ICS006 Firewall Configuration

Lab Objective

The objective of this lab is to understand the basics of firewall configuration, to design a modular firewall policy, and to configure an intrusion detection system for a control system environment.

In this lab, you will learn to:

- Configure the router in the ICS lab kit for remote shell management
- Implement and test configuration changes to the firewall using IPTABLES

Lab Environment

This lab requires an SSH client (such as PuTTY for Windows) and the ICS lab kit.

Lab Duration

45 minutes

Lab Tasks

Configure the PLC in the ICS lab kit with simple Modbus-enabled firmware, and use a simple HMI application to generate Modbus/TCP traffic. Configure remote shell access and message logging on the router included in the kit, then configure the firewall rules to permit or restrict Modbud/TCP traffic. Different levels of security will be implemented throughout this exercise. Because the later steps build upon the work done in the previous ones, the steps outlined in this document should be *read carefully* and completed in order in which they are given.

Background

The purpose of a firewall is to moderate and control network traffic, for the purpose of logging network activity and/or blocking potentially harmful network traffic. Firewalls provide a layer of protection between *private* (trusted) networks and *public* (untrusted) networks, a need which becomes increasingly important as more individuals and organizations connect to the Internet.

One of the most common categories of firewalls is a *packet filtering router*, a device which forwards packets between two or more networks. The router's operating system kernel checks the headers of each network packet to determine whether it should be forwarded to its intended destination or not. To make this determination, the router consults its set of *rules*; once a rule which matches the packet is found, the corresponding rule action is obeyed. The rules are searched in order, so only the first match counts; for this reason, these rules are referred to as "*rule chains*."

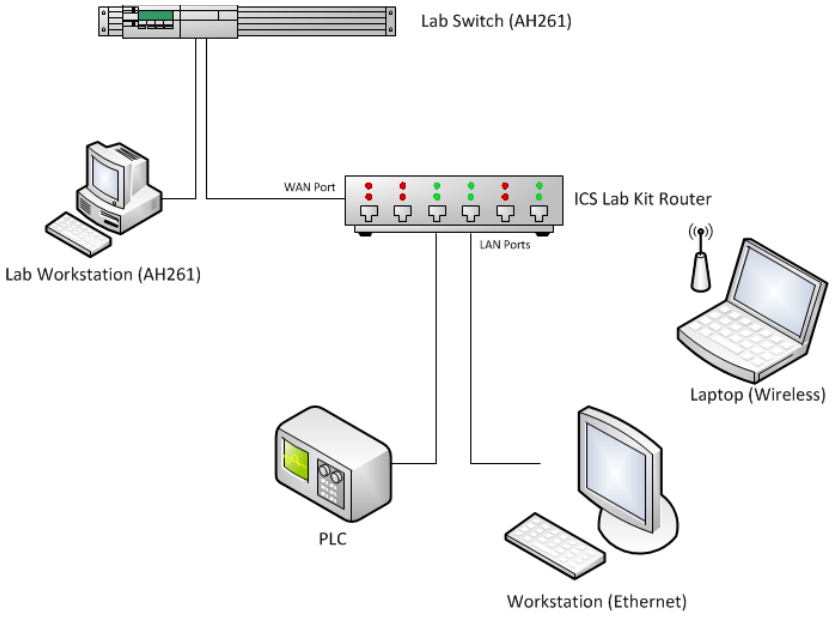
Modern commercial routers are typically shipped with a default set of rules which can provide a reasonable level of security for many applications. However, it is important to remember that a firewall which is not properly configured for the needs of your particular network can be more of a security risk than not having one at all! Relying on a firewall's default configuration can

provide a false sense of security, so it is important to be able to analyze and understand this configuration so that it can be strengthened with additional layers of protection as required for your network.

This lab assignment will involve analyzing and expanding the default firewall rule set for the router included with your ICS lab kit, with an emphasis on securing the Modbus protocol. In order to modify the firewall rules using a command-line interface, it will first be necessary to configure the router to allow remote shell access. This can be done from any workstation which has a Web browser and a secure shell (SSH) client installed. PuTTY is a free SSH and telnet client for Windows. See http://www.putty.org/ for more information, including downloads for all major versions of Windows. Other operating systems (Mac OS X, Linux, FreeBSD, etc.) typically include an SSH client as part of their standard configuration. In order to test our firewall configuration with the PLC included with your ICS lab kit, it will also be necessary to configure the PLC for networked HMI applications. A simple HMI program has been provided with this exercise, along with the corresponding ladder logic firmware.

