

Noroff University College

TUTORIALS FOR THE ICS SIMULATION TESTING

Submitted in partial fulfilment
of the requirements of the degree of

BACHELOR OF CYBERSECURITY

of Noroff University College

Ali Humayun

Trondheim, Norway
July 2024

Declaration

I declare that the work presented for assessment in this submission is my own, that it has not previously been presented for another assessment, and that work completed by others has been appropriately acknowledged.

Name: Ali Humayun

Date: July 25, 2024

Contents

References	1
A Tutorials	2
A.1 Tutorial: ICS Test lab Setup	2
A.2 Description of the Test Setup	2
A.2.1 Required Software	4
A.2.2 Simulation Setup	6
A.3 Tutorial 2: Pivoting and lateral movement	6
A.3.1 Identify Active Hosts	7
A.4 Tutorial: PLC False Command Injection	9
A.5 Tutorial: Hardening the Industrial Switch	11
B Research Code	17
B.1 PLC Ladder Logic	17
B.2 VyOS Configuration Script	19
B.3 Scada-LTS Docker compose file	20

List of Figures

A.1	GNS3 virtual network topology	3
A.2	Test process visual representation	4
A.3	Scada-LTS watch list	5
A.4	Modified GNS3 topology for PLC command injection	11
A.5	virt-manager with added Network interface	12
A.6	Ettercap poisoning targets ARP cache	12
A.7	Wireshark showing filtered SCADA-PLC traffic	13
A.8	Metasploit <i>modbusclient</i> module in action	13
A.9	OpenPLC device monitoring screen after attack	14

List of Tables

A.1	GNS3 components and their respective platforms	5
A.2	Current software versions	5
A.3	Compromised machine details	7
A.4	Host discovery results	8

References

IEEE (Mar. 1999). *IEEE Standards for Local and Metropolitan Area Networks: Virtual Bridged Local Area Networks*. Technical report, pages 1–214. DOI: [10.1109/IEEESTD.1999.89204](https://doi.org/10.1109/IEEESTD.1999.89204).



Tutorials

A.1 Tutorial: ICS Test lab Setup

In order to simulate an ICS environment, we will set up a test lab which involves virtualisation of a minimum number of devices and network settings required for a specific industrial task. This initial setup will be obligatory to complete all the tutorials in the next sections and will allow the user to engage with the target ICS environment.

This tutorial assumes that the user is comfortable using Unix based Operating systems, shell scripting, containerisation technologies, and virtual machines. Additionally, a solid knowledge of networking concepts is required and the user is completely familiar with ICS and penetrating testing basics.

A.2 Description of the Test Setup

Figure A.1 show the target ICS architecture that we will be achieving at the end of this tutorial. The setup we will create in this tutorial is a minimal working representation of an ICS. The resulting testlab contains one general-purpose router which routes traffic between enterprise network and industrial network. It also routes traffic between different components present inside the Control zone (Level 0 to 2) and site operations (Level 3). Control zone and Industrial zones are separated by two system acting also as an HMI. There is local workstation in the zone which is used to access PLC and rest of the field devices. The Site operations zone contains 1 operator workstation which has access to the control zone via switch 1. This machine will act as the compromised system during penetration testing exercises of the environment. An attack machine running pentesting operating system is also added to the ICS setup in order to execute offensive assessments.

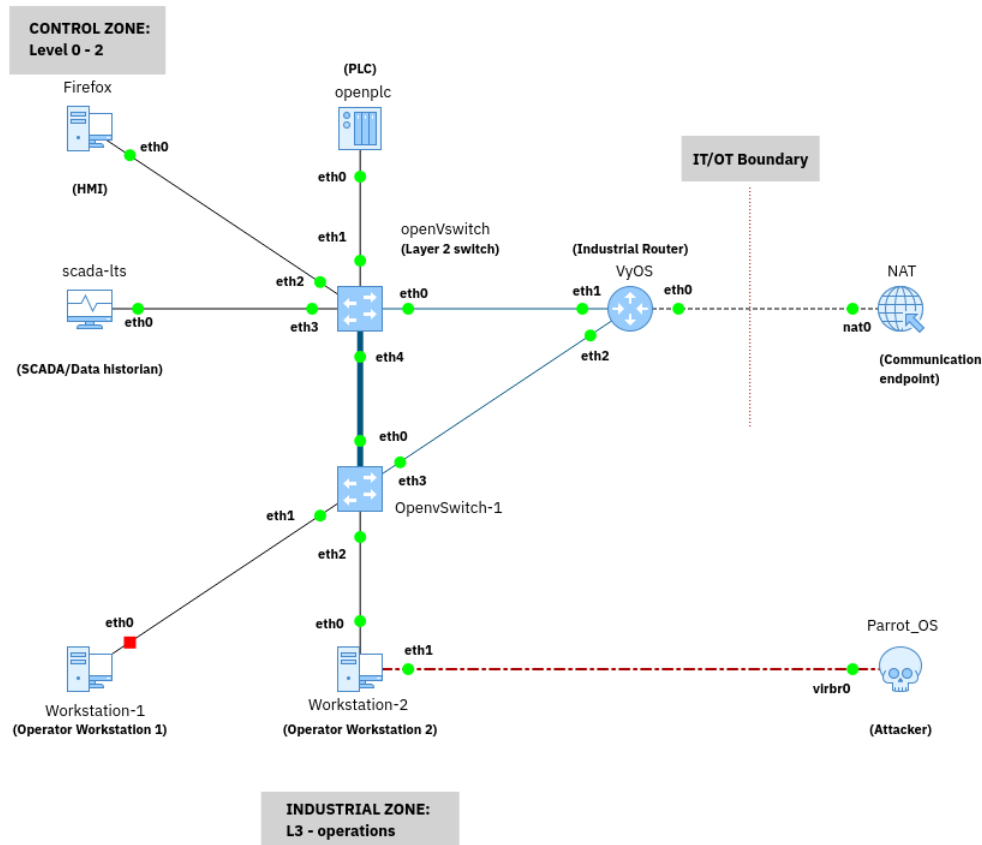


Figure A.1: GNS3 virtual network topology

The discussed setup can be deployed in a number of ways, depending on the level of expertise of the user, aim, and time constraints. For those interested in a much detailed setup, refer to the video links provided on how to configure OpenPLC and Scada-LTS. Installation of the required software on the user's machine is beyond the scope of this project. In order to assist the user, demonstration videos and pre-configured virtual machines are shared on Onedrive. Additionally, the project can be followed at the following github repository: <https://github.com/MetalAlloys/ICS-simulation>. Kindly refer to the respective user support platforms of the mentioned software if you encounter any installation issues. In addition to the Github repository, the code used in the configurations of Virtual machines inside GNS3 is provided as an artifact in Appendix E.

