

Securing Your Network

Introduction: What to do?

You want to operate your network without any unauthorized access, or any other security problems. Basically these measures are necessary to achieve security.

- Secure your hosts and networks.
- Keep up to date on the latest security threats.

In securing your hosts and networks, you strive to let nobody gain unauthorized access to your hosts and networks, e.g. somebody log in using someone else's account. Because nothing is absolutely secure, you also have to keep up to date to the latest security threats. This way you know whether your hosts and networks are vulnerable or not, and if so, you can immediately take the necessary measures to close the vulnerability by, for example, applying patches to your software or enforcing a firewall rule.

In this tutorial you will find practical measures to secure your hosts and networks. You will also find how Network Address Translation (NAT) may pose security threats and how you can reduce the threats.

The first step to achieve network security is to have a security policy. The AI³ network has such policy, and each SOI-ASIA partner, as part of the AI³ network, has to follow the policy.

Securing Your Hosts

Account security

Below are several practical measures to secure your hosts via account security:

- Eliminate group and shared accounts.
An account should be used by a single person.
- Use "good" password.
- Use password ageing.
- Limit users who can gain access to `root` user.
These users are those in group `wheel` and those who can do `sudo`.

- Do not log on using `root` user.
Log on using your account then do `su`. This way, it is easy to track who gained root access, by examining the system log file.
- Do not leave idle console unattended.

Network servers

You should confirm that network servers running on your hosts are secure. To ensure this, the best policy is not to run a network server unless it is necessary and it doesn't have a security vulnerability. Two commands that can be used to check which network servers are running are:

- `netstat -na` shows the active Internet connections.
Below is an example of the results.

```

Proto Recv-Q Send-Q Local Address           Foreign Address         (state)
tcp6      0      20 2001:200:0:8802::22    2001:d30:101:1:2.2139  ESTABLISHED
tcp4      0      0 *.2601                 *.*                     LISTEN
tcp46     0      0 *.2601                 *.*                     LISTEN
tcp4      0      0 127.0.0.1.25          *.*                     LISTEN
tcp4      0      0 *.22                   *.*                     LISTEN
tcp46     0      0 *.22                   *.*                     LISTEN
udp4      0      0 *.514                  *.*                     *
udp6      0      0 *.514                  *.*                     *
udp4      0      0 *.68                   *.*                     *
icm6     8184   0   *.*                    *.*                     *
icm6      0      0 *.*                    *.*                     *

```

The `Proto` column shows the connection protocol, e.g. `tcp6` is TCP for IPv6. Local and Foreign addresses columns are in *address.port* format. Pay attention to the ones whose Foreign Address is `*.*`; these are entries of local ports opened network servers.

- `sockstat -46` shows the processes that have Internet connections.
An example is shown below. You can see for example that `root` is running `zebra` that is listening on TCP port 2601.

```

USER      COMMAND  PID  FD PROTO  LOCAL ADDRESS  FOREIGN ADDRESS
root      zebra    213  8 tcp4    *:2601         *.*
root      sendmail 173  3 tcp4    127.0.0.1:25  *.*
root      sshd     168  4 tcp4    *:22           *.*
root      syslogd  159  5 udp4    *:514         *.*
root      dhclient  68   4 udp4    *:68           *.*

USER      COMMAND  PID  FD PROTO  LOCAL ADDRESS  FOREIGN ADDRESS
husni     sshd     10073 5 tcp6    2001:200:0:8802:2d0:9 2001:d30:101:1:290:99
root      sshd     10071 5 tcp6    2001:200:0:8802:2d0:9 2001:d30:101:1:290:99
root      zebra    213  5 icm6    *.*           *.*
root      zebra    213  7 tcp46   *:2601        *.*
root      sshd     168  3 tcp46   *:22          *.*
root      syslogd  159  4 udp6    *:514         *.*
root      rtadvd   144  3 icm6    *.*           *.*

```

You can then decide whether to kill the network servers based on the results of both utilities.