Lab Scenario

After we configure and test the PLC, our next step will be to configure the router included with the ICS lab kit. It must accept remote SSH connections, and message logging must be enabled. Once this has been accomplished, the firewall can be configured at the command line using IPTABLES. Once a suitable configuration has been tested, it can be made permanent by creating a firewall configuration script. The router's WAN port will be connected to the lab network, and the workstation(s) will be connected to the router, either wirelessly or through the router's LAN ports:



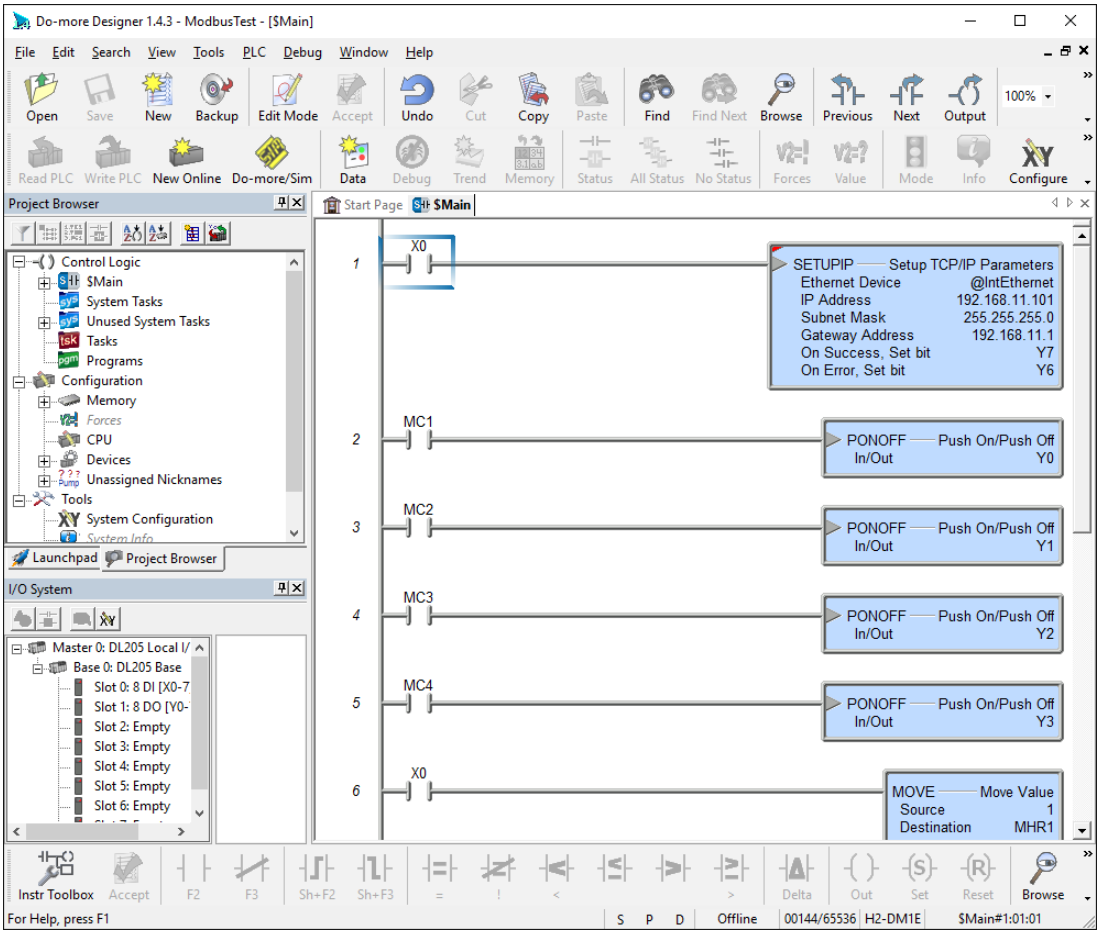
In this scenario, workstations on the lab network will act as Internet hosts, and we can simulate an attack on the ICS lab kit's internal network by initiating connection attempts from these hosts.

Lab Procedures

Part One: PLC Configuration

Before we proceed with our work on the router, we must first prepare the ICS lab kit for use. Set up your kit at your workstation and connect it to a power source, and connect the PLC to your lab workstation using one of the provided USB cables. You will also find two CAT-5 network cables at your workstation, one labeled "WS" (connected to your workstation) and the other labeled "LAN" (connected to the lab network). Connect the workstation network cable to the router's internal (LAN) network, and the lab network cable to the router's WAN port.

On your workstation desktop, you will find a folder named "Firewall Configuration Lab" which contains two additional folders: "ModbusTest" and "HMITest". Launch Do-more Designer 1.4 (there is a shortcut on the Start menu), click "Open," and then browse to the "ModbusTest" folder and open "ModbusTest.dmd" (if you receive a PLC connectivity error during this process, you can ignore it). You should see the following ladder logic program:



Switch your PLC to “Term” mode and download this program as instructed earlier, then be sure that all input switches on your PLC are set to the “Off” position. Switch your PLC to “Run” mode, then switch on the topmost input switch.

