Network Management and Security[1]

HARDNESS : 8/10

University of Tehran
School of Electrical and Computer Engineering

دانشگاه تهران

دانشکده‌ی مهندسی برق و کامپیوتر

Computer Network Lab
آزمایشگاه شبکه‌های کامپیوتری

Professor:
Dr. Ahmad Khonsari
دکتر احمد خونساری
a_khonsari@ut.ac.ir

Amir Haji Ali Khamseh'i
امیر حاجی‌علی‌خمسه‌ء
khamse@ut.ac.ir

Reza Sharifnia
رضا شریف نیا
Reza.sharifnia@ut.ac.ir

Muhammad Borhani
محمد برهانی
borhani.m@ut.ac.ir

AmirAhmad Khordadi
امیراحمد خردادی
a.a.khordadi@ut.ac.ir

Sina Kashipazha
سینا کاشی‌پزها
sina_kashipazha@ut.ac.ir

Hadi Safari
هادی صفری
hadi.safari@ut.ac.ir

December 21, 2021

۳۰ آذر ۱۴۰۰

[1] S. Panwar, S. Mao, J.-dong Ryoo, and Y. Li, "Network management and security," in TCP/IP Essentials: A Lab-Based Approach, Cambridge: Cambridge University Press, 2004, pp. 187–213.

Objectives

- SNMP and MIBs, using NET-SNMP as an example, and using NETSNMP utilities to query MIB objects.
- Encryption, confidentiality, and authentication, including DES, AES, RSA, MD5 and DSS.
- Application layer security, using SSH and Kerberos as examples.
- Transport layer security, including SSL and the secure Apache server.
- Network layer security, IPsec and Virtual Private Networks.
- Firewalls and IPTABLES.
- Accounting, auditing, and intrusion detection.

Part I

SNMP Exercises

For exercises 1-7 in this laboratory manual, the network topology is given in Figure 1.3, where all the hosts are connected in a single network segment using their IP addresses, i.e. from 128.238.66.100 to 128.238.66.107.

Table 1: The IP addresses of the hosts (Table 1.2)

Host	IP Address	Subnet Mask
h0	128.238.66.100	255.255.255.0
h1	128.238.66.101	255.255.255.0
h2	128.238.66.102	255.255.255.0
h3	128.238.66.103	255.255.255.0
h4	128.238.66.104	255.255.255.0
h5	128.238.66.105	255.255.255.0
h6	128.238.66.106	255.255.255.0
h7	128.238.66.107	255.255.255.0

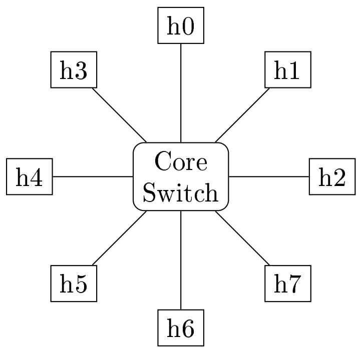


Figure 1: A single segment network (Figure 1.3)

1 SNMP Service and MIB Struct

Before the lab, you should do these instructions on *h_0's Console*:

1. Backup the original `snmpd` configuration file:

   h_0's Console
   ```
   mv /etc/snmp/snmpd.conf /etc/snmp/snmpd.conf.save
   ```

2. Create a simple configuration file `/etc/snmp/snmpd.conf` containing a single line defining a read-only community **guest**:

   h_0's Console
   ```
   nano /etc/snmp/snmpd.conf
   # after open the file
   rocommunity guest
   ```

Save the file using key combination `Ctrl + O` and then `Enter`, and exit from `nano` editor using `Ctrl + X`.

Try to stop and then start `snmpd`:

h_0's Console

```
service snmpd stop
service snmpd start
```

Run this command to check if `snmpd` is started:

h_0's Console

```
pgrep snmpd
```

Study the default configuration file you have backed up (`/etc/snmp/snmpd.conf.save`). This file is well commented. Read the comments and study the configuration options:

h_0's Console

```
nano /etc/snmp/snmpd.conf.save
```

Study the MIB (Management Information Base) files in the `/usr/share/snmp/mibs` directory. Examine the Interface MIB (`IF-MIB.txt`) and the TCP MIB (`TCP-MIB.txt`) files to see the MIB objects and data types:

h_0's Console

```
nano /usr/share/snmp/mibs/IF-MIB.txt
#  after study
nano /usr/share/snmp/mibs/TCP-MIB.txt
```

Save these two files for the lab report.