Network servers are usually run at boot time by `inetd` and local startup scripts at `/usr/local/etc/rc.d` directory. To minimize network servers:

- Disable `inetd`. Edit `/etc/rc.conf` and add the below line.

```
inetd_enable="NO"
```
- Disable unnecessary local startup scripts.
 For example you have a startup script `netserver.sh` for a network server. Disable it by running

```
chmod -x netserver.sh
```

Firewall

Firewall can secure your networks by blocking packets. For example, you want to give access to your web server only from your your networks. You can do it by giving Access Control List to your web server, or you filter packets to your web server using firewall.

You have to enable `IPFIREWALL` kernel options to active firewall capability on FreeBSD. The procedure is

1. As `root`, edit the startup configuration `/etc/rc.conf` and add the following lines to use firewall with open access.

```
firewall_enable="YES"
firewall_type="open"
```

2. Create a new kernel configuration file.
 For example, create it by copying the `GENERIC` kernel and make new kernel configuration `MYFIREWALL`.

```
cd /usr/src/sys/i386/conf
cp GENERIC MYFIREWALL
```

3. Add kernel options to activate firewall.
 Edit `MYFIREWALL`, add the following line.

```
options IPFIREWALL
```

4. Compile and install the kernel.

```
config MYFIREWALL
cd ../../compile/MYFIREWALL
make depend
make
make install
```

5. Reboot FreeBSD to use the new kernel.
6. After reboot, as `root` type the following command to see the firewall rules.

```
ipfw list
```

The rules should be

```
00100 allow ip from any to any via lo0
00200 deny ip from any to 127.0.0.0/8
00300 deny ip from 127.0.0.0/8 to any
65000 allow ip from any to any
65535 deny ip from any to any
```

Packet filtering using FIREWALL are done by traversing the firewall rules from the lowest to the highest number. A packet is processed according to the first rule that matches the packet. We give several examples of commands to add and delete firewall rules. You have to execute `ipfw` as `root`.

```
ipfw add 500 allow tcp from 10.0.0.0/8 to 10.1.1.1 80
ipfw add 501 deny tcp from any to 10.1.1.1 80
ipfw add 1000 allow icmp from 10.1.0.0/24 to 10.1.1.2
ipfw add 1001 deny icmp from any to 10.1.1.2
```

The above commands add entries to the firewall rule list. The rule number 500 and 501 state that only TCP packets from hosts with IP address from 10.0.0.0 to 10.255.255.255 are be allowed to access port 80 of 10.1.1.1. The rule number 1000 and 1001 allow ICMP packets to 10.1.1.2 only from 10.1.0.0/24. After adding these rules, the firewall rule list becomes.

```
00100 allow ip from any to any via lo0
00200 deny ip from any to 127.0.0.0/8
00300 deny ip from 127.0.0.0/8 to any
00500 allow tcp from 10.0.0.0/8 to 10.1.1.1 80
00501 deny tcp from any to 10.1.1.1 80
01000 allow icmp from 10.1.0.0/24 to 10.1.1.2
01001 deny icmp from any to 10.1.1.2
65000 allow ip from any to any
65535 deny ip from any to any
```

You can use `ipfw delete <ruлено>` command to delete a firewall rule.

Secure Shell

Secure shell (SSH) is a remote access service that uses encryption, thus it is secure from eavesdropping. FreeBSD 4.x, and many other Unix based OSes, uses SSH as the default remote access service. The commonly used implementation is OpenSSH (<http://www.openssh.org/>). From time to time OpenSSH upgrade its package for improvements or vulnerability patches. You should update your SSH package whenever a security problem is found on the OpenSSH version that are running in your host. Below is how to install OpenSSH on FreeBSD:

1. Download the latest version of OpenSSH.
You can download OpenSSH from <http://www.openssh.org/> or a mirror site. You should download the portable OpenSSH. The Instructor will give the URL of a local copy for use in this workshop.
2. Unpack the OpenSSH package. For OpenSSH 3.8.1p1

```
tar zxvpf openssh-3.8p1.tar.gz
cd openssh-3.8p1
```
3. Read file named INSTALL and check the requirement for OpenSSH.

```
less INSTALL
```

