{print $1}' /etc/mail/access` ; do
msg="

Do you want to RELAY for $i (domain/IP)?"
dialog --backtitle "$bak" --title "$tit" --yesno "$msg" 12 70

    if [ $? = 1 ] ; then
       cat /etc/mail/access | grep  -v $i > /etc/mail/access
    fi
done
msg="

Do you want to insert(domain or IP) to access database?"

dialog --backtitle "$bak" --title "$tit" --yesno "$msg" 10 70
while [ $? = 0 ] ; do
      export IPbak=$bak
      export IPtit=$tit
      export IPmesg="
      Enter domain or IP to be added to access database :

      "
ksh /root/fim/code/get-IP/INPUT-IP
ip=`cat /root/fim/code/get-IP/IPnumber`
echo "$ip         RELAY" >> /etc/mail/access
msg="

Do you want to insert more (domains or IPs) to access database"

dialog --backtitle "$bak" --title "$tit" --yesno "$msg" 10 70

done
```

2.1.6.8 Set Domain Name Masquerading

```
#!/bin/ksh
#s4.4.3:Set Domain Name Masquerading

bak="*** SERVER ***"
tit="Securing Electronic MAIL"
txt="
```

```
The mail clients can be set so that the uniform mail address
is used on all out-bound mail, but managing all the mail clients
in even a small group can be tedious. Sendmail can be configured
to rewrite the headers of all out-bound mail so that they
masquerade as the central mail server.


Enter your domain name masquerading :
"
dialog --backtitle "$bak" --title "$tit" --inputbox "$txt" 17 70 2>dn

ans=`cat dn`

echo "MASQUERADE_AS(\`$ans')" >> redhat.mc
echo "FEATURE(masquerade_entire_domain)" >> redhat.mc
echo "FEATURE(allmasquerade)" >> redhat.mc
echo "FEATURE(masquerade_envelope)" >> redhat.mc
rm -f dn
```

2.1.6.9 Secure the POP and IMAP daemons

```
#!/bin/ksh
#s4.4.4: Secure the POP and IMAP daemons

bak="*** SERVER ***"
tit="Securing Electronic MAIL"
txt="
For mail servers that collect all incoming mail for anorganization,
a common means to deliver that mail to clients is for them to
retrieve the mail using the Post Office Protocol (POP) or the
Internet Message Access Protocol (IMAP).
POP is the older and simpler of the two protcols,providing basic
commands for authentication,retrieval and deletion of mail message
from the mail server.
IMAP is more flexible and supports creating,deleting,and renaming
mail folders (mailboxes),searching,selective retrieval of message
attributes and more.

Do you want to limit access to POP and IMAP ?
(Remark : The limiting access  is very important for security issue)

For better security, we recommend YES .
"

dialog --backtitle "$bak" --title "$tit" --yesno "$txt" 20 72
if [ $? = 0 ] ; then
   txt="
Enter domain of mail hub that holds & deivers mail to clients :
"
dialog --backtitle "$bak" --title "$tit" --inputbox "$txt" 10 70 2>domain
dm=`cat domain`
export IPbak=$bak
export IPtit=$tit
export IPmesg="
Enter IP of your mail server network:
"
ksh /root/fim/code/get-IP/INPUT-IP
net=`cat /root/fim/code/get-IP/IPnumber`
x=`grep ALL /etc/hosts.deny | wc -w `
if [ $x = 0 ] ; then
```

```
      echo "ALL: ALL" >> /etc/hosts.deny
fi
echo "ipop3d:   .$dm $net." >> /etc/hosts.allow
echo "imapd:    .$dm $net." >> /etc/hosts.allow
rm -f network
rm -f domain
fi
```

2.1.7 NFS Service

2.1.7.1 Turn off NFS Server

```
#!/bin/ksh
#s3.7.1: Turn off NFS Server

x="We will turn off NFS exports and remove NFS daemons"
if [ $( ls /etc/rc.d/init.d/nfslock | wc  -w ) -gt 0 ]; then
/etc/rc.d/init.d/nfslock  stop
fi
```

2.1.7.2 Set access to RPC Se rvices

```
#!/bin/ksh
#s4.6.1: Set access to RPC Services

bak="*** SERVER ***"
tit="Securing NFS"
txt="
Selecting \"YES\" will allow NFS access for this IP host.

selecting \"NO\" will deny it access to NFS service."