The whole setup will be run inside GNS3 with a combination of QEMU virtual machines and docker containers. Table A.1 enlists the components inside the setup with their respective platforms. It is important to note that there is no special requirement of running these components the way they are described here e.g. you can run GNS3 on a remote server or a cloud machine but that is out of the scope of these tutorials.

At this point, user should be able to work with GNS3 appliances. They are community made and ready-to-use virtual machines or docker containers that can be imported directly into a GNS3 project. Read the how-to documentation at <https://docs.gns3.com/docs/using-gns3/beginners/import-gns3-appliance/>. One more tutorial on how to create GNS3 templates is also required to import you own QEMU VMs in a GNS3 project.

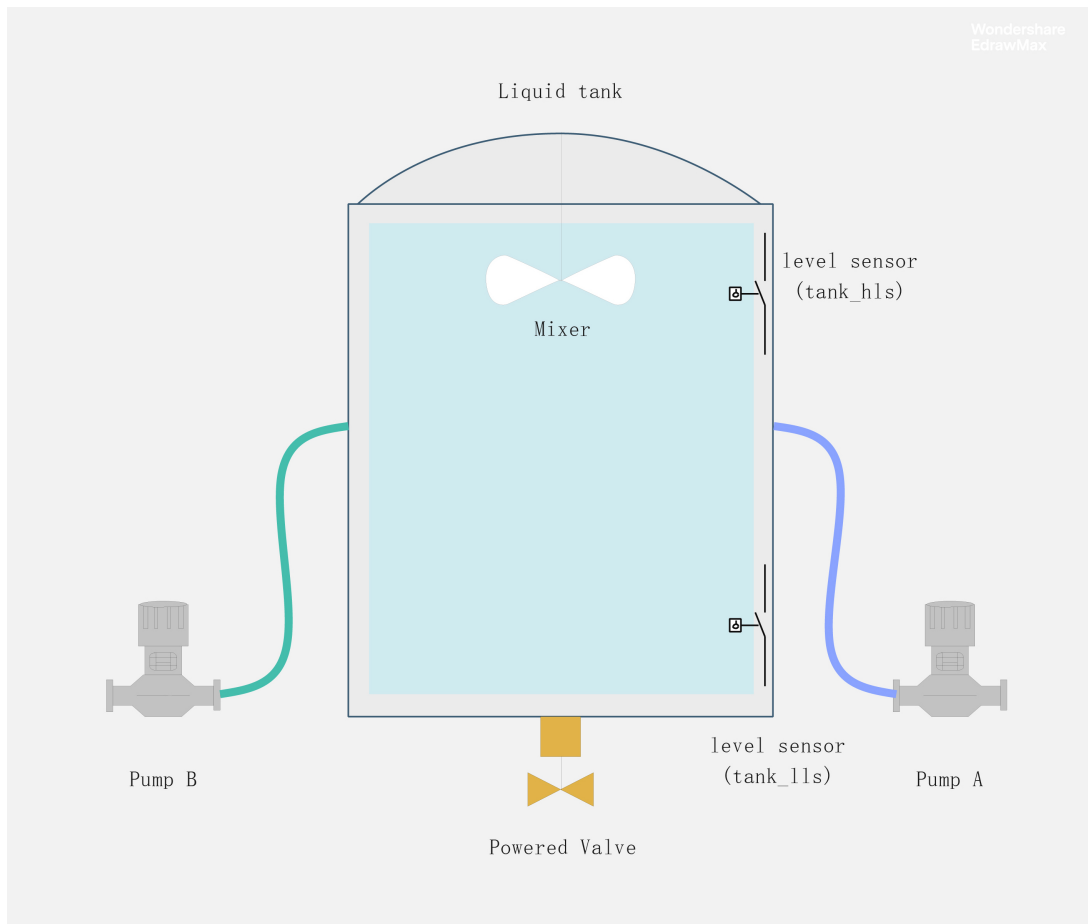
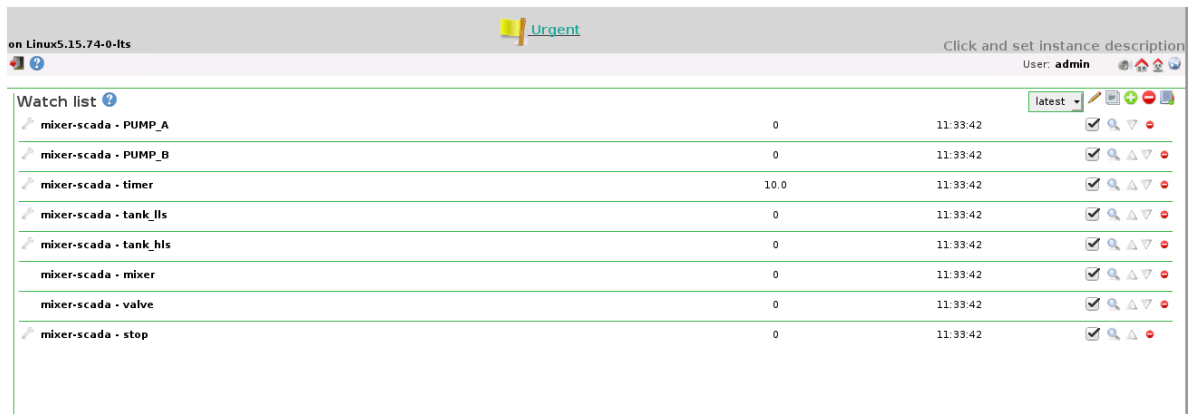


Figure A.2: Test process visual representation

A.2.1 Required Software

The following list of software is required to run the complete setup. Note that the above setup has been tested on Linux kernel *6.0.10-arch2-1*, but is expected to run fine on older kernel versions and other operating systems. Table A.2 shows the list of software with their current versions on my host machine for reference.