In general, you need a working installation of Zlib and OpenSSL with the version that is stated in INSTALL file.
4. Install the package. As root, do the following:

```
./configure
make
make install
```
5. Edit sshd configuration file.
Your SSH daemon is located in `/usr/local/sbin/sshd`, while the original one is located in `/usr/sbin/sshd`, so you need to edit your `/etc/rc.conf`. Insert the below lines to your `/etc/rc.conf` file.

```
sshd_program="/usr/local/sbin/sshd"
sshd_flags="-f /etc/sshd_config"
```
6. Reboot your host.
7. Confirm that the sshd process is running.

```
ps xa|grep sshd
```
8. Try to access your host via SSH.

```
ssh localhost
```

Network Address Translation

Network Address Translation (NAT) is a mechanism that changes IP address of packets. NAT is a way for networks using private IP addresses to communicate with the Internet.

Figure 1 shows how NAT works. The figure shows a network connected to the Internet via a NAT router. If a client (C1) behind the NAT router communicates with a server (S1) on the Internet, the NAT router changes the source IP address of a packet sent by C1 with its IP address (NR), then sends the packet to the server. Because of NAT, the sender only knows that packets from C1 come from the NAT router. When the server sends a packet back to the client, it sends the packet to the NAT router (NR). The NAT router knows that the packet has to be delivered to the client, thus it change the destination IP address of the

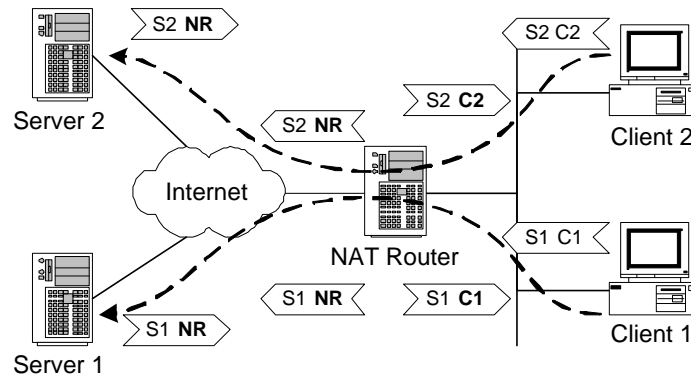


Figure 1: NAT mechanism

packet to C1 and sends it to the client. A NAT router has a NAT table whose entries are IP address mapping, so it knows how to process the IP packets it receives.

NAT provides anonymity to its client: it becomes the source IP address for all connections between its client and hosts on the Internet and it doesn't keep an IP address mapping log. Logging IP address mapping is possible, but not practical since each translated packet should be logged, so the log file size would be huge. With these situations, there are dangers that a NAT router may be used for abusive actions, e.g. credit card frauds. Therefore, the safest way to use NAT is to limit Internet access using NAT to clients that are secure. For example: the client users are logged.

AI³ Security Policy

1. Refuse anything except the ones defined.
2. Use firewall on border routers.
3. Share security information via ai3-ix@ai3.net
4. Run port scan and SSH version check regularly.
5. PC hosts must use SSH only.
6. Zebra beasts can be accessed from localhost only.
7. Limit access via NAT only to authorized clients.
8. Avoid NAT for web access, use Squid.

Exercise

Ex. 1 Account security

1. Check who logged in recently.
last

2. Check the following files to know who can access `root` via `su`.
`/etc/passwd`
`/etc/group`
List users who are in `wheel` group.
3. Find out who gained access to `root`
`cat /var/log/messages|grep "su:"`

Ex. 2 Network servers

1. Check Internet connections.
`netstat -na`
How many ports are opened?
2. Check processes that open ports.
`sockstat -46`
Which processes that open ports?

Ex. 3 Firewall

1. Add firewall capability to your kernel. Follow the procedure explained in the Firewall section.
2. Ask other participants to remote access your host via SSH.
What's the result?
3. Create a firewall rule to deny access to the SSH port of your host.
4. Ask other participants to remote access your host via SSH.
What's the result?
5. Delete the firewall rule you created before.

Ex. 4 Installing SSH

Do the procedure in the SSH section.