If your PLC was able to configure itself for the network, output LED #7 will be lit. Your PLC should now resemble the picture shown below:

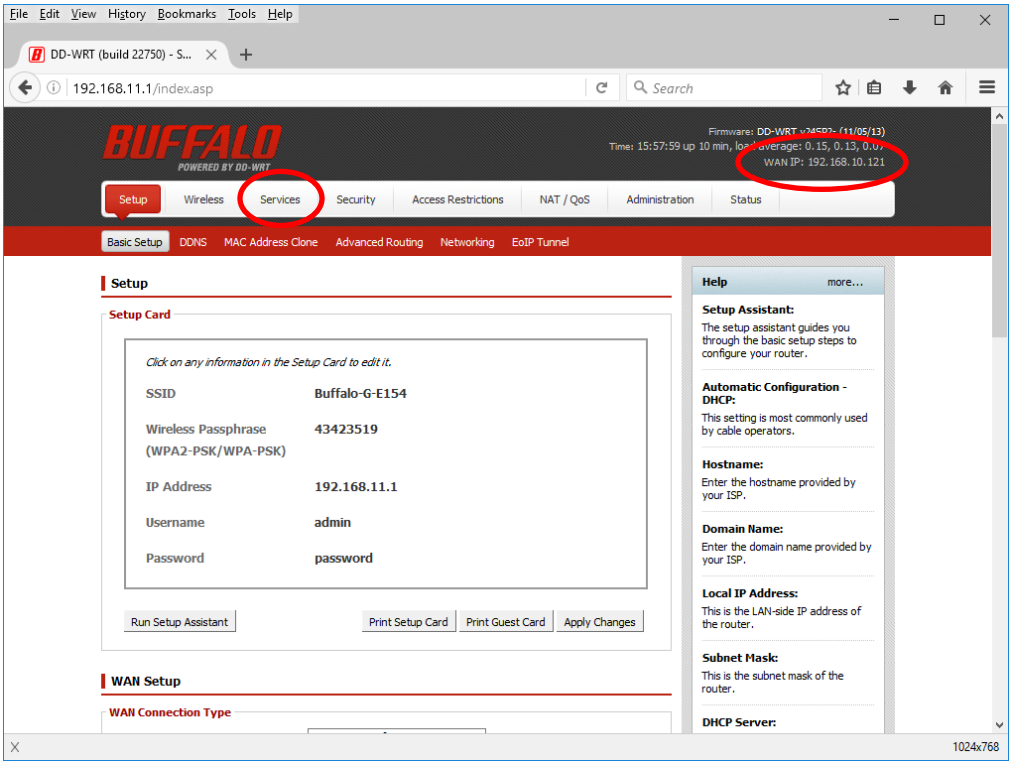
To test the PLC’s network connectivity, double-click the “HMITest” shortcut in the “Firewall Configuration Lab” folder to open a simple HMI program. The HMI window contains four momentary contact switches labeled Y0 through Y3; clicking any of these switches should toggle the corresponding output LED on the PLC.

If the HMI program functions as expected, the PLC has been successfully configured for the network, and the HMI can be closed. We will not be changing the PLC configuration further in this exercise, so Do-more Designer can also be closed. Be sure to leave the PLC in “Run” mode before proceeding to the next step.

Part Two: Router Configuration

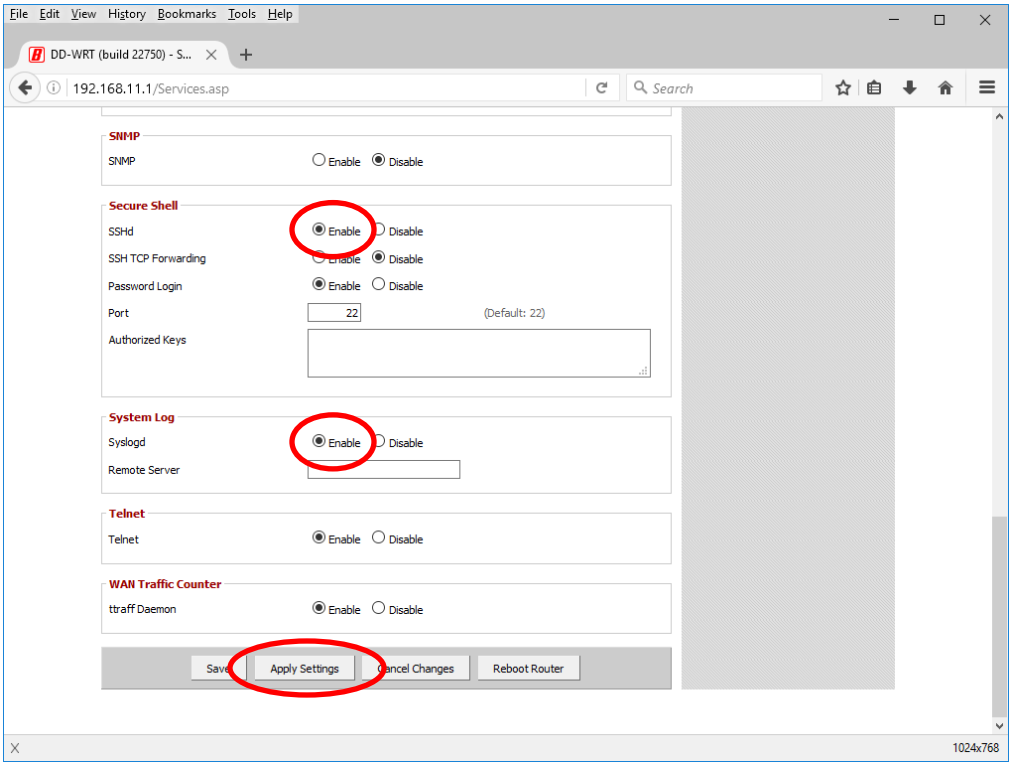
We will begin by enabling the router's secure shell (SSH) daemon, as well as a number of logging options. Open a Web browser on your lab workstation and enter the router's default IP address into the address bar: **http://192.168.11.1** (if you have assigned a new IP address to your router, enter this address instead). You may need to wait a few moments if you have just connected your workstation to the router, and/or if you have just powered up your ICS lab kit. When prompted, enter the default username and password (**admin** and **password**, respectively); if you have already assigned a new username and password, enter these instead.