Note: You can use `Ctrl+W` to search for specific keyword in *nano* editor.

Report

1. What is the community name used in this lab? What is the use of the community name?
2. Looking at file `IF-MIB.txt`, What is the data type for the MIB object `ifMtu`? What is the definition of the MIB object `ifPhysAddress` and `ifInOctets`?
3. Looking at file `TCP-MIB.txt`, What is the data type and definition of `tcpRtoAlgorithm`? What values are allowed for `tcpRtoAlgorithm`? What is the definition of `tcpMaxConn`?

2 SNMP List Objects

Run this command to display the TCP MIB:

h_0's Console

```
snmpwalk -v 2c -c guest localhost tcp
```

Run this command to display the Interface MIB:

h_0's Console

```
snmpwalk -v 2c -c guest localhost interface
```

Compare the outputs with the MIB files you saved in the previous exercise. Also compare the outputs of the last command (Interface MIB) with the output of the following command:

h_0's Console

```
ifconfig -a
```

Note: If needed, you can run below command to find out the meanings of the options:

h_0's Console

```
man snmpwalk
```

Retry the `snmpwalk` commands, but change **guest** to **public**. Can you display the MIBs this time?

h_0's Console

```
snmpwalk -v 2c -c public localhost tcp
#  afer run above command
snmpwalk -v 2c -c public localhost interface
```

Report

1. What is the MTU of the *Ethernet* interface? What is the MTU of the *Loopback* interface? Justify your answer with the `snmpwalk` output and the `ifconfig` output.
2. Why did the `snmpwalk` command with a community name public fail?

3 SNMP Remote Access

Execute `wireshark` on *h0* to capture SNMP messages.

Run `telnet` from host *h1*.

Note: `netlab` is the *Login ID* and the *Password*.

h_1's Console

```
telnet 128.238.66.100
```

While telnet is running, try to run this command on *h0*:

h_0's Console

```
snmpwalk -v 2c -c guest localhost tcp
```

Try to run this on *h_2's Console*:

h_2's Console

```
snmpwalk -v 2c -c guest 128.238.66.100 tcp
```

Try to run below command on *h2* to get the MIB object `IF-MIB::ifMTU.1` from a remote machine:

h_2's Console

```
snmpget -v 2c -c guest 128.238.66.100 IF-MIB::ifMtu.1
```

Note: you can list all interfaces for the hosts by executing `snmpwalk -v 2c -c guest h0.netlab interface` command.

Save the `snmpget` output.

To terminate `telnet` which is running on *h1*, go to the same terminal which has `telnet` running, and use key combination `Ctrl + D` (which logs out the user from the current `telnet` session).[1]

[1] Another way to close `telnet` session is using `Ctrl + ]` and then entering `quit` (may not work on `docker` `term` nodes).

Use `wireshark` to analyze the format of the captured SNMP **Get** request and response messages. Print the messages for the lab report.

Report

1. What is the port number used by the SNMP agent?
2. What are the full text-based and numerical object ID's of the MIB object `interface.ifMTU.x` ($x \in \{2, 3, 4, \cdots\}$ that list in `snmpwalk ... interface`)? What was the value returned? Justify the answer using Figure 9.3 of reference book and the `ifconfig` output.
3. Draw the format of one of the captured SNMP request and response messages, including the name and value of each field.

Part II

Exercises on Secure Applications

4 Plain Transfer

Execute `wireshark` to capture packets between the *h0* machine and the *h1* machine.

Run this command on *h_1's Console*:

h_1's Console

```
ftp 128.238.66.100
```

When prompted, enter 1111 for the *Login ID*, and 2222 for the *Password*.

Then stop `wireshark` and exit `ftp` connection using:

h_1's Console

```
exit
```

Use `wireshark` to analyze and print the packets that carry the login ID and the password for the lab report.

Repeat the above experiment, but now using `telnet` *:*

Execute `wireshark` to capture packets between the *h0* machine and the *h1* machine.

Run this command on *h1*:

h_1's Console

```
telnet 128.238.66.100
```

When prompted, enter 1111 for the *Login ID*, and 2222 for the *Password*.

Use `wireshark` to analyze and print the packets that carry the login ID and the password for the lab report.

Report

1. Can you see the Login ID and the password in the `ftp` experiment? Submit the two packets you printed.
2. Can you see the Login ID and the password in the `telnet` experiment? Submit the packets you printed.
3. What is the difference between `ftp` and `telnet` in their transmission of user ID's and passwords? Which one is more secure?