echo > fff
echo > /root/fim/code/get-IP/IPnumber

x1=`grep "portmap" /etc/hosts.allow|cut  -f2 -d:|tr -s [:space:]`
for i in $x1 ;do
      msg="
      Do you want to allow this IP : $i ?"

      dialog --backtitle "$bak" --title "$tit" --yesno "$msg$txt" 15 70
      if [ $? = 0 ]; then
            j=`cat fff`
            echo $j $i >> fff
      fi
done
y=`cat fff`
export IPbak=$bak
export IPtit=$tit
export IPmesg="
Secure Network File System (NFS) by set access to NFS services.
This is done by control access throhgh /etc/hosts.allow.
When user's IP is added to this file; this allow him to use
NFS services, else , he will be denied."

ksh /root/fim/code/get-IP/INPUT-IP
x=`cat /root/fim/code/get-IP/IPnumber`
z1=`grep portmap /etc/hosts.allow`
if [ $? = 1 ]; then
```

```
      echo portmap : $x >> /etc/hosts.allow
else
      z2=`echo portmap : $y $x`
      sed s/"$z1"/"$z2"/ /etc/hosts.allow > ff
      cp -f ff /etc/hosts.allow

fi

rm -f f
rm -f ff
rm -f fff
```

2.1.8 HTTP Service

2.1.8.1 Turn off HTTP Server

```
#!/bin/ksh
#s3.8.1: Turn off HTTP Server

      /etc/rc.d/init.d/httpd stop
      rpm -e apache
      rpm -e apache-devel
```

2.1.8.2 Limit HTTP Access to Localhost Only

```
#!/bin/ksh
#s3.8.2: Limit HTTP Access to Localhost Only

# ================================================
# This step for apache HTTP daemon can be bound to
#      a specific network interface.
# ================================================

echo Enter the interface IP
read ip
echo Enter the port to connect an outside network
read port

# ================================================
cp /etc/httpd/conf/httpd.conf httpd.fim
sed -e '/VirtualHost:/ i\' -e "Listen $ip:$port" \
/etc/httpd/conf/httpd.conf > httpd.conf
# ================================================

# You must remove httpd.conf after this step
mv -f httpd.conf /etc/httpd/conf/httpd.conf
# ================================================ =

/etc/rc.d/init.d/httpd restart
```

2.1.8.3 Set basic access to default deny

```
#!/bin/ksh
#s4.10.1: Set basic access to default deny

bak="*** SERVER ***"
tit="Securing HTTP Server"
```

```
txt="
We will make the access to all directories and fiels on the
server is \"denied\".


Press \"Enter\" to continue ..."
dialog --backtitle "$bak" --title "$tit" --msgbox "$txt" 10 70

access="/etc/httpd/conf/access.conf"
j=`grep -n -E "^<Directory />" $access|cut  -f1 -d:`
head -n$j $access > file0
echo "Options None" >> file0
echo "AllowOverride None" >> file0
echo "order deny,allow" >> file0
echo "deny from all" >> file0
((found=0))
for i in `grep -n "^</Directory>" $access|cut  -f1 -d:`; do
  if [ $i > $j ] && [ $found = 0 ]; then
     t=$i
     found=1
  fi
done
((t=`wc -l $access|awk '{print $1}'`-$t+1))
tail -n$t $access >> file0
cp -f file0 $access

rm -f file0
rm -f file1
rm -f file2
rm -f file3
rm -f 20
rm -f f2
rm -f f
```

2.1.8.4 Selectively open access to specific directories

```
#!/bin/ksh
#s4.10.2: Selectively open access to sp ecific directories

access="/etc/httpd/conf/access.conf"
dir=`grep -n "<Directory /" $access|grep  -v "<Directory />"|cut  -f2 -d:`
j=`grep -n "<Directory /" $access|grep  -v "<Directory />"|cut  -f1 -d:`
((counter=0))
((incr=0))
dir=`echo $dir|sed s/"Director y "/"Directory"/g`
intro=`cat introf`
back="*** SERVER ***"
title="Securing HTTP service"
dialog --backtitle "$back" --title "$title"  --textbox\
 /root/fim/code/server/s4.10.introf 20 70
for k in $dir; do
      k=`echo $k|sed s/"Directory"/"Directory "/g`
      #echo $i >> debugi #debug line

# The following block of code does the dialog box. dont mess with it.
      ((counter=$counter+1))
      #echo $counter #debug line
      d=`echo $dir|cut  -f$counter -d">"`
      d=$d">"
      export IPbak=$back
```

```
      export IPtit=$title
      export IPmesg="
Choose the IP numbers of the hosts or networks that are
allowed to access $k."
      ksh /root/fim/code/get-IP/INPUT-IP
      x1=`cat /root/fim/code/get-IP/IPnumber`
      checkmsg="
Please choose the appropriate options for
$k.