- Host machine running on a CPU that supports virtualization
- [GNS3](#) to emulate, configure, test and troubleshoot virtual and real networks. Two GNS3 appliances will be used inside the setup: [OpenVswitch](#) (Switching), and a minimal Linux based OS running just a browser called webterm.
- Docker and [Docker Compose](#) (recommended) which is an Open source containerization technology. Running applications as containers in this case helps remove installation problems on varying platforms. Docker will be used for the following tasks:
 - Run [OpenPLC](#) and [Scada-lts](#) as docker containers inside Virtual machines
 - OpenVswitch appliance runs a docker container inside GNS3



Component	Value	Timestamp	Actions
mixer-scada - PUMP_A	0	11:33:42	[Check] [Search] [Refresh] [Close]
mixer-scada - PUMP_B	0	11:33:42	[Check] [Search] [Refresh] [Close]
mixer-scada - timer	10.0	11:33:42	[Check] [Search] [Refresh] [Close]
mixer-scada - tank_hls	0	11:33:42	[Check] [Search] [Refresh] [Close]
mixer-scada - tank_hls	0	11:33:42	[Check] [Search] [Refresh] [Close]
mixer-scada - mixer	0	11:33:42	[Check] [Search] [Refresh] [Close]
mixer-scada - valve	0	11:33:42	[Check] [Search] [Refresh] [Close]
mixer-scada - stop	0	11:33:42	[Check] [Search] [Refresh] [Close]

Figure A.3: Scada-LTS watch list

Component	Platform	Information
Router	QEMU	Alpine linux
Switch	Docker	GNS3 appliance
PLC	QEMU + Docker	Alpine Linux
SCADA	QEMU + Docker	Apline linux
Workstation	QEMU	Apline Linux
Local machine	Docker	GNS3 appliance
Attack machine	<i>User's choice</i>	Parrot OS

Table A.1: GNS3 components and their respective platforms

- QEMU (A generic and open source machine emulator and virtualizer). In the spirit of using only open source software for this project, QEMU machines will be run inside GNS3, however, other preferred platforms such as VMware or VirtualBox can be used as well.
- [Virt-manager](#) which is a desktop user interface for managing virtual machines through [libvirt](#) (This requirement is optional, but very helpful as QEMU originally works from the command line which can be troublesome)
- VNC viewer of any kind for having graphical access to the VMs running inside GNS3

Software	Version
Linux Kernel	6.1.6-arch1-3
QEMU	7.2.0
Docker runtime	20.10.22
Docker compose	2.14.2
GNS3 server	2.2.34
TigerVNC Viewer	1.12.90
libvirt	9.0.0
virt-manager	4.1.0

Table A.2: Current software versions

A.2.2 Simulation Setup

This setup involves importing pre-configured QEMU images into GNS3 server running on the host machine. Ideal for those who are pressed for time or are advanced users quickly wanting to jump to the actual cybersecurity part.

1. Clone the project github repository mentioned above
2. Make sure that GNS3, Docker, and VNC viewer are up and running
3. Open GNS3 and create a new project
4. Download and import the following GNS3 appliances:
 - [Open vSwitch](#)
 - [webterm](#)
5. Download the QEMU disks *openplc_run.qcow2*, *scadabr.qcow2*, *VyOS-vyatta-hda.qcow2* from the onedrive link (shared currently with testers only)
6. Find the GNS3 working directory by clicking Edit -> Preferences -> General -> Local paths. Move all the downloaded *.qcow2* files into *gns3/images/QEMU*.
7. Import the disks as QEMU VMs in GNS3.
8. Go to File -> Open project and select the file *ics_mod.gns3* in repository main github directory
9. Turn on the machines beginning from the router to the OpenPLC QEMU machine. Give about 20-30 seconds for every machine to boot up
10. Test connectivity by opening the console to all the nodes/machines in the topology. Connectivity can be tested by pinging machines if they can reach each other. Other than that, inspect the ip addresses and open ports using *ip* and *ss* commands.
11. You can also check the IP addresses leased by the VyOS DHCP server to some of the nodes by running *show dhcp server leases* on the VyOS console

A.3 Tutorial 2: Pivoting and lateral movement

After gaining access to a victim host, it is often required to interact with other machines present inside the network which the attack machine does not have direct access to. This technique is called pivoting which also enables an attacker to make lateral movements inside the compromised network. Note that these tutorials are not aimed at pentesting enterprise networks for vulnerabilities. Therefore, at this point, we assume that the phase 1 of the ICS cyber kill chain is complete. This means that an important workstation in the supervisory level has been compromised.

Start by verifying that you have SSH access to the machine with hostname 'Workstation-1'. Assuming that during ICS kill chain phase 1, found out that a user named 'engineer-1' has access to this machine

and we were able to ssh into that account by using stolen or cracked credentials. Find the IP of the attack machine (assigned by DHCP server on the VyOS router) by running the following command from the VyOS terminal:

```
| show dhcp server leases
```

And SSH;

```
| ssh engineer-1@<COMPROMISED-MACHINE-IP>
```

Gather basic information about the machine. Basic shell commands such as `uname`, `ifconfig`, `ip`, `ss`, `id`, `hostname`, `sudo` can be helpful in this regard. We will not be enumerating this machine for any further weaknesses or passwords. However, this requirement depends on the goals of a pentest. In our case, we are only interested in breaching the industrial network.

Item	Description
OS	Alpine Linux x86_64
Hostname	ics-server
Kernel version	7.2.0
Open ports	22
Network interfaces	eth1 (192.168.0.77/24) eth0 (192.168.122.95/24)
Privileged	No

Table A.3: Compromised machine details

We note that this machine is part of another subnet via `eth0` interface. It would be interesting to scan for devices in this internal network. In case of no or weaker firewall policies, we can use a chain of proxies to route network traffic from our pentesting OS to the target network. We will test this functionality later but right now, we will transfer a statically compiled *nmap* binary into this machine. At this point, whenever we need to transfer a tool, we can simply use the command `scp` to copy the files over SSH.

A.3.1 Identify Active Hosts

In order to scan the industrial zone for potential targets, we will use a static *nmap* binary. We assume that the user `engineer-1` is a low privilege user and does not have the ability to install packages. Therefore, we will use a common technique of using static binaries for popular networking tools.

