Xaptum ENF Walkthrough

"ACME Occupancy Monitoring Systems"

It's May 2020 and your small technology company, ACME Occupancy Monitoring Systems, has been inundated with inquiries. The first wave of the COVID-19 pandemic has started to subside and employers are eager to resume work, while still complying with necessary social distancing mandates. Although initially designed for utilization tracking and resource planning purposes, ACME Room Occupancy Monitors seem to be the perfect tool for enforcing these social distancing requirements: install an occupancy sensor in each room, configure occupancy limits, and automatically notify supervisors when workers exceed the limit. The non-stop ringing of your sales phone confirms that your customers agree. ACME just might be the next tech unicorn!

There's one big problem though. Many of your customers' IT departments are pushing back, concerned about security. They don't want your occupancy sensors connected to their networks and absolutely refuse to open ports in the firewall to allow communication with the cloud-hosted ACME dashboard. Some doubts start to creep into your own mind. Your engineers are experts in occupancy sensing, not software and network security. You try to follow security guidelines and best practices, but is that really enough? Even the best experts make mistakes, and it only takes one buffer overflow or one misconfiguration for a hacker to gain access. Eager to save these deals and protect your reputation, you turn to Xaptum for help.

Xaptum's ENF overlay network segments your sensors from the corporate network traffic and public Internet, satisfying the IT departments' security concerns. Traffic is tunneled through the standard outbound TLS sessions on port 443, so no firewall changes are required. You can rest assured that any security vulnerabilities in the ACME system aren't visible to be exploited by hackers. And best of all, it requires minimal changes to your sensor firmware and cloud-hosted dashboard, so integration and testing takes only a couple of days.

Overview

In this walkthrough, you'll integrate the Xaptum ENF into a (mock) occupancy monitoring system, complete with MQTT broker, cloud-hosted dashboard, and (virtual) occupancy sensors.

Use Case: Deploy room occupancy sensors to count the number of people in rooms to enforce COVID-19 occupancy limits.

Setup:

- One sensor deployed in each room, connected to enterprise LAN via ethernet or WiFi.
- Cloud-hosted dashboard used to set occupancy limits and view current counts.
- Sensors and dashboard communicate using MQTT through a cloud-hosted broker.

Constraints:

- Enterprise IT department refuses to open new ports in firewall (e.g., 1883 for MQTT).
- All traffic (MQTT, etc.) must be segmented from the physical network (LAN and Internet).

After completing the walkthrough, we kindly ask that you complete the survey questions to help us better understand your experience, any difficulties you faced, and thoughts on the ENF product. Thanks in advance!

Walkthrough

Prerequisites

To get complete this walkthrough, you will need:

1. A demo account on the Xaptum ENF. Register for one at https://www.xaptum.com/signup.
2. An account with a cloud provider like AWS or Azure. The AWS EC2 introductory "Free Tier" is sufficient for this walkthrough.
3. A computer with the following software installed:
    a. Docker
    b. Git
    c. Ruby
    d. enfcli. To install, run `gem install enfcli`.