We recommend strongly that you do not  choose an option
you don't think is absolutely necessary.

"
      msg1="Execute CGI scripts"
      msg2="Follow Symbloic Links"
      msg3="Server Side Includes"
      msg4="IncludeNOEXEC"
      msg5="Indexes"
dialog --backtitle "$back" --title "$title" --checklist "$checkmsg" 18 70 5
1 "$msg1" off 2 "$msg2" off 3 "$msg3" off 4 "$msg4" off 5 "$msg5" off 2>f2


# The following code does the changes in the config file. bugs happen here
      i=`grep -n "$k" $access|cut -f1 -d:`
      head -n$i $access > file0
      cat f2|sed s/\"//g > f2
      options=''
      for i2 in `cat f2`; do
      case $i2 in
          1 ) options="$options ExecCGI" ;;
          2 ) options="$options FollowSymLinks" ;;
          3 ) options="$options Includes" ;;
          4 ) options="$options IncludesNOEXEC" ;;
          5 ) options="$options Indexes" ;;
      esac
      done
      if [[ $optionsA -eq A ]]; then options='None'
      fi
      echo "Options $options" >> file0
      echo "AllowOverride None" >> file0
      echo "order deny,allow" >> file0
      echo "deny from all" >> file0
      if [[ $x1 -eq '' ]]; then echo "allow from None" >> file0
      else echo "allow from $x1" >> file0
      fi
      ((found=0))
      for z in `grep -n "^</Directory>" $access|cut -f1 -d:`; do
          if [ $z -gt $i ]&&[ $found = 0 ]; then
                t=$z
                found=1
          fi
      done
      ((t=`wc -l $access|awk '{print $1}'`-$t+1))
      tail -n$t $access >> file0
      ((incr+=5))
      cp -f file0 $access
done

rm -f file0
rm -f file1
```

```
rm -f file2
rm -f file3
rm -f f
rm -f f2
rm -f 20
```

2.1.8.5 Help file for HTTP

```
#s.10.introf: Help file for HTTP

---------------------------------------------------------------
Use "UP/DOWN arrows" or "Space" buttons for continue reading.
---------------------------------------------------------------


Now, we are going to secure your http service. We will:

1. Limit the service to only those whome you trust.
2. Make appropriate options in the http directories.

Those whome you trust might be individual clients, i.e., specific IP
numbers, or complete networks, e.g., your internal network.

The options available are:

-ExecCGI: This option means that users can run CGI scripts in the
      specified directory. This option should only be allowed for CGI
      directories.

-FollowSymLinks: If users have write access to the HTML directories, they
       can set symbolic links to areas that contain sensitive data.

-Icludes: Server side includes can be used to bypass default file access
      restrictions.

-IncludesNOEXEC: Safer version of Includes that disables the #exec
      statement and #include of CGI scripts.

-Indexes: The daemon will print a directory listing for any directory
      without an index file (index.html). This may expose the names of data
      files normally hidden.
```

2.1.9 FTP Service

2.1.9.1 Remove Anonymous FTP Server

```
#!/bin/ksh
#s3.9.1: Remove Anonymous FTP Server

sed -e '/class   all   real,guest,anonymous   */c \' \
-e 'class   all   real   *' /etc/ftpaccess > file0
cp -f file0 /etc/ftpaccess

rpm -e anonftp

rm -f file0
```

2.1.9.2 Limit access with TCP Wrapper

```
#!/bin/ksh
#s4.9.1: Limit access with TCP Wrapper

## NOTES ##
# (1) for non anonymous; we call input-IP script (it is not complete yet)

bak="*** SERVER ***"
tit="Securing FTP"
txt="
We make the FTP server provide data to :
(1) a limited set of machines; or
(2) a general anonymous, so will be accessible to the world.