Clone the following github repository on your attack machine and copy it into the home folder of target machine using `scp`:

```
| git clone https://github.com/opsec-infosec/nmap-static-binaries
```

```
| scp -r ./nmap-static-binaries engineer-1@<compromised-workstation-IP>:~/
```

The directory above contains all the necessary files needed to run **nmap** as a standalone x86_64 binary. We will proceed by scanning the subnet `192.168.0.0/24` for active hosts. Run the following command from the compromised machine to run an *nmap* 'no port' scan:

```
cd nmap-static-binaries/linux/x86_64/
# Then run
./run_nmap.sh -sn 192.168.0.0/24
```

The excerpt below shows the output of running the scan:

```
Starting Nmap 7.93SVN ( https://nmap.org ) at 2023-04-18 14:44 UTC
Nmap scan report for 192.168.0.1
Host is up (0.0037s latency).
Nmap scan report for 192.168.0.41
Host is up (0.0048s latency).
Nmap scan report for 192.168.0.43
Host is up (0.0081s latency).
Nmap scan report for 192.168.0.90
Host is up (0.000038s latency).
Nmap scan report for 192.168.0.102
Host is up (0.0036s latency).
Nmap done: 256 IP addresses (5 hosts up) scanned in 2.21 seconds
```

After that we can proceed to scan individual hosts for potential targets. The following excerpt shows the output of scanning the host at 192.168.0.41:

Continue scanning other hosts in the subnet and note down the output.

After a complete port and services scan, note all the retrieved information in a table. The table A.4 registers all the scanning results.

Table A.4: Host discovery results

Host IP	Port(s) (TCP only)	Type
192.168.0.1	22 (ssh), 53 (dns)	Router
192.168.0.41	22 (ssh), mbap (502), 8080 (http)	PLC
192.168.0.43	22 (ssh), 8000 (http) , 3306 (mysql)	SCADA/HMI
192.168.0.84	None	Local machine

After scanning the ports completely, we see that the machine at IP address 192.168.0.41 has port 502 open which corresponds to Modbus TCP standard port according to IANA. It also has an http server running. Lets take a look at that. Since our attack machine browser does not have direct access to that machine, we will use SSH static port forwarding to get access to that port. Use the following command to locally port forward port 8080 from that host:

```
ssh -L 127.0.0.1:8888:192.168.0.41:8080 engineer-1@192.168.0.95

ics-server:~/nmap-static-binaries/linux/x86_64$ ./scan.sh 192.168.0.41
...
Scanning 192.168.0.41 [65535 ports]
Discovered open port 8080/tcp on 192.168.0.41
Discovered open port 22/tcp on 192.168.0.41
Discovered open |colorbox{green}{port 502/tcp}|on 192.168.0.41
Completed Connect Scan at 14:48, 1.74s elapsed (65535 total ports)
...
```

Navigate to <http://localhost:8888> on your Pentesting machine and you will be welcomed by the OpenPLC control panel which will confirm that this machine is actually a PLC controller!

You can use similar techniques to figure out the identity of the machine at IP address 192.168.0.43 which actually is SCADA server.

As an additional activity, the MySQL server running at port 3306 can also be scanned for vulnerabilities and CVEs. The actual PLC firmware can also be targeted for weaknesses.

Similarly, note down all the potential attack points present in the control zone and proceed by developing your own attack methodology according to the research case or preferred aim.

A.4 Tutorial: PLC False Command Injection

In order to learn penetration testing on any system, it is important to simulate attack scenarios to understand various important aspects of its execution. This not only enables a cybersecurity professional to understand how APTs attack systems but also helps devise stronger defence strategies.

At this point, there are a range of attacks that we can attempt through our compromised workstation, but we will choose a command injection attack. The attack will try to modify the holding register value of the PLC in order to disrupt the normal functioning of the process, something that was done by Stuxnet.

In order to write a false value to the holding register, we need the Modbus address and data type for a register inside OpenPLC. To find the details about the register, we need to intercept the traffic between Scada-LTS and OpenPLC to infer the information. We will use a MITM attack for this purpose.

A Man-in-the-Middle attack is a method of actively listening to the network traffic between two or more networked devices. The aim is to position yourself between the victim machines and intercept packets before forwarding them. It is important that the attack machine forwards the received packets, otherwise, it will cause a DOS attack on the service. Although, DOS attack is itself an attack, but it defeats the purpose of security testing in this case as we are trying to map the behaviour of victim machines.

A MiTM attack can be achieved by a variety of techniques. Some of the most commonly encountered techniques are:

- ARP (Address resolution protocol) spoofing
- DNS spoofing
- IP spoofing
- Session hijacking
- Wireless tap
- SSL hijacking

We will emulate ARP spoofing/ ARP cache poisoning attack on the machines inside the target ICS. The choice of attack is not arbitrary but is based on the level of access we have and the type of services running on the system. The idea is to send spoofed ARP messages to the target host. The Address Resolution Protocol (ARP) is vulnerable to spoofing because ARP messages include no authentication (RFC 826 ¹, updated RFC 5227 ², 5494 [1–3] ³) and therefore any host connected to the target network can emit an ARP request or response coming from another host. This way, the attacker poisons the ARP cache of the target host and replaces the MAC address of one of the communicating hosts in the target's ARP cache.

Run the following command(s) on the compromised machine to get an idea of how an ARP cache looks like. A sample output looks like as shown in listing A.1.

```
arp -a
...
192.168.0.43 dev eth1 lladdr 0c:b6:47:70:00:00 REACHABLE
192.168.0.41 dev eth1 lladdr 0c:60:42:d0:00:00 REACHABLE
192.168.122.1 dev eth0 lladdr 52:54:00:f4:c5:1d REACHABLE
192.168.0.1 dev eth1 lladdr 0c:9e:f4:26:00:01 REACHABLE
...
```

Listing A.1: ARP cache of SCADA server

In order to do a full MITM attack in a simpler way, we will modify our GNS3 topology to add our attack machine directly to the control zone switch (openVswitch). Figure A.4 shows the modified GNS3 topology for this attack.

After connecting, it should automatically get an IP address assigned through DHCP server running on VyOS router. If you are unable to get an IP address, modify your network settings by first adding a new Virtual Network Interface to your pentesting OS.

Change the Network source to 'bridge' and device name to `virbr0`. Figure A.5 shows how it can be done in *virt-manager* for those using QEMU.

For ARP poisoning, we will use a tool called *Ettercap* ⁴. It is capable of sniffing live TCP/IP connections, filter contents and has a GUI.

Open Ettercap in graphical mode and select the Network interface that is connected to the industrial switch (openVSwitch). Scan for hosts and add hosts 192.168.0.41 and 192.168.0.43 to the target list. Go the menu and select ARP poisoning with default settings and it will poison the target ARP caches immediately. Refer to figure A.6 for how Ettercap would look like after starting the ARP poisoning attack.

Open Wireshark and read the traffic between the PLC and the Scada server. Among other things, we will look for any requests made to read holding registers i.e. Modbus function code 3. Use the filter `modbus.func_code == 3` to filter for Modbus function code 3. Without any internal knowledge of the process, we realise that there is a register at address 1024 with a set value of 10.