5 Secure Transfer

Start `ssh` service on host *h0*:

h_0's Console

```
service ssh start
```

Execute `wireshark` on the *h0* host or the *h1* host to capture packets between two machine.

Run this command on *h1*:

h_1's Console

```
sftp 1111@128.238.66.100
```

When prompted, type `yes` to continue the connection and type `2222` for the *password*:

Use `wireshark` to analyze and print one or two `ssh` packets for the lab report.

Repeat the above experiment, but now using `ssh` *, that is;*

Execute `wireshark` on the *h0* host or the *h1* host to capture packets between two machine.

Run this command on *h1*:

h_1's Console

```
ssh 1111@128.238.66.100
```

When prompted, enter `2222` for the *password*.

Use `wireshark` to analyze and print one or two `ssh` packets for the lab report.

Report

1. In each experiment, can you extract the password from the captured packets? Can you read the IP, TCP, SSH headers? Can you read the SSH data?
2. What is the client protocol (and version) used in both cases?
3. What is the port number used by the `ssh` server? What is the port number used by the `sftp` server?

   Justify your answer using the `wireshark` output and the content of `/etc/services` file.

Part III

Exercises on Firewalls and `iptables`

In this exercise using two hosts. You can use Figure 1.3.

6 Firewall Basic

Run this command on *h0* to list the existing rules in the filter table:

h_0's Console

```
iptables -L -v
```

Save the output for the lab report.

Append this rule to the end of the `INPUT` chain on *h0*:

h_0's Console

```
iptables -A INPUT -v -p TCP --dport 23 -j DROP
```

Run `iptables -L -v` again on the *h0* and the *h1* to display the filter table. Save the output.

h_0's Console

```
iptables -L -v
```

Execute `wireshark` on *h0*. Then, run below command from another machine like *h1* and wait for it to complete (may take some minutes):

h_1's Console

```
telnet 128.238.66.100
```

Stop `wireshark` and save its output for the lab report.

Report

1. Can you `telnet` to the *h0* host from the other machine (e.g. from *h1*)?
2. From the `wireshark` output, how many retries did `telnet` make? Explain the **exponential backoff algorithm** of TCP timeout and retransmission.

7 Firewall Action

Delete the rule created in the last exercise on *h0*:

h_0's Console

```
iptables -D INPUT -v -p TCP --dport 23 -j DROP
```

Then, append this rule to the `INPUT` chain:

h_0's Console

```
iptables -A INPUT -v -p TCP --dport 23 -j REJECT --reject-with tcp-reset
```

Execute `iptables -L -v` to display the new rule.

h_0's Console

```
iptables -L -v
```

Restart `wireshark` on *h0*. Then, run below command from *h1*:

h_1's Console

```
telnet 128.238.66.100
```

Save the `wireshark` output for the lab report.

Report

1. Explain the difference between the `wireshark` outputs of this exercise and the previous exercise. How many attempts did TCP make this time?

Part IV

Exercises on Secure Apache Server

In the exercises in this section, using two hosts from *server* and *gui* hosts (also you can use Figure 3.server.gui). Run `apache2` service on *server*.

Server's Console

```
service apache2 start
```

8 Generate Certificate

Run this command to study the OpenSSL command line tool:

Server's Console

```
man openssl
```

Create a new private key and a new self-signed certificate for the Apache server, using:

Server's Console

```
cd /etc/apache2/ssl
openssl genrsa 4096 > server.key
openssl req -new -x509 -sha256 -days 365 -key server.key -out server.crt
```

Then you will be asked a number of questions, regarding the location, affiliation, etc. of the Apache server. Input these answers:

Server's Console

```
Country Name (2 letter code) [AU]:IR
State or Province Name (full name) [Some-State]:Tehran
Locality Name (eg, city) []:Tehran
Organization Name (eg, company) [Internet Widgits Pty Ltd]:University of Tehran
Organizational Unit Name (eg, section) []:ECE Department
Common Name (e.g.\ server FQDN or YOUR name) []:ece.ut.ac.ir
Email Address []:netlab@ut.ac.ir
```

After you type in the answers, a self-signed certificate is created at `/etc/apache2/ssl/server.crt`

Save the output for the lab report.