Please select one of the following options:

For BETTER security; select: (Set of addresses).
"
dialog --backtitle "$bak" --title "$tit" --menu "$txt" 18 70 2 1\
"Set of addresses" 2 "All users" 2>f
if [ $? = 0 ]; then
      x=`cat f`
      if [ $x = 1 ]; then
            export IPbak=$bak
            export IPtit=$tit
            export IPmesg="
Please enter the domain (IP no.) of the user who can
use FTP server. "
            ksh /root/fim/code/get-IP/INPUT-IP
            ip=$(cat /home/fim/IPnumber)
            y="in.ftp : $ip"
      else
            y="in.ftp : ALL"
      fi
      echo $y >> /etc/hosts.allow
fi

rm -f f
rm -f file-write
```

2.1.9.3 Limit permitted operations

```
#!/bin/ksh
#s4.9.2: Limit permitted operations

bak="*** SERVER ***  "
tit="Securing FTP"
txt="
We will prevent ananomyous users from modifying the contnts of
a writable directory; that is done regardless of the directory
permissions.

Press Enter to continue ..."
dialog --backtitle "$bak" --title "$tit" --msgbox "$txt" 10 70
set="chmod delete overwrite rename"

for i in $set; do
      x=$(grep "$i" /etc/ftpaccess)
      y=""$i"           no    guest,ananymous"
```

```
      sed s/"$x"/"$y"/ /etc/ftpaccess > file1
        cp  -f file1 /etc/ftpaccess
done
rm -f file1
```

2.1.9.4 Protect incoming "directory"

```
#!/bin/ksh
#s4.9.3: Protect incoming "directory"

bak="*** SERVER ***  "
tit="Securing FTP"
txt="
It is never good idea for security issue to allow write access to
an anonymous FTP directory.
So; We will prevent this service by stop the "incoming" directory; by
delete the file (/home/ftp/incoming).


Press Enter to continue ..."


dialog --backtitle "$bak" --title "$tit" --msgbox "$txt" 12 75
set="chmod delete overwrite rename"

rm -f /home/ftp/incoming
```

2.1.10 SSH Service

2.1.10.1 Start the SSH daemon

```
#!/bin/ksh
#s4.2.2: Start the SSH daemon

#-------------------------------------------------------------------------
### This script for install secure shell (SSH) for remote access.
### the assumptions are :
### First: ensure if SSH is downloaded or not. If it  is not;then the SSH
###        is downloaded manually according steps in sshssh text file that
###        is displayed through the script.
### Second: when we ensure that SSH is downloaded,we must check if it is
###         on or off. IF it is not work ;th en we apply and make it on.
#-------------------------------------------------------------------------
bak="*** SERVER ***"
tit="Setting up SSH"
txt1="
Secure Shell (SSH) provides strong authentcation and
transparent encryption of network communication fo r
login and file copying operations



Press Enter to continue ..."
txt2="
you don't have SSH package; you must download it...



Press Enter to continue ..."
```

```
txt3="
                ******* DOWNLOADING SSH manually *******

ssh is available for download from ftp://ftp.cs.hut.fi/pub/ssh/
after download you have to apply the following commands :
1- tar xzf ssh-vesion.tar.gz
2- cd ssh-version
3- ./configure --with-libwrap --without-rsh --disable-suid-ssh
 --disable-scp-stats
4- make
5- make install



Press Enter to continue ..."

dialog --backtitle "$bak" --title "$tit" --msgbox "$txt1" 12 70
if [ $? = 0 ]; then
      x=`ls /etc/rc.d/init.d/ | grep -q "sshd" `
      if [ $? = 0 ];then
            #echo "you already download";echo
            if [ $(lsof -i -Fc|grep '^c'|cut -b2-20|sort -u|grep -q
"ssh") ]; then
                  echo "you have already"
                  else
                        chkconfig sshd on
                        /etc/rc.d/init.d/sshd start
            fi
      else
      dialog --backtitle "$bak" --title "$tit" --msgbox "$txt2" 10 70
      dialog --backtitle "$bak" --title "$tit" --msgbox "$txt3" 19 70
      fi
fi
```

2.1.10.2 Replace "r" Programs with SSH

```
#!/bin/ksh
#s4.2.6: Replace "r" Programs with SSH

# *1* Comment all "r" clinets at server daemon
#-------------------------------------------

x="in.rlogin in.rshd in.rexecd"
for i in $x; do
        y=`grep $i /etc/inetd.conf|awk '{print $1}'`
        sed s/$y/#$y/ /etc/inetd.conf > hhh
        cp -f hhh /etc/inetd.conf
done