¹<https://www.ietf.org/rfc/rfc826.txt>

²<https://www.ietf.org/rfc/rfc5227.txt>

³<https://www.ietf.org/rfc/rfc5494.txt>

⁴<https://www.Ettercap-project.org/>

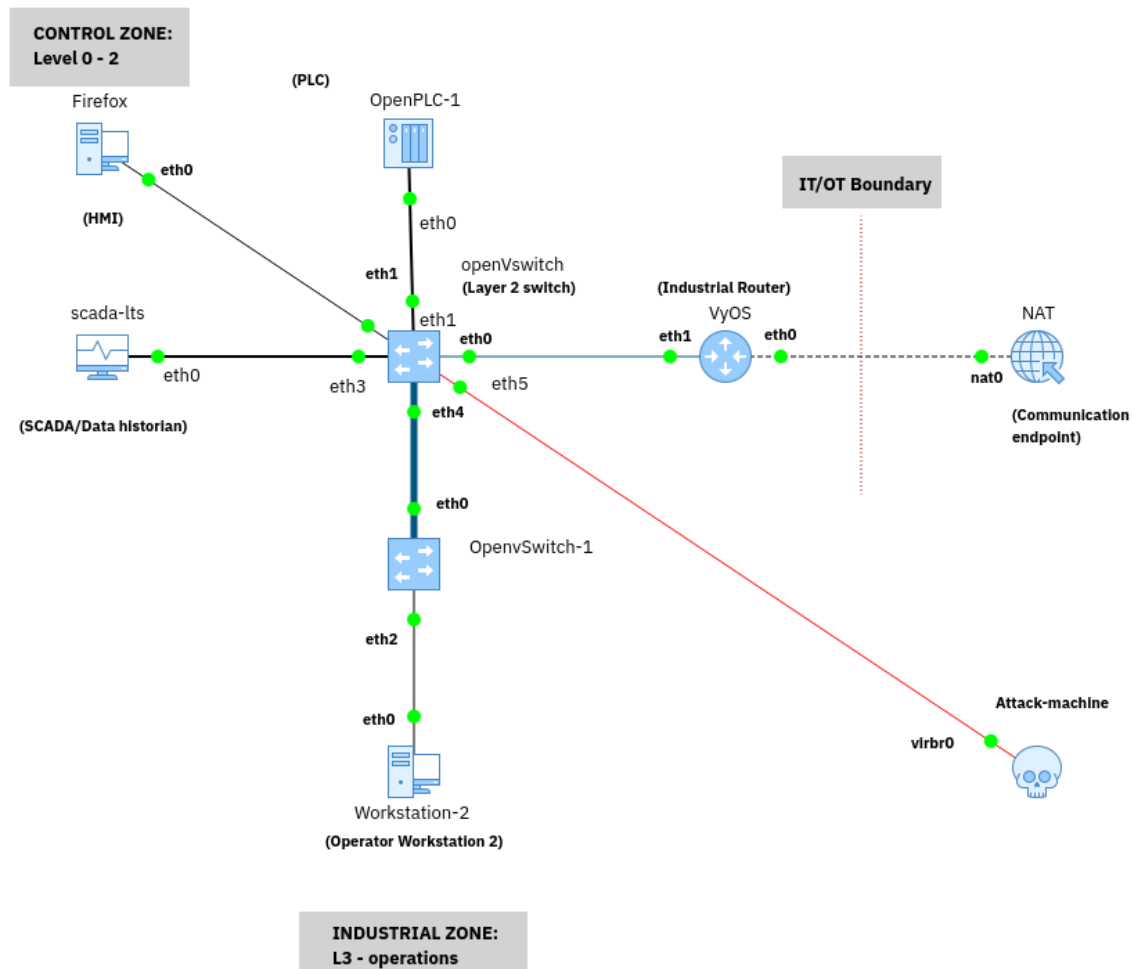


Figure A.4: Modified GNS3 topology for PLC command injection

What if we change the value in transit to cause disruption on the process? We are already aware that Modbus is an unauthenticated and plain-text protocol and therefore we can communicate with it from a rogue machine.

We will use a metasploit module to write to that register with our own value. Figure A.8 shows how the settings for that specific metasploit module would be and the return message when the exploit is run. Set the values of module parameters according to your attack machine while keeping the DATA_ADDRESS same.

After a successful run, you can check the monitoring page of the OpenPLC controller to that the value is indeed changed to the desired value as can be seen in Figure A.9

A.5 Tutorial: Hardening the Industrial Switch

The flexibility of the GNS3 topology that we have created allows us to test various defensive strategies. These strategies can be chosen based on the level of complexity and system requirements. e.g. the MITM attack in the previous tutorials can be mitigated using a range of techniques. One can either