The router's Web-based management interface should appear in your browser, as shown below. If this is your first time accessing this interface, a Setup Wizard may appear to configure the router's WAN port, to detect an Internet connection, etc. You may safely cancel the wizard if your router's WAN port is not connected to another network, or if the wizard takes an inordinate amount of time to complete.

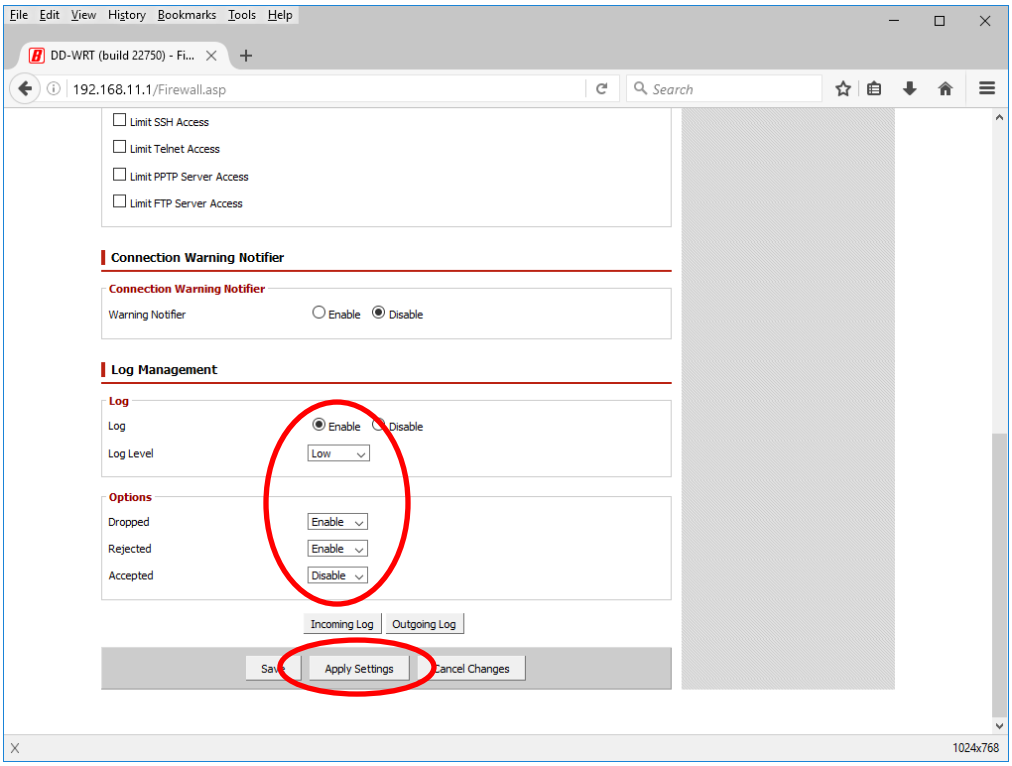


Note the "WAN IP" address shown near the top of your screen; we will be making use of this address later.

Open the "Services" option in the menu bar (circled above), then scroll down in the "Services" tab until you see the "Secure Shell" area. Click the "Enabled" radio button for the "SSHd" option, and then enable the "System Log" option in the area directly below, as indicated in the following screenshot. Click the "Apply Settings" button near the bottom of the screen.

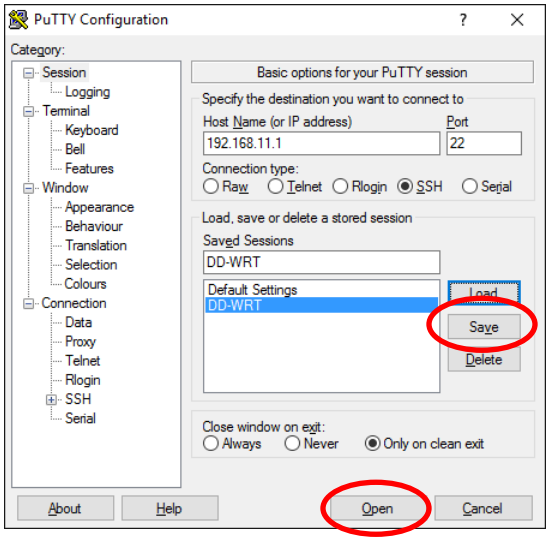


Wait a few moments for the router to finish committing your changes. Next, open the "Security" tab, scroll down to the "Log Management" area, and set the logging options as indicated below. Click "Apply Settings" to commit your changes.