# *2* Remove the 'rsh' package; and then do the necessary 'linking'
#-----------------------------------------------------------------
# for redirectionyou can use: 1>f 2>f ;; 2>f 1>f ;; >f 2>&1

rpm -e rsh 1> /dev/null 2> /dev/null
ln -s /usr/local/bin/ssh /usr/bin/rsh 1> /dev/null 2> /dev/null
ln -s /usr/local/bin/slogin /us r/bin/rlogin 1> /dev/null 2> /dev/null
ln -s /usr/local/bin/scp /usr/bin/rcp 1> /dev/null 2> /dev/null

rm -f hhh
rm -f f
```

2.1.11 Printing Service

2.1.11.1 List allowed remote hosts

```
#!/bin/ksh
#s4.5.1: List allowed remote hosts

# *1** the file /etc/hosts.lpd is not  already exist; I created it.
# *2** we execute the first substep only.
# *3** we do not need to do substep 4.5.2;;; i.e :
#      we do not need to replace berkeley lpr/lpd with LPRng. the first is
#      enough.

export IPbak="*** SERVER ***"
export IPtit="Securing PRINTING SERVICES"
export IPmesg="
For more security; we control the using of printing services.
This is done by putting the names(IPs) of hosts allowed to use
the print server in /etc/hosts.lpd
Please Enter the IP\(s\) of Hosts Allowed to Use The Printer..."

ksh /root/fim/code/get-IP/INPUT-IP
x=`cat /root/fim/code/get-IP/IPnumber`
echo $x >>/etc/hosts.lpd
rm -f f
```

2.1.12 Samba Service

2.1.12.1 Get the latest version of samba

```
#!/bin/ksh
#s4.7.1: Get the latest version of samba

bak="*** SERVER ***"
tit="Securing SAMBA SERVER"
txt="
You have version for Samba less than 2.0.5a.
Please,update your package for Samba to correct security
problems, then complete Install.



Press Enter to continue ..."

rpm -q samba > rpm
cat rpm|cut -f2 -d"-" > version
x1=`cut -f1 -d"." version`
x2=`cut -f2 -d"." version`
x3=`cut -f3 -d"." version|cut -c1`
if [[ $x1 < 2 ]] ; then
   dialog --backtitle "$bak" --title "$tit" --msgbox "$txt" 12 70

   elif [[ $x1 = 2 && $x2 = 0 && $x3 < 5 ]] ; then
         dialog --backtitle "$bak" --title "$tit" --msgbox "$txt" 12 70
fi
rm -f rpm
rm -f version
```

2.1.13 Central Logging

2.1.13.1 Configure sysloged to accept remote log message

```
#!/bin/ksh
#s4.8.1: Configure sysloged to accept remote log message

### The restarting configuring is done by only *r oot*
# the most time in this script is consumed in: how do the subsitution?
# 1- find the line number.{grep -n "xxx" file |cut .... }
# 2- begin the search of intended word (daemon syslogd) only after word
#     (start);then and at this procedure we can ex change.

bak="*** SERVER ***"
tit="Securing SYSLOG"
txt="
By this step, we will configure syslogd to accept remote
log message.


Press Enter to continue ..."

dialog --backtitle "$bak" --title "$tit" --msgbox "$txt" 10 70
#---> First : add "-r" option to the line that starts
#             the syslog daemon: /etc/rc.d/init.d/syslog
z="start)"
y="daemon syslogd"
grep "$y" /etc/rc.d/init.d/syslog | grep  -q "r"
if [ $? = 1 ]; then
      x=$(grep -n "$z" /etc/rc.d/init.d/syslog|cut  -f1 -d":")
      sed "$x~1"s/"$y"/"$y  -r"/ /etc/rc.d/init.d/syslog > file1
      chmod 755 file1
      cp -f file1 /etc/rc.d/init.d/syslog
fi
#---> Second : restart the syslog service ;; this is done by
#              the root only.