NOTE: (Not Need)You can generate the private key and the self-signed certificate using one command (instead of the two above):

Server's Console

```
cd /etc/apache2/ssl
openssl req -new -x509 -sha256 -days 365 -newkey rsa:4096 -nodes \
-keyout server.key -out server.crt \
-subj "/emailAddress=netlab@ut.ac.ir/C=IR/ST=Tehran/L=Tehran/O=University of Tehran/OU=ECE
    Department/CN=ece.ut.ac.ir"
```

Let's go over exactly what this means.

- `openssl` : This is the basic command line tool provided by OpenSSL to create and manage certificates, keys, signing requests, etc.

- `req` : This specifies a sub-command for X.509 certificate signing request (CSR) management. X.509 is a public key infrastructure standard that SSL adheres to for its key and certificate management. Since we are wanting to create a new X.509 certificate, this is what we want.
- `-x509` : This option specifies that we want to make a self-signed certificate file instead of generating a certificate request.
- `-sha256` : This option instructs the usage of SHA-256 cryptographic hash function.
- `-days 365` : This specifies that the certificate we are creating will be valid for one year.
- `-newkey rsa:4096` : This option will create the certificate request and a new private key at the same time. This is necessary since we didn't create a private key in advance. The `rsa:4096` tells OpenSSL to generate an RSA key that is 4096 bits long.
- `-nodes` : This option tells OpenSSL that we do not wish to secure our key file with a passphrase. Having a password protected key file would get in the way of Apache starting automatically as we would have to enter the password every time the service restarts.
- `-keyout server.key` : This parameter names the output file for the private key file that is being created.
- `-out server.crt` : This option names the output file for the certificate that we are generating.
- `-subj` : Sets subject name for new request or supersedes the subject name when processing a request.

9 HTTPS Handshake and Request

Restart Apache server to load the new key and the new certification:

Server's Console

```
service apache2 restart
```

Execute `wireshark` to capture the packets between the *gui* host and the *server* host.

Open GUI interface of the *client* by double-clicking on the *client* node in GNS3, or by right-clicking on the *client* and selecting `Console`. This will open a VNC connection to the *client* on your machine's web browser.

On the client, start *Mozilla Firefox* web browser and navigate to this URL:

gui's web browser

```
https://128.238.66.100
```

Click on *Advanced* button and then on *View Certificate* link. Then a *Certificate Viewer* window pops up, displaying detailed information about the received certificate. Examine the certificate and dump the window into a picture. Save the picture for the lab report.

Click the *Continue* button in the *Website Certified by an Unknown Authority* dialog window to accept the certificate. Then terminate `wireshark` and Mozilla.

Use `wireshark` to examine the operation of SSL.

Report

1. What is the port number used by the secure Apache server?
2. Compare the general information of the received certificate with the `openssl` output saved in the previous exercise. Are they consistent?
3. What is the Subject of the received certificate? Who is the Issuer of this certificate? Are they the same?
4. What is the *Certificate Signature Algorithm* used to generate and distribute this certificate?
5. When was the certificate signed? When will it expire?

Part V

** Exercises on Auditing and Intrusion Detection

After adding the Certificate to exceptions from the previous exercise, **browse several pages** in the *client's Firefox* by clicking the links. Also open random URL which does not exist on the web server, for example:

gui's web browser

```
https://128.238.66.100/thisurldoesnotexist
```

10 ** Trace Log

Go to `/var/log/apache2/` to examine the log files in the *server*:

Server's Console

```
cd /var/log/apache2/
```

If a log (e.g. the *Apache Access Log* at `/var/log/apache2/access.log`) is too long, use the command `grep keyword access.log | tail` to display last log entries containing the *keyword*; For example:

Server's Console

```
grep GET access.log | tail
```

Enter the keyword **failed** or **404** to display logged failures:

Server's Console

```
grep failed access.log | tail
grep 404 access.log | tail
```

See other log files.

11 ** Log Analyzer

Linux uses a utility called `webalizer` to analyze the web server log files. `webalizer` reads the `apache2` log files and creates a set of web reports on server statistics. Another utility is `goaccess` that analyzes various access type and lists them.

To analyze local log files using `webalizer`, execute:

Server's Console

```
webalizer
```

To view the reports of the `webalizer`, start *Mozilla Firefox* on the *client* and navigate to this URL:

gui's web browser

```
https://128.238.66.100/usage/index.html
```

Examine the web statistics displayed in the browser. Also click on the month links in the *Summary by Month* table to see the statistics of each month.