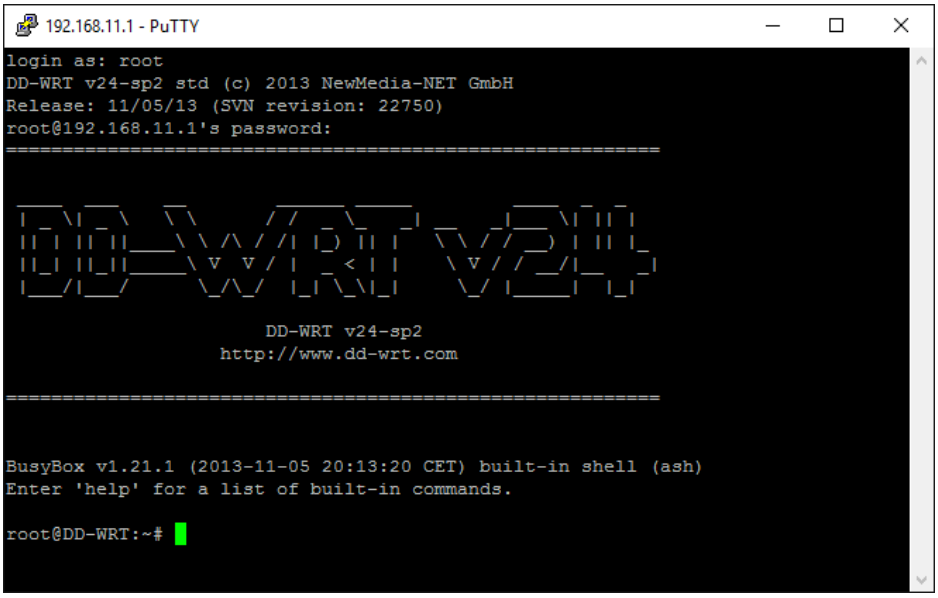


Part Three: SSH Client Configuration

In this step, we will use the PuTTY client for Windows to open an SSH session with the router. After launching PuTTY, enter the router's IP address in the "Host Name" field, enter "DD-WRT" in the "Saved Sessions" field, then choose "Save" (this is so you can re-open your connection later without entering the router's IP address again). Your screen should resemble the following:



Next, click "Open" to open an SSH session. If your router has been configured as described earlier, you will be presented with a login prompt. The default username is **root** with a password of **password**; again, if you have assigned a new username and password, enter these now.



A welcome banner should appear after you log in, as shown above. You are now at the shell prompt and are ready to issue shell commands to the router, including the commands which will configure the firewall.

Part Four: Firewall Configuration

The router included with your ICS lab kit is powered by an embedded Linux kernel, and like other distributions of Linux, the firewall rules can be configured using the IPTABLES command-line tool. This is the best way to test a new set of firewall rules: the changes made to the firewall configuration will take effect immediately, but they will not be permanently saved. If any errors are made during the testing process, the default configuration can be restored simply by rebooting the router. Later, we will see how to permanently save a set of IPTABLES commands as a firewall script. First, let us examine the default firewall rules. The rule chains are divided into *tables*; the default table is the *filter table*, which includes the following chains:

- INPUT (for packets destined to or entering the router's local sockets)
- OUTPUT (for packets sourced from or leaving the router's local sockets)
- FORWARD (for packets being forwarded through the router)

To see the current state of the INPUT rule chain, including the *target* (the corresponding action) for each rule, enter the **`iptables -L INPUT`** command at the prompt:

```
root@DD-WRT:/# iptables -L INPUT
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     0    --  anywhere             anywhere            state RELATED,ESTABLISHED
DROP       udp  --  anywhere             anywhere            udp dpt:route
DROP       udp  --  anywhere             anywhere            udp dpt:route
ACCEPT     udp  --  anywhere             anywhere            udp dpt:route
DROP       icmp --  anywhere             anywhere
DROP       igmp --  anywhere             anywhere
ACCEPT     0    --  anywhere             anywhere            state NEW
ACCEPT     0    --  anywhere             anywhere            state NEW
DROP       0    --  anywhere             anywhere
```

(For illustration, we will show the user's input at the shell prompt in bold, followed by an example of the output. The output that you see on your screen may be slightly different.)

This lists the default set of rules as defined by the router vendor. We can get more detailed information, including IP addresses and packet counts, by adding the **`-v`** and **`-n`** switches and the **`--line-numbers`** option, as shown here:

```
root@DD-WRT:/# iptables -vnL INPUT --line-numbers
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
num   pkts bytes target  prot opt in     out  source     destination
1    13705 1906K ACCEPT  0    --  *      *    0.0.0.0/0  0.0.0.0/0    state RELATED,ESTABLISHED
2        0     0 DROP    udp  --  vlan2  *    0.0.0.0/0  0.0.0.0/0    udp dpt:520
3        0     0 DROP    udp  --  br0    *    0.0.0.0/0  0.0.0.0/0    udp dpt:520
4        0     0 ACCEPT  udp  --  *      *    0.0.0.0/0  0.0.0.0/0    udp dpt:520
5        0     0 DROP    icmp --  vlan2  *    0.0.0.0/0  0.0.0.0/0
6        0     0 DROP    2    --  *      *    0.0.0.0/0  0.0.0.0/0
7      363 21080 ACCEPT  0    --  lo     *    0.0.0.0/0  0.0.0.0/0    state NEW
8     3319  178K ACCEPT  0    --  br0    *    0.0.0.0/0  0.0.0.0/0    state NEW
9        0     0 DROP    0    --  *      *    0.0.0.0/0  0.0.0.0/0
```

(The packet counts shown for the rules listed here were accumulated after a few minutes of light network activity. Again, your totals will be different.)

