This unit builds on the GNS3 setup from U18.

21.1 At your (Web-)Service

After setting up the internals of the company network, you decide to get started with the web server. To save a little bit of money you decided to set up an old machine as the web server inside the office network.

- Adjust your address plan and gateway-configuration for an additional subnet you will be using for the web server.

Solution:

Department	VLAN	Network	Gateway Address
Engineering	10	10.0.10.0/24	10.0.10.1
		fdae:889e:ec80:a::/64	fdae:889e:ec80:a::1
Accounting	20	10.0.20.0/24	10.0.20.1
		fdae:889e:ec80:14::/64	fdae:889e:ec80:14::1
Servers	30	10.0.30.0/24	10.0.30.1
		fdae:889e:ec80:1e::/64	fdae:889e:ec80:1e::1

- Place a *www*-node into the subnet and configure its network parameters using a static IP address. (You can access the interfaces file from the node configuration dialog)

Solution:

```
# /etc/network/interfaces

auto eth0
iface eth0 inet static
  address 10.0.30.2
  netmask 255.255.255.0
  gateway 10.0.30.1

iface eth0 inet6 static
  address fdae:889e:ec80:1e::2
  netmask 64
  gateway fdae:889e:ec80:1e::1
```

- Configure your DNS server to resolve the name `www.startup.local.` to the web server's IP address

Solution: Edit `/etc/dnsmasq.conf` and add a line for the webserver.

```
host-record=www.startup.local,10.0.30.2,fdae:889e:ec80:1e::2
```

As an alternative you can also add the required entries to the `/etc/hosts` file or configure a dedicated zone file.

- Verify your setup by trying to access the web server from each of the department networks.

Solution: Reachability can be checked using the browser / curl on the department workstations.

To thoroughly check your setup you need to check:

- IPv4 connectivity (`ping`)
- IPv6 connectivity (`ping6`)
- DNS resolution (e.g. `ping` / `ping6`, `dig` or `nslookup`)

21.2 Securing the Network

Now that you setup the web server it is time to think more about security. Before we open up our network to the outside world, we have to configure some firewall rules.

Design a firewall to be installed on `gateway` realizing the following policy. Use a **whitelisting** approach.

- Services running on `gateway` should not be accessible from the Internet
- Everyone should be able to browse the web site.
- Each department network should be able to to access the other.
- The department networks should be able to access the internet without restrictions.
- The server network should not be able to access the department networks (but still the departments should be able to access the web server).
- The servers should only be able to establish connections to the Internet using *HTTP* and *HTTPS* (ports 80/tcp and 443/tcp)
- ICMP messages required for proper IP operation should not be blocked by the firewall. All other ICMP message types should be blocked.

Solution: The following commands configure a possible **IPv4** firewall to enforce the given policy:

```
# Clear all existing rules
iptables -F

# Remove any pre-existing custom chains
iptables -X

# Set Default Policies
iptables -P INPUT DROP
iptables -P OUTPUT ACCEPT
iptables -P FORWARD DROP


### Define a custom chain for sensible ICMP traffic
# This way we can use the same rules from the INPUT and FROWARD chain
```

```
# (for typenames issue: iptables -p icmp -h)
iptables -N SENSIBLE_ICMP
iptables -A SENSIBLE_ICMP -p icmp --icmp-type echo-request -j ACCEPT
iptables -A SENSIBLE_ICMP -p icmp --icmp-type echo-reply -j ACCEPT
iptables -A SENSIBLE_ICMP -p icmp --icmp-type destination-unreachable -j ACCEPT
iptables -A SENSIBLE_ICMP -p icmp --icmp-type ttl-exceeded -j ACCEPT
iptables -A SENSIBLE_ICMP -p icmp --icmp-type parameter-problem -j ACCEPT

# Verbosely reject any other ICMP traffic
iptables -A SENSIBLE_ICMP -j REJECT --reject-with icmp-admin-prohibited


###### INPUT ############################################################

# Filter ICMP traffic
iptables -A INPUT -p icmp -j SENSIBLE_ICMP

# Allow Traffic for already established connections
iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT

# Allow DHCP for the _local_ subnets only
iptables -A INPUT -i eth1.10 -p udp --dport 67 -m state --state NEW -j ACCEPT
iptables -A INPUT -i eth1.20 -p udp --dport 67 -m state --state NEW -j ACCEPT

# Allow access to DNS for the _local_ subnets only
iptables -A INPUT -i eth1.10 -p udp --dport 53 -m state --state NEW -j ACCEPT
iptables -A INPUT -i eth1.20 -p udp --dport 53 -m state --state NEW -j ACCEPT


###### FORWARD ##########################################################

# Filter ICMP traffic
iptables -A FORWARD -p icmp -j SENSIBLE_ICMP

# Allow everyone to access the web server using HTTP
iptables -A FORWARD -d 10.0.30.2 -p tcp --dport 80 -m state --state NEW -j ACCEPT

# Allow Inter-Department Traffic
iptables -A FORWARD -i eth1.10 -o eth1.20 -j ACCEPT
iptables -A FORWARD -i eth1.20 -o eth1.10 -j ACCEPT

# Allow Traffic for already established connections
iptables -A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT

# Allow Internet Access from Departments
iptables -A FORWARD -i eth1.10 -o eth0 -m state --state NEW -j ACCEPT
iptables -A FORWARD -i eth1.20 -o eth0 -m state --state NEW -j ACCEPT

# Only allow HTTP/S connections from the server network to the internet
iptables -A FORWARD -i eth1.30 -o eth0 -p tcp --dport 80 -m state --state NEW \
  -j ACCEPT
iptables -A FORWARD -i eth1.30 -o eth0 -p tcp --dport 443 -m state --state NEW \
  -j ACCEPT
```