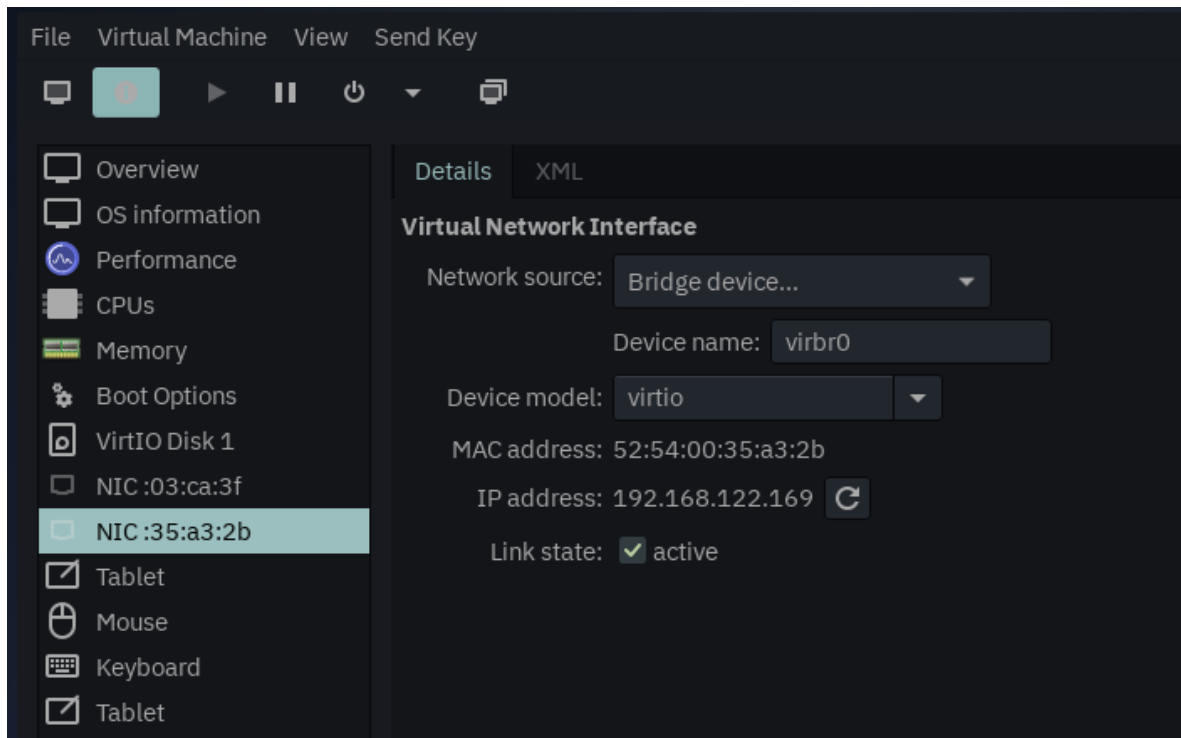


Figure A.5: virt-manager with added Network interface

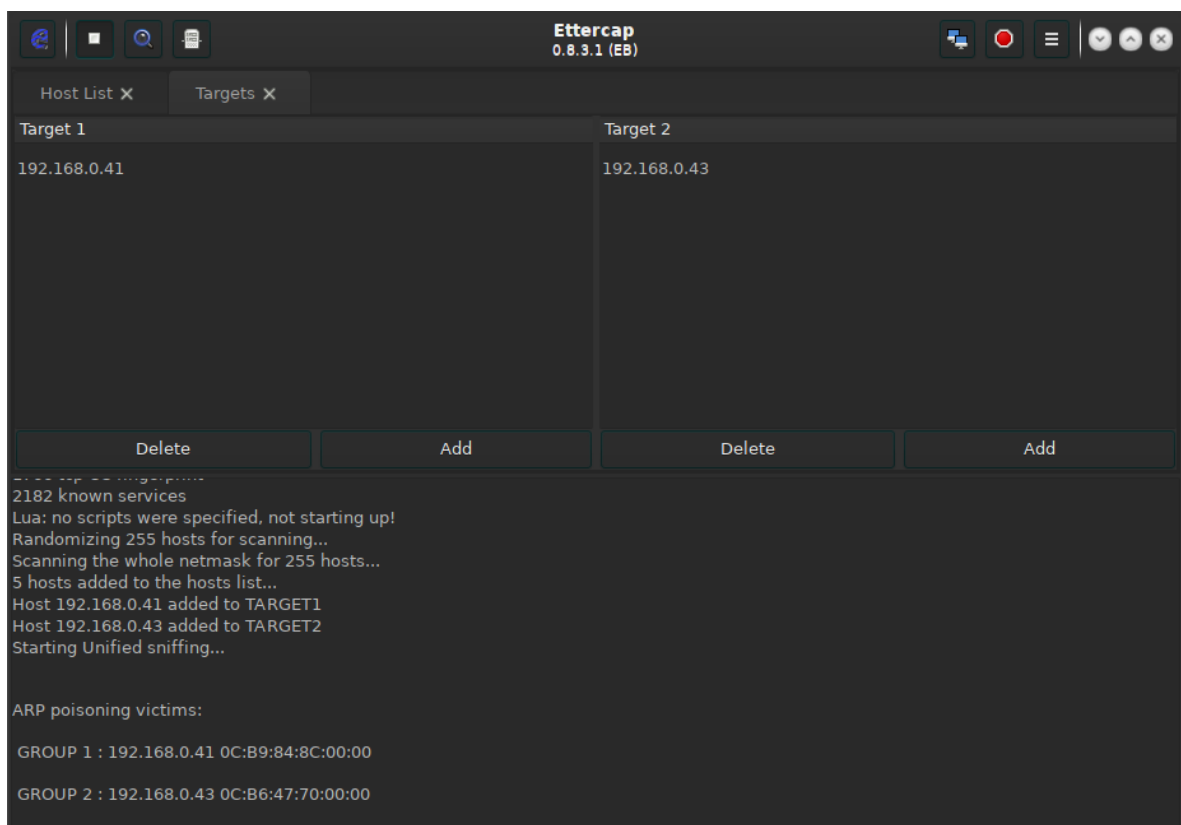


Figure A.6: Ettercap poisoning targets ARP cache

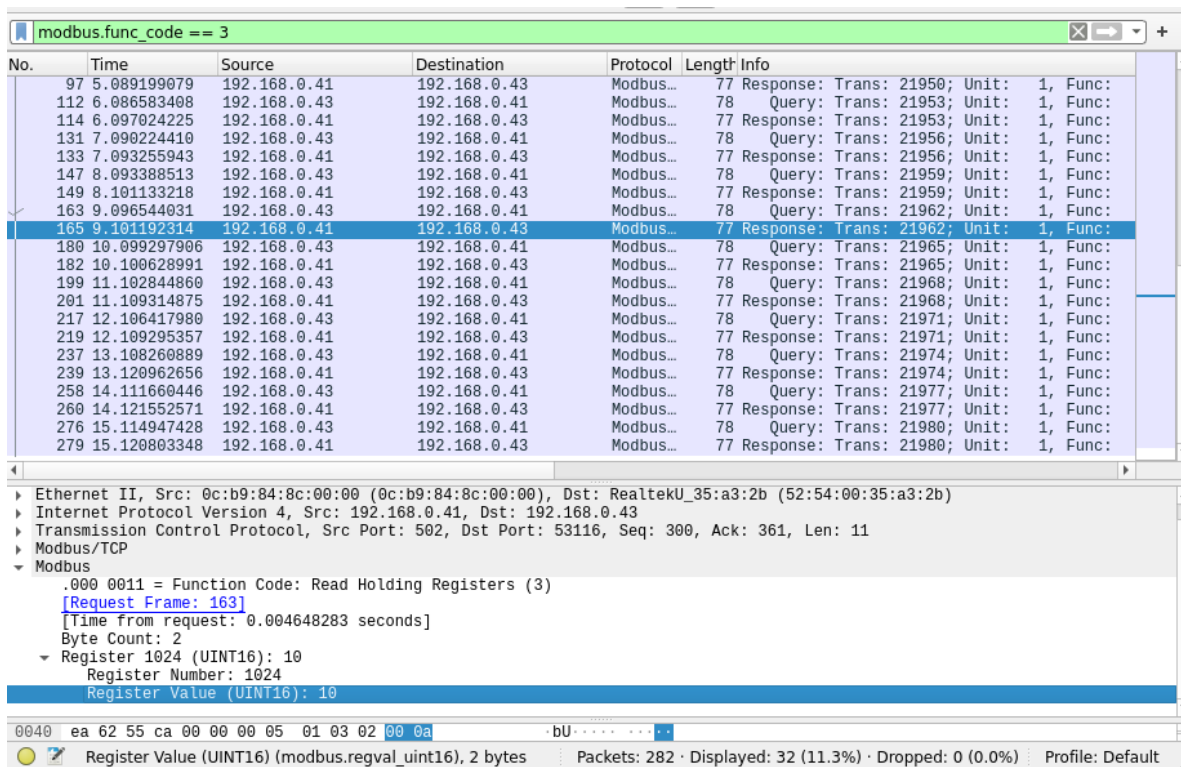
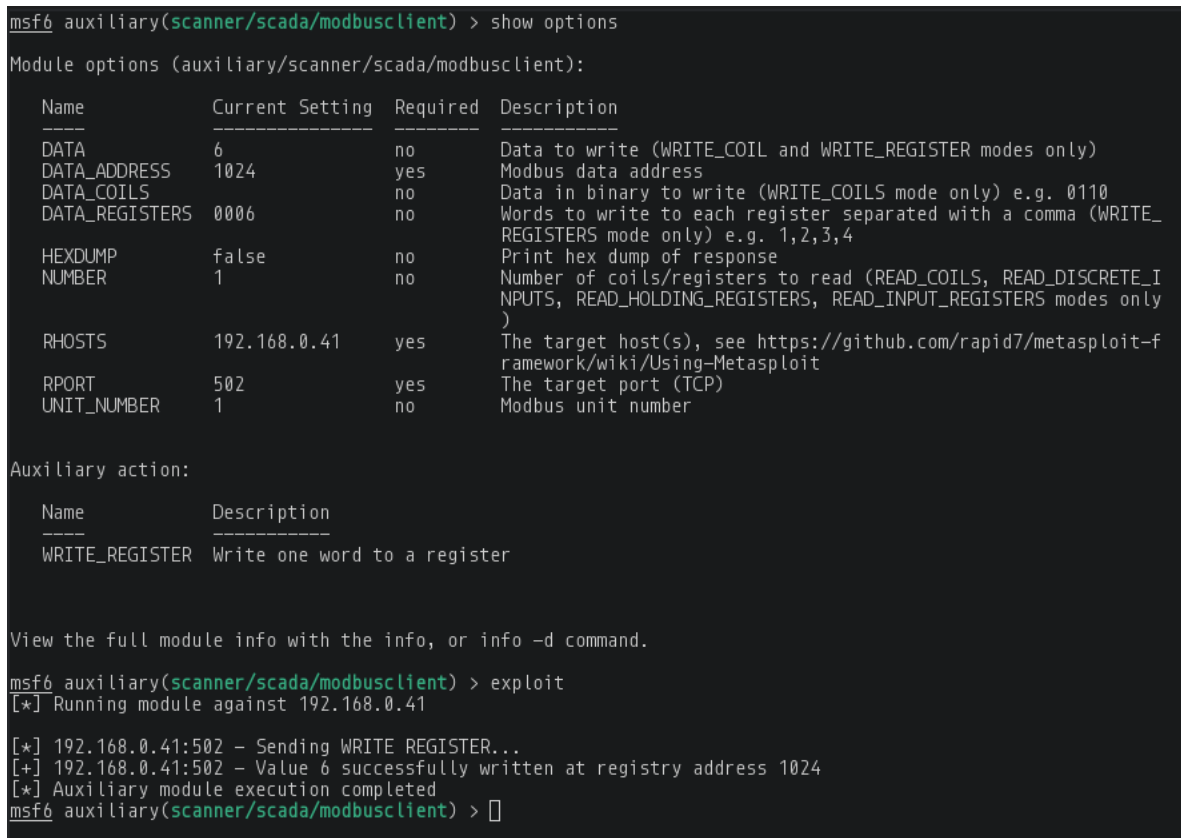


Figure A.7: Wireshark showing filtered SCADA-PLC traffic

Figure A.8: Metasploit *modbusclient* module in action

Monitoring

Refresh Rate (ms): Update








Point Name	Type	Location	Forced	Value
TANK_LLS	BOOL	%QX100.4	No	 FALSE
TANK_HLS	BOOL	%QX100.5	No	 FALSE
STOP	BOOL	%QX100.6	No	 FALSE
PUMP_A	BOOL	%QX100.0	No	 FALSE
PUMP_B	BOOL	%QX100.1	No	 FALSE
MIXER	BOOL	%QX100.2	No	 FALSE
VALVE	BOOL	%QX100.3	No	 FALSE
IntTime	INT	%MW0	No	<div><div>6</div></div>

Figure A.9: OpenPLC device monitoring screen after attack

configure the switch with proper access-control lists (ACL) and virtual LAN (VLANs) or introduce multiple switches to provide network segmentation in the network topology. Testing various strategies will be helpful in devising a custom and industry-specific defence strategy for an ICS.

The reason we were able to do a MITM attack on this topology is because of an unmanaged switch. One simple solution would be to add another switch to the topology and keep the industrial zone and the control zone separated by different creating different subnets. However, this simple approach increases the complexity of the system. Introducing multiple switches can cause delays and domain collisions. A proper defence in a large scale network is to create VLANs with proper ACLs. This also makes adding a separate firewall easier because of only one uplink.

VLAN stands for Virtual LAN which is defined by IEEE 802.1Q Tagging Protocol. VLANs separate a LAN logically or virtually by adding a 4-byte tag to every Ethernet frame sent over the network (IEEE 1999). This makes it easier to split a single physical switch based on the operational and security requirements of the system without introducing downtime and unnecessary complexity. This also improves the overall performance of the network by reducing the collision domains and grouping devices together. It also reduces broadcast traffic which makes it easier to manage broadcast domains.

The aim of this exercise is to plan a defence strategy of hardening the switch by managing it properly. All the modern switch manufacturers provide the option to configure VLANs. The maximum number of VLANs that can be created is defined by the manufacturer and is often limited by the device type. OpenVswitch also allows the creation of not only VLANs, but also VXLANs defined by RFC 7348⁵.