To use `goaccess`, you need to edit its configuration file (`/etc/goaccess.conf`):

Server's Console

```
nano /etc/goaccess.conf
```

Enable `NCSA Combined Log Format`. You can do this by making sure than only this log format directive is enabled (i.e. is not commented):

server:/etc/goaccess.conf

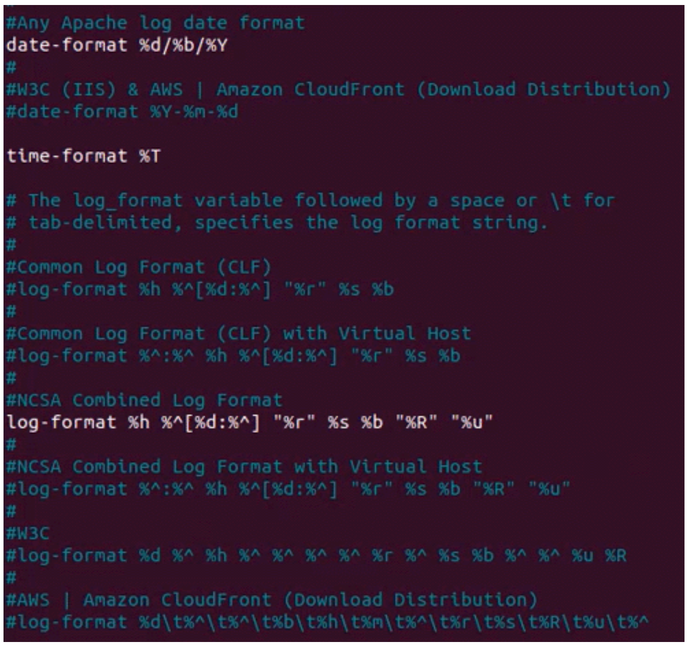
```
#NCSA Combined Log Format
log-format %h %^[%d:%^] "%r" %s %b "%R" "%u"
```

Also add this time format directive to the same file:

server:/etc/goaccess.conf

```
time-format %T
```

Finally, the content of `/etc/goaccess.conf` should look like Figure 2.



```
#Any Apache log date format
date-format %d/%b/%Y
#
#W3C (IIS) & AWS | Amazon CloudFront (Download Distribution)
#date-format %Y-%m-%d

time-format %T

# The log_format variable followed by a space or \t for
# tab-delimited, specifies the log format string.
#
#Common Log Format (CLF)
#log-format %h %^[%d:%^] "%r" %s %b
#
#Common Log Format (CLF) with Virtual Host
#log-format %^:%^ %h %^[%d:%^] "%r" %s %b
#
#NCSA Combined Log Format
log-format %h %^[%d:%^] "%r" %s %b "%R" "%u"
#
#NCSA Combined Log Format with Virtual Host
#log-format %^:%^ %h %^[%d:%^] "%r" %s %b "%R" "%u"
#
#W3C
#log-format %d %^ %h %^ %^ %^ %^ %r %^ %s %b %^ %^ %u %R
#
#AWS | Amazon CloudFront (Download Distribution)
#log-format %d\t%^\t%^\t%b\t%h\t%m\t%^\t%r\t%s\t%R\t%u\t%^
```

Figure 2: Content of `/etc/goaccess.conf` after modifications

To analyze local log files using `goaccess`, execute:

Server's Console

```
goaccess -a > /var/www/html/report.html
```

To view the report of the `goaccess`, start *Mozilla Firefox* on the *client* and navigate to this URL:

gui's web browser

```
https://128.238.66.100/report.html
```

Examine the web statistics displayed in the browser. Also click on the side panel to see other sections.

Report

1. List the most frequently visited pages at the local Apache server during the most recent month, respectively.
2. List the web pages that have the most number of bytes transferred by the local during the most recent month, respectively.

12 ** System Status

Execute `netstat -l` to display the listening sockets in server host.

Server's Console

```
netstat -l
```

Run these commands to see the system services info and their status:

- Server's Console

  ```
  service --status-all
  ```

  Note: The symbols shown to the left of each service name indicates its status:

  `[+]` : Running

  `[-]` : Stopped

  `[?]` : Unknown

- Server's Console

  ```
  systemctl list-units --type=service --state=running
  ```

  Note: This command may not be available on `docker` `term` nodes.

- Server's Console

  ```
  systemctl list-unit-files --type=service --state=enabled
  ```

Note: This command may not be available on `docker` `term` nodes. Save their output for the lab report.

Report

1. Are the `rlogin` , `ssh` and `apache2` services running on the *server*?