/etc/rc.d/init.d/syslog restart
rm  -f file1
```

2.1.13.2 Configure Log Rotat ion

```
#!/bin/ksh
#s4.8.2: Configure Log Rotation

# the most time in this script is consumed in: how do the subsitution?
# The best way is looking for "daily" or "weekly" in the begining line;
# then subsituting it by "monthly".
#-----------------------this the first method--------------------------
# Our objective is to change the "weekly" -in current configuration -
#   to "monthly"-in new configuration - .
#   the way to our search target at /etc/logrotate.conf is
#   the previous commnt: "# rotate log fil es ...".
# when find this commnt; we change the next line of it.
#-------------------------------------------------------------------------

bak="*** SERVER ***"
tit="Securing SYSLOG"
```

```
txt="
\"logrotate\" program in Linux system is designed to rotate,preser ve
and delete log fiels after a certain period of time, or when the
fiels reach a certain size.
But; for a loghost, log rotation should be turned off.

Press Enter to continue ..."

dialog --backtitle "$bak" --title "$tit" --msgbox "$txt" 11 72
rm -f /etc/cron.daily/logrotate
```

2.1.14 Input-IP

2.1.14.1 Script for Get IP Address

```
#!/bin/ksh
#INPUT-IP: Script for Get IP Address

echo > /root/fim/code/get-IP/IPnumber
input=0 #--> the value by it determine continuo or exit from the main loop

txt="

Enter IP numbers, separated by space(s)
      the standard IP is in the form  : xxx.xxx.xxx.xxx
      each field is in the range : 0  -- 255"

init=""
echo > file-write #-> This file will contain the correctd ent ered IP(s)
# ============================
#--> This loop for input IP; It finish when the IP or the sequence
#      of IPs is entered correctly
# ============================
while [ $input = 0 ];do
      dialog --backtitle "$IPbak" --title "$IPtit" --inputbox "$IPmesg\
          $txt"  15 70 "$init" 2>file-ip
      #--> This if statement for exit from the script when you press
        #      cancel ay any time
      if [[ $? = 1 ]];then
            echo;cancel=0
            input=1
      #--> else you do not select cancel; which means you  want continuo
        #      active the script
      else
            cat file-ip|tr -s [:space:]>file-ip
            x=$(cat file-ip | wc -w)
            i=1
            echo > file-wrong  #--> All these three files will contain
            echo > file-wrong1 # the wrong entered IP(s);each one will
            echo > file-wrong2 # be used in dialog boxes
            while [ $i -le $x ]; do
                  y=`cut -f$i -d' ' file-ip`
                  #ksh IP-chk1-fim $y
                  # The next commands interested in check the IP;
                  #  the checking IP is done by gerp command in only
                        #  one l ine. (this is great, is'nt that?)

                  echo $y|grep -q "\.$"
                  if  [[ $? != 0 ]]
                  then    #echo "must add dot"
```

```
                        xg=$y.
                  else    xg=$y
                  fi
                  echo $xg|grep -E -x -q "([0-9]\.|[0-9][0-9]\.|[0-1][0-
9][0-9]\.|2[0-4][0-9]\.|25[0-5]\.){1,4}"
                  if [[ $? != 0 ]]; then
                        echo -n " $y" >> file-wrong
                        echo -n "$i," >> file-wrong1
                        echo -n " #$i: ($y) ;" >> file-wrong2
                  else
                        echo -n " $y" >> file-write
                  fi
                  ((i+=1))
            done
            if [ $(cat file-wrong1|wc -w) -gt 0 ]; then
                  ### create init value for input dialog
                  # ====================================
                  w=$(cat file-wrong)
                  init=$(echo $w)
                  #** create title text for dialog input
                  # ====================================
                  m1=$(cat file-wrong1)
                  m2=$(echo $m1)
                  title="There is wrong in IP(s):$m2"
                  # ==================================
                  infotxt="
                                    There is wrong Mr."
                  dialog --title "MISTAKE" --infobox "$infotxt" 7 30
                  sleep 1s
                  # ** create text for msgbox;(the spaces area for
                        #    nice viewing
                  # ==========================================
                  m3=$(cat file-wrong2)
                  msgtxt="There is wrong in the following IP(s):

                                    $m3"
                  dialog --msgbox "$msgtxt" 15 60
            else
                  input=1
            fi
      fi
done
if [[ $cancel != 0 ]]; then
      IPIP=$(cat file-write)
      echo $IPIP > /root/fim/code/get-IP/IPnumber
      echo
fi

# Finaly; delete the temporary files of the script

rm -f file-ip
rm -f file-write
rm -f file-wrong
rm -f file-wrong1
rm -f file-wrong2
```

Table of Contents