Fortunately, we did not have to configure openVswitch right from the start because it is provided as a pre-configured GNS3 appliance. However, by default, it connects all the hosts through ports attached to the bridge br0.

Open the terminal to the OpenVswitch instance from GNS3 and run the following command to inspect the bridge br0.

⁵<https://datatracker.ietf.org/doc/html/rfc7348>

```
| ovs-vsctl show
```

The output of the command is long but we are looking at port-interface mapping for the bridge.

```
| ...  
Bridge br0  
    datapath_type: netdev  
    Port br0  
        Interface br0  
            type: internal  
    Port eth3  
        Interface eth3  
    Port eth7  
        Interface eth7  
    Port eth15  
| ...
```

In other words, none of our ports are 'VLAN-tagged' right now. So we will create separation between Industrial Zone and Control Zone (see figure A.1).

Issue the following command from openVswitch console to tag its port eth1 with 10. You can hover over the icon for openVswitch in GNS3 also to find out which port is connected where.

Here, eth1 is connected to the machine 'OpenPLC'.

```
| ovs-vsctl set port eth1 tag=10
```

Now, any other port which we set with a tag of '10' will be able to communicate with each other. You can already see VLANs at work by pinging OpenPLC machine from Scada-LTS machine. Now it is unable to ping it anymore. Since we want the machines Scada-LTS and Firefox to communicate with each other and the PLC, we will tag them with 10 as well.

```
| ovs-vsctl set port eth2 tag=10
```

```
| ovs-vsctl set port eth3 tag=10
```

Verify that they can communicate again by pinging each other.

Similarly, we create a new VLAN with tag '20' in the industrial zone.

```
| ovs-vsctl set port eth1 tag=20
```

```
| ovs-vsctl set port eth2 tag=20
```

Verify that operator workstation cannot ping machines in the control zone.

That's it! Now we have created two VLANs for separate zones. However, this setup might be too rigid for some purposes. What if one of the engineering workstation in the industrial zone wants to connect to the PLC for maintenance/development purposes?

For this, we have to create a special link between the two switches i.e. a trunking interface.

Issue the following commands on both openVswitch instances if you want to allow communication between the two zones using a VLAN trunk:

```
| # For openvswitch-1  
| ovs-vsctl set port eth4 trunks=10,20  
  
| # For openvswitch-2  
| ovs-vsctl set port eth0 trunks=10,20
```

Now we can ping the machines in the industrial zone and the control zone. This tutorial shows that there is no one true strategy for defending critical infrastructure in the OT. Strategies have to evolve based on changes and requirements of the industry. Hence, a simulation provides an excellent playground for testing various defence strategies for an ICS.



Research Code

B.1 PLC Ladder Logic

```
PROGRAM OB1
  VAR
    TANK_LLS AT %QX100.4 : BOOL;
    TANK_HLS AT %QX100.5 : BOOL;
    STOP AT %QX100.6 : BOOL;
  END_VAR
  VAR
    ELAPSED_TIME : TIME;
  END_VAR
  VAR
    PUMP_A AT %QX100.0 : BOOL;
    PUMP_B AT %QX100.1 : BOOL;
    MIXER AT %QX100.2 : BOOL;
    VALVE AT %QX100.3 : BOOL;
  END_VAR
  VAR
    SR0 : SR;
    TP0 : TP;
    SR1 : SR;
  END_VAR
  VAR
```

```
    IntTime AT %MW0 : INT := 5;
END_VAR
VAR
    INT_TO_TIME44_OUT : TIME;
END_VAR

SR0(S1 := TANK_LLS, R := NOT(STOP) OR TANK_HLS);
PUMP_A := SR0.Q1;
PUMP_B := SR0.Q1;
INT_TO_TIME44_OUT := INT_TO_TIME(IntTime);
TP0(IN := NOT(VALUE) AND TANK_HLS, PT := INT_TO_TIME44_OUT);
MIXER := TP0.Q;
ELAPSED_TIME := TP0.ET;
SR1(S1 := NOT(MIXER) AND TANK_HLS, R := NOT(STOP) OR TANK_LLS);
VALVE := SR1.Q1;
END_PROGRAM

CONFIGURATION Config0

    RESOURCE Res0 ON PLC
        TASK task0(INTERVAL := T#20ms,PRIORITY := 0);
        PROGRAM instance0 WITH task0 : OB1;
    END_RESOURCE
END_CONFIGURATION
```

B.2 VyOS Configuration Script

```
#!/bin/vbash

source /opt/vyatta/etc/functions/script-template

configure

set interfaces ethernet eth0 address dhcp
set interfaces ethernet eth0 description 'external'
set interfaces ethernet eth1 address '192.168.0.1/24'
set interfaces ethernet eth1 description 'internal'

set service ssh

set service dhcp-server shared-network-name LAN subnet 192.168.0.0/24
→ default-router '192.168.0.1'
set service dhcp-server shared-network-name LAN subnet 192.168.0.0/24
→ lease '86400'
set service dhcp-server shared-network-name LAN subnet 192.168.0.0/24
→ range 0 start 192.168.0.9
set service dhcp-server shared-network-name LAN subnet 192.168.0.0/24
→ range 0 stop '192.168.0.254'

set service dhcp-server shared-network-name LAN subnet 192.168.0.0/24
→ domain-name 'vyos.net'
set service dhcp-server shared-network-name LAN subnet 192.168.0.0/24
→ name-server '192.168.0.1'

set service dns forwarding cache-size '0'
set service dns forwarding listen-address '192.168.0.1'
set service dns forwarding allow-from '192.168.0.0/24'

set nat source rule 100 outbound-interface 'eth0'
set nat source rule 100 source address '192.168.0.0/24'
set nat source rule 100 translation address masquerade

commit
exit
```


B.3 Scada-LTS Docker compose file

```
version: '3'
services:
  database:
    container_name: mysql
    image: mysql/mysql-server:5.7
    ports:
      - "3306:3306"
    environment:
      - MYSQL_ROOT_PASSWORD=root
      - MYSQL_USER=root
      - MYSQL_PASSWORD=root
      - MYSQL_DATABASE=scadalts
    expose: ["3306"]
    volumes:
      - ./databases:/var/lib/mysql
  scadalts:
    image: scadalts/scadalts:latest
    ports:
      - "8080:8080"
    depends_on:
      - database
    expose: ["8080", "8000"]
    links:
      - database:database
```