Notice that the INPUT chain has an ACCEPT policy, meaning that any incoming packets which do *not* match any of the specified rules are *accepted* (allowed to pass) by default. Fortunately, there is also a "catch-all" rule (#9) at the end of the chain which drops all packets which did not match any of the preceding rules (recall that the rules are processed in order, starting with the topmost rule). Notice also that the "**num**" column assigns a unique line number to each rule; we can refer to this number when replacing or deleting existing rules, or when inserting new rules.

Let us identify and critique each of the default rules:

- Rule #1 allows packets that are associated with an existing connection (the "ESTABLISHED" state), or packets that are starting a new connection that is related to an existing connection (the "RELATED" state). Put simply, this allows communication in both directions for connections that have been initiated from *inside* the firewall, but *not* new connections initiated from *outside* the firewall. Notice that the "**in**" column for Rule #1 contains a "*****", which means that this rule applies to input from *any* network source.
- Rules #2 and #3 drop all RIP (Routing Information Protocol) UDP traffic originating from **vlan2** (the WAN port of the router) or from **br0** (the bridge connecting the WAN port to the internal LAN), while Rule #4 allows RIP traffic from within the local LAN. This is an example of a *layered security policy*: Rules #2 and #3 *explicitly* deny a specific type of traffic from sources that are deemed to be unsafe, while Rule #4 *implicitly* allows traffic of the same type from sources that are deemed to be safe.
- Rule #5 drops ICMP (Internet Control Message Protocol) packets originating from **vlan2** (the WAN). Note that the target for this rule is DROP, which causes all ICMP requests (including ping requests) from the WAN to be denied *without* returning a message. By contrast, a target of REJECT *would* return a message.
- Rule #6 drops *all* IGMP (Internet Group Management Protocol) traffic.
- Rules #7 and #8 allow new connections originating from the internal network. This allows LAN users to initiate new communications, provided that these communications were not forbidden by one of the preceding rules.
- As stated earlier, Rule #9 drops *all* packets which did not match *any* of the preceding rules. In other words, if a given packet was not explicitly or implicitly allowed by one of the preceding rules, it is dropped.

This security policy is a reasonable starting point, as is the policy described in the FORWARD table. We will be changing the rules in these tables to enable additional security for the Modbus protocol; the steps outlined on the following pages will walk you through this process in detail. We will begin with a lax security policy, and will then strengthen it by adding more restrictions.

Along the way, we will attempt to use a Telnet client to open a connection to the PLC through TCP port 502 (the standard port for Modbus) from a workstation *outside* the firewall. Input entered at the keyboard will either be dropped or echoed back to the screen, and no output from the PLC will be received. However, the point is not to exchange meaningful data with the PLC in this way, but merely to test whether a connection attempt will be successful under the current rules; if it fails (either with an error message or a timeout), this will indicate that our rules are successfully blocking Modbus traffic, and will not allow an HMI to access the PLC either.

1. First, let's see what a successful connection attempt through Telnet looks like. Open a command line window on your workstation (the one connected to the router's internal LAN) and enter the following command to open a connection to the PLC using the Telnet client:

   `telnet 192.168.11.101 502`

   If your PLC is running the firmware that you downloaded earlier, the screen will clear, and any input entered at the keyboard will be echoed to the screen. Press "**CTRL -]**" (the default Telnet escape character), and a Telnet prompt should appear. Enter "**q**" to quit.

2. We will begin by configuring the firewall to allow Modbus traffic from the WAN. This will require the addition of two rules: a PREROUTING rule to forward all Modbus traffic to the PLC using NAT (Network Address Translation), and a FORWARD rule to allow this traffic to be passed by the firewall to its intended destination. Switch to the PuTTY client, then enter the following commands at the shell prompt (enter each command on a single line):

   `iptables -t nat -I PREROUTING -p tcp -d $(nvram get wan_ipaddr) --dport 502 -j DNAT --to 192.168.11.101:502`

   `iptables -I FORWARD -p tcp -d 192.168.11.101 --dport 502 -j ACCEPT`

   To test this configuration, try to use Telnet to connect to the PLC from *outside* the firewall. From another workstation on the lab network (several of these are provided in the lab), open a Telnet session as described in Step 1, replacing **`192.168.11.101`** with the WAN address of your router (you saw this earlier through the Web-based router management interface). Next, go to a *different* workstation on the lab network and attempt to open a *second* Telnet session in the same way. If the configuration succeeded, you should be able to connect from *both* lab workstations, just as you did *inside* the firewall. Make note of the IP addresses of *both* workstations (use the **`ipconfig`** command) before proceeding.