These commands will only configure the running kernel. To persist the iptables configuration you can e.g. make use of the *iptables-restore* package to configure the firewall at boot:

```
service iptables save
rc-update add iptables default
```

Caution: These firewall rules will only apply to IPv4 traffic. Any IPv6 traffic can still pass the firewall unhindered (because we did not configure it and the default is ACCEPT). You can setup an IPv6 firewall analogous to the IPv4 case using `ip6tables`.

21.3 Opening Up

The last step in building our network is enabling Internet access for the employees and making the web server available to the outside world.

Unfortunately our ISP does not supply us with a complete subnet (or even a static ip address), but just one publicly routable IP address which is chosen randomly every 24h.

In order to make the web server accessible to the outside world we have to configure our gateway to forward requests to the HTTP port to our web server.

- Enable Internet access for all departments as well as the server network (in accordance with the security policy).

Solution: Even though routing is enabled Internet access is not working out of the box. Since the ISP is not aware of the RFC1918 (i.e. non-routable) addresses on the inside of `gateway`, the gateway has to employ some form of network address translation (e.g. *Masquerading*).

In Linux NAT is also configured using the `iptables` command. You can enable masquerading by adding the following line to your firewall configuration.

```
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

- Setup iptables rules to forward and allow access to the web server from the Internet.

Solution: To forward HTTP packets addressed to the external interface of `gateway` to the internal webserver add the following line to your firewall configuration.

```
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j DNAT --to 10.0.30.2
```

Notice this rule only takes care of the *inbound* traffic. *Outbound* traffic (i.e. the replies) is transparently handled by the MASQUERADE rule from before.

- Verify your configuration (e.g. by attaching a *webterm*-node to the hub/switch between the gateway and the Internet).

Solution: Since we do not have a public name for our webserver, we can only test the forwarding using the external IP address of `gateway`. Since `gateway`'s external IP address is allocated by the ISP using DHCP, we cannot be sure what it is. To find the IP address of the external interface you can e.g. use the command `ip -4 address show dev eth0`.

21.4 Tunneling Through

Business is booming. A big company has approached you and your colleagues. They are asking you to help them with a big networking project. In order to be closer to the customer it is agreed that a few of the engineers will work in a new branch offce close to the new customer.

The new branch office will have to be connected to your main network. Since dedicated network connections are too expensive, you opt for using the available Internet connection by setting up some tunnels.

- Modify your address plan to accommodate the branch office (IPv4 and IPv6).

Solution:

Department	VLAN	Network	Gateway Address
Engineering	10	10.0.10.0/24	10.0.10.1
		fdae:889e:ec80:a::/64	fdae:889e:ec80:a::1
Accounting	20	10.0.20.0/24	10.0.20.1
		fdae:889e:ec80:14::/64	fdae:889e:ec80:14::1
Servers	30	10.0.30.0/24	10.0.30.1
		fdae:889e:ec80:1e::/64	fdae:889e:ec80:1e::1
Branch Office		10.0.99.0/24	10.0.99.1
		fdae:889e:ec80:ffff::/64	fdae:889e:ec80:ffff::1

- Setup a remote site with an independent gateway (hostname `branch`, connected to the Internet) and at least one workstation.

- Setup Internet connection for the workstation(s) at the remote site (DNS resolution, NAT, ...).

Solution: See above.

For the sake of simplicity we will set up an *unencrypted* tunnel, which is obviously *not secure*, but sufficient for this tutorial.

- Setup an ipip tunnel between `gateway` and `branch`.

Solution: You have to set up a tunnel interface on each of the gateways:

```
ip tunnel add tunnelv4 mode ipip remote <remote Internet IP>
ip link set up dev tunnelv4
```

This makes the kernel of both gateways aware that there is a tunnel. In order to transfer data using the tunnel, we have to tell the kernel where the tunnel leads (i.e. which networks lie on the other side).

If you check the routing table on `gateway` you see that it contains only the connected routes (the link-local networks) and the default gateway. It does not know about the remote network at the branch office:

```
# ip route show
default via 10.0.2.2 dev eth0  metric 202
10.0.2.0/24 dev eth0  proto kernel  scope link  src 10.0.2.16
10.0.10.0/24 dev eth1.10  proto kernel  scope link  src 10.0.10.1
10.0.20.0/24 dev eth1.20  proto kernel  scope link  src 10.0.20.1
10.0.30.0/24 dev eth1.30  proto kernel  scope link  src 10.0.30.1
```

Therefore packets destined to 10.0.99.0/24 will be routed to the default gateway and not to the branch office.

```
# ip route get 10.0.99.1
10.0.99.1 via 10.0.2.2 dev eth0  src 10.0.2.16
  cache
```

We can tell the kernel to route packets through the ipip tunnel by installing a now route:

```
# ip route add 10.0.99.0/24 dev tunnelv4
```

The kernel will then route packets to 10.0.99.0/24 through the ipip tunnel:

```
# ip route get 10.0.99.1
10.0.99.1 dev tunnelv4  src 10.0.2.16
  cache
```

Same goes for the routing table at the branch office. We can aggregate the subnets at the main office, so we only have to add one entry to the routing table:

```
# ip route add 10.0.0.0/16 dev tunnelv4
```

This will enable routing between the two offices using the tunnel. You can verify this by pinging a workstation in the branch office from one of the department workstations in the main office. You can also verify packets are encapsulated using wireshark.

- Setup a 6in4 tunnel between `gateway` and `branch`.

Solution: The process for IPv6 is analogous to the IPv4 tunnel:

```
# ip tunnel add tunnelv6 mode sit remote <remote Internet IP>
# ip link set up dev tunnelv6
```

On `gateway`:

```
# ip route add fdae:889e:ec80:ffff::/64 dev tunnelv6
```

On `branch`:

```
# ip route add fdae:889e:ec80::/48 dev tunnelv6
```

- Update your firewall configuration on `gateway` to allow tunnel traffic through the firewall if necessary.

Solution: Tunnel packets can be identified using their IP protocol number. You can use either the raw protocol number (ip encapsulation uses protocol number 4, 6in4 uses the protocol number for IPv6 which is 41) or the corresponding protocol name (see `/etc/protocols` for the mapping).

We can allow tunnel packets to the gateways by adding the following rules to the firewall:

```
iptables -A INPUT -i eth0 -p ipencap -j ACCEPT
iptables -A INPUT -i eth0 -p ipv6 -j ACCEPT
```

- **Challenge Task:** Make sure you can ping workstations at the branch office from `gateway`, workstations at the main office from `branch`, as well as the internal addresses of the respective remote gateway.

Solution: The kernel does its best to automatically determine the outgoing interface and source address for a packet to be sent. Even though this generally works well in most cases, it will fail for certain unusual constellations. Our tunnel setup is one of those constellations.

If you pay close attention to the kernel's routing decision you can see that the kernel chooses the Internet-facing IP address as source address when transmitting data (e.g. an ICMP echo-request) from `gateway` to the branch office:

```
# ip route get 10.0.99.1
10.0.99.1 via 10.0.2.2 dev eth0  src 10.0.2.16
  cache
```

Alternatively you can take a closer look at the packets using Wireshark.

Since the source of the inner and outer IP header of the tunnel packet are the same, the kernel will consider them invalid and drop them.

To mitigate this problem we can assign a fixed (internal) source address to the tunnel routes. We can update the routes we installed accordingly.

On `gateway`:

```
# ip route change 10.0.99.0/24 dev tunnelv4 src 10.0.10.1
# ip route change fdae:889e:ec80:ffff::/64 dev tunnelv6 src fdae:889e:ec80:a::1
```

On `branch`:

```
# ip route change 10.0.0.0/16 dev tunnelv4 src 10.0.99.1
# ip route add fdae:889e:ec80::/48 dev tunnelv6 src fdae:889e:ec80:ffff::1
```

Hints:

- Don't worry about persisting the tunnel configuration for now. Since the outside connections of the office gateways use dynamic IP addresses, just set up a temporary tunnel on the shell.
- Tunnel interfaces do not necessarily require ip addresses.
- Pay special attention to the routing table.
  - Show the routing table: `ip route show`
  - You can get detailed information about the kernel's routing decision for a specific destination using the command `ip route get <dest. ip>`