3. This configuration is very convenient, since it would allow our PLC to be managed and monitored remotely, but it is also very insecure. One improvement would be to block all Modbus traffic from a specific IP address (for example, the address of an outward-facing server in your DMZ which could be a source of attack). Add a rule to the FORWARD table which drops Modbus traffic originating from *one* of the two IP addresses from Step 2. We will accomplish this by implementing a *layered* security policy: we will *explicitly* deny Modbus traffic from the unwanted address, and then *implicitly* allow Modbus traffic from other addresses. For this to work, the new rule must be added *before* the rule we added in Step 2. New rules are added to the top of the list by default, so enter the following:

   `iptables -I FORWARD -p tcp -s 192.168.10.107 -j DROP`

   (Replace **`192.168.10.107`** with the IP address of *one* of the two lab workstations you used earlier. Either of the addresses will do; just remember which address corresponds to which workstation!)

To test this configuration, attempt to connect using Telnet from *both* of the workstations you used in Step 2. If the configuration succeeded, you will *not* be able to connect from the workstation whose IP address was specified in the new firewall rule, but you *will* still be able to connect from the *other* workstation. To see all of the rules in the FORWARD table, switch back to PuTTY and enter the following command at the shell prompt:

```
root@DD-WRT:/# iptables -vnL FORWARD --line-numbers
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
num pkts bytes target      prot opt in     out   source          destination
1      0     0 DROP        tcp  --  *      *     192.168.10.107  0.0.0.0/0
2     21   890 ACCEPT      tcp  --  *      *     0.0.0.0/0       192.168.11.101 tcp dpt:502
3      0     0 ACCEPT      47   --  *      vlan2 192.168.11.0/24 0.0.0.0/0
4      0     0 ACCEPT      tcp  --  *      vlan2 192.168.11.0/24 0.0.0.0/0      tcp dpt:1723
5    136 37691 ACCEPT      0    --  *      *     0.0.0.0/0       0.0.0.0/0      state
RELATED,ESTABLISHED
6      7   364 lan2wan     0    --  *      *     0.0.0.0/0       0.0.0.0/0
7      7   364 TCPMSS      tcp  --  *      *     0.0.0.0/0       0.0.0.0/0      tcp
flags:0x06/0x02 TCPMSS clamp to PMTU
8      0     0 ACCEPT      0    --  br0    br0   0.0.0.0/0       0.0.0.0/0
9      0     0 TRIGGER     0    --  vlan2  br0   0.0.0.0/0       0.0.0.0/0      TRIGGER
type:in match:0 relate:0
10     7   364 trigger_out 0    --  br0    *     0.0.0.0/0       0.0.0.0/0
11     7   364 ACCEPT      0    --  br0    *     0.0.0.0/0       0.0.0.0/0      state NEW
12     0     0 DROP        0    --  *      *     0.0.0.0/0       0.0.0.0/0
```

The first two rules are the rules we have added. New rules are added to the *top* of the list by default, so Rule #1 was added in this step, and Rule #2 was added in the previous step.

4. This configuration is still very insecure, so we will change it to block *all* Modbus traffic from the WAN. We are also interested in tracking failed Modbus connection attempts, since these may represent an attempted attack on our network, so we will use the logging facilities we enabled earlier to log these connection attempts *before* dropping them.

   First, we must delete the PREROUTING and FORWARD rules that we added earlier. Switch back to PuTTY and enter the following commands at the shell prompt:

   ```
   iptables -t nat -D PREROUTING 1
   iptables -D FORWARD 2
   iptables -D FORWARD 1
   ```

   (If you wish, you can confirm that these rules have been successfully deleted by viewing the PREROUTING *and* FORWARD *tables again, as described earlier.)*

   Next, add a new PREROUTING rule to forward all incoming Modbus traffic to the router itself, where it can be managed by an INPUT rule (again, enter this command on one line):

   ```
   iptables -t nat -I PREROUTING -p tcp -d $(nvram get
   wan_ipaddr) --dport 502 -j DNAT --to 192.168.11.1:502
   ```

   Finally, enter the following command (on one line) to create an INPUT rule which drops Modbus traffic with logging:

`iptables -I INPUT -p tcp -d 192.168.11.1 --dport 502 -j logdrop`

*(Note the target for this rule: "***logdrop***" is like "***DROP***" in that the connection will be dropped without returning a message, but unlike "***DROP***", the connection will be logged.)*

Before testing this rule with Telnet, view the INPUT table with the following command:

```
root@DD-WRT:~# iptables -vnL INPUT --line-numbers
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
num   pkts bytes target     prot opt in     out  source      destination
1        0     0 logdrop    tcp  --  *      *    0.0.0.0/0   192.168.11.1   tcp dpt:502
2     4522  435K ACCEPT     0    --  *      *    0.0.0.0/0   0.0.0.0/0      state
RELATED,ESTABLISHED
3        0     0 DROP       udp  --  vlan2  *    0.0.0.0/0   0.0.0.0/0      udp dpt:520
4        0     0 DROP       udp  --  br0    *    0.0.0.0/0   0.0.0.0/0      udp dpt:520
5        0     0 ACCEPT     udp  --  *      *    0.0.0.0/0   0.0.0.0/0      udp dpt:520
6        3   108 DROP       icmp --  vlan2  *    0.0.0.0/0   0.0.0.0/0
7       99  3168 DROP       2    --  *      *    0.0.0.0/0   0.0.0.0/0
8        0     0 ACCEPT     0    --  lo     *    0.0.0.0/0   0.0.0.0/0      state NEW
9      610 33777 ACCEPT     0    --  br0    *    0.0.0.0/0   0.0.0.0/0      state NEW
10     286 25546 DROP       0    --  *      *    0.0.0.0/0   0.0.0.0/0
```

Your "**packets**" and "**bytes**" counters will likely be different, but note that both counters are still at zero for our new rule (#1). Attempt to open a connection using Telnet from the two lab workstations that you used in Step #3; if the configuration succeeded, the attempt should fail from *both* workstations. Now, view the INPUT table again:

```
root@DD-WRT:~# iptables -vnL INPUT --line-numbers
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
num   pkts bytes target     prot opt in     out  source      destination
1        2   104 logdrop    tcp  --  *      *    0.0.0.0/0   192.168.11.1   tcp dpt:502
2     4544  437K ACCEPT     0    --  *      *    0.0.0.0/0   0.0.0.0/0      state
RELATED,ESTABLISHED
3        0     0 DROP       udp  --  vlan2  *    0.0.0.0/0   0.0.0.0/0      udp dpt:520
4        0     0 DROP       udp  --  br0    *    0.0.0.0/0   0.0.0.0/0      udp dpt:520
5        0     0 ACCEPT     udp  --  *      *    0.0.0.0/0   0.0.0.0/0      udp dpt:520
6        3   108 DROP       icmp --  vlan2  *    0.0.0.0/0   0.0.0.0/0
7      100  3200 DROP       2    --  *      *    0.0.0.0/0   0.0.0.0/0
8        0     0 ACCEPT     0    --  lo     *    0.0.0.0/0   0.0.0.0/0      state NEW
9      613 33933 ACCEPT     0    --  br0    *    0.0.0.0/0   0.0.0.0/0      state NEW
10     288 25702 DROP       0    --  *      *    0.0.0.0/0   0.0.0.0/0
```

Check the "**packets**" and "**bytes**" counters for Rule #1 again, and notice that they have changed (the sample shows the counts from a single Telnet connection attempt). This indicates that the rule successfully blocked our connection attempt(s); according to the corresponding rule action, these connection attempts should also have been logged. To view the log, use the **`cat`** (concatenate) command:

`cat /var/log/messages`

This command will cause the contents of the entire log file to be dumped to the console. Notice the most recent (bottommost) entry, which should resemble the following:

```
Jun 17 16:33:16 DD-WRT kern.warn kernel: DROP IN=vlan2 OUT=
MAC=cc:e1:d5:78:e1:54:f8:bc:12:a6:c5:54:08:00:45:00:00:30 SRC=192.168.10.107 DST=192.168.11.1
LEN=48 TOS=0x00 PREC=0x00 TTL=128 ID=8150 DF PROTO=TCP SPT=52578 DPT=502 SEQ=3461684026 ACK=0
WINDOW=8192 RES=0x00
```

This log entry indicates the date and time of the blocked connection attempt, as well as the IP address that the connection attempt originated from. In this example, the attempt was made from the same lab workstation that was explicitly blocked in Step 3.

5. Having successfully tested our firewall configuration, we will now make our changes permanent by saving them to a firewall script (remember, the IPTABLES commands entered at the shell prompt will only remain active until the router is rebooted!). The easiest way to do this is from the Web-based management interface, where we can enter the commands that are needed to change the default firewall configuration to the configuration we want.

   Reopen the management interface in your Web browser, then open the "Administration" tab and look for a multi-line "Commands" field. Copy the following commands (the same ones we just tested at the shell prompt) into this field, then click the "Save Firewall" button:

   ```
   iptables -t nat -I PREROUTING -p tcp -d $(nvram get
   wan_ipaddr) --dport 502 -j DNAT --to 192.168.11.1:502

   iptables -I INPUT -p tcp -d 192.168.11.1 --dport 502 -j
   logdrop
   ```

6. To ensure that these changes have been successfully saved, reboot the router either by cycling the power or entering the "**reboot**" command at the shell window. Your PuTTY session (if it is still open) will be terminated. After rebooting, you may need to renew your IP address to open a new session (open a command window at your workstation and enter the commands "**ipconfig /release**" and "**ipconfig /renew**").

   Once the router has finished rebooting, relaunch PuTTY and open a new SSH session, and then view the PREROUTING and INPUT tables again:

   ```
   iptables -vnL INPUT --line-numbers

   iptables -t nat -vnL PREROUTING --line-numbers
   ```

   The rules from the script should appear at the top of each table. If so, congratulations! You have successfully tested and committed your firewall configuration.