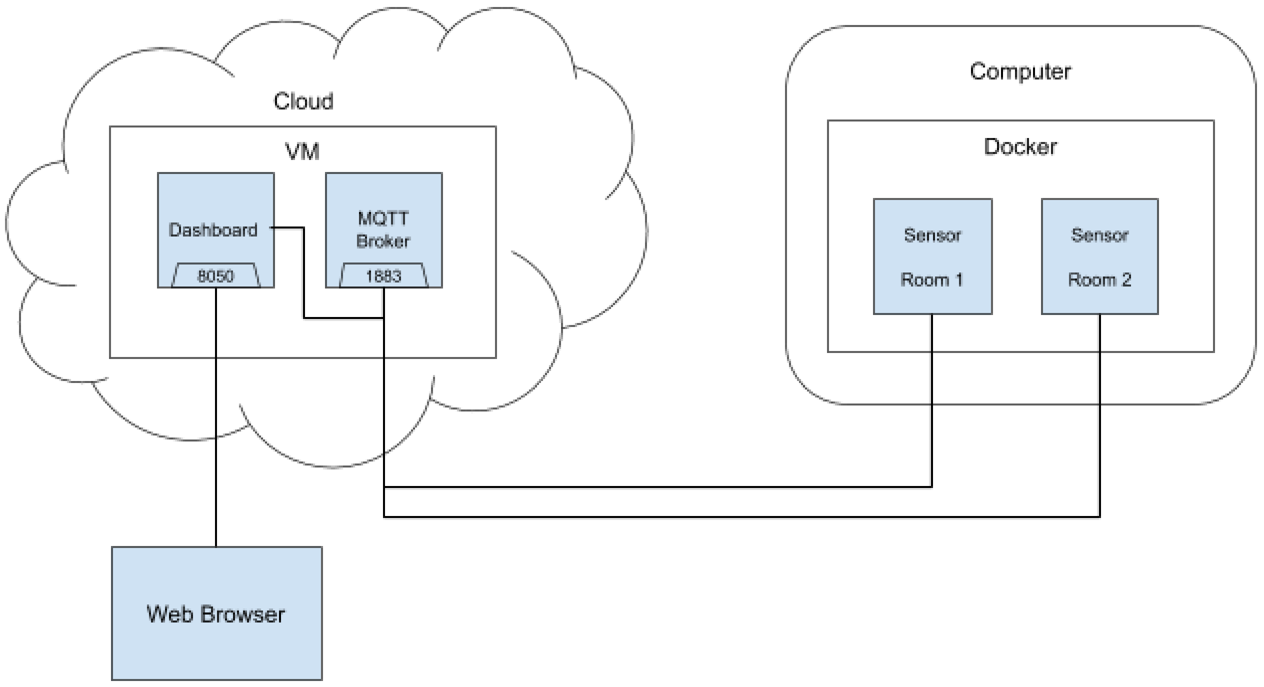
Please register for these accounts and install the required software on your computer before proceeding.

Before proceeding, it is highly recommended to do the Xaptum *Getting Started* tutorial at https://docs.xaptum.com/getting_started. This will teach you the basics of using the ENF and make completing this walkthrough much easier.

Occupancy Monitoring Without the ENF

We will first install and run the ACME occupancy monitoring system without using the ENF.

The system consists of three components: (virtual) occupancy sensors, an MQTT broker, and web-hosted dashboard. For the walkthrough, the sensor is a Docker container displaying buttons to "enter" or "leave" a room.



All components are available via Git at *https://github.com/xaptum/acme-occupancy*.

Install the Broker and Dashboard

1. Launch a Debian 9 “Stretch” virtual machine with public Internet access on the cloud provider of your choice.

   For AWS EC2, the suggested AMI is `debian-stretch-hvm-x86_64-gp2-2020-10-31-2842` with a public IP assigned.

2. Configure your cloud firewall (ACL, security group, etc.) to allow inbound TCP traffic on ports 8050 and 1883 (and 22 for SSH).

   Port 8050 will serve the dashboard and port 1883 will serve the MQTT broker.

   Note that we are not yet using the ENF, so these ports are publicly exposed. Consider opening them only to your computer’s IP address, not the entire Internet. After enabling the ENF, no ports will need to be open.

3. On the VM, install the `git` client and clone the ACME occupancy system source code:

   ```
   git clone https://github.com/xaptum/acme-occupancy.git
   ```

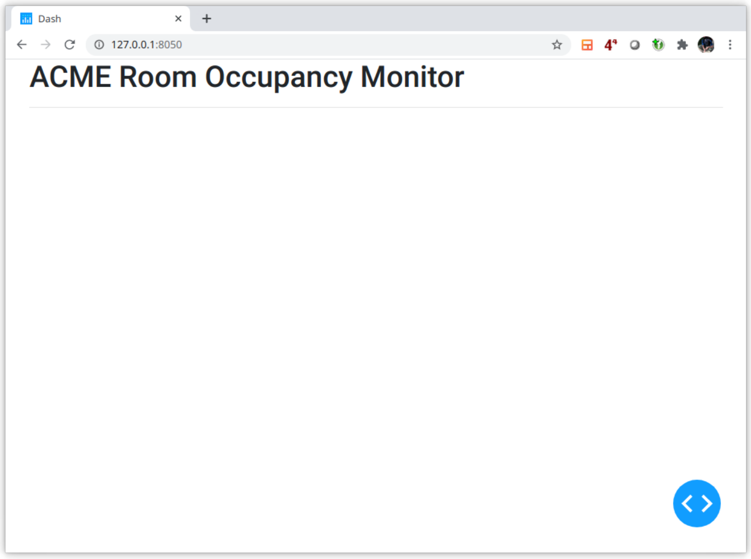
4. Start the MQTT broker in the `broker` directory:

```
cd acme-occupancy/broker
python3 broker.py
```

5. Start the dashboard in the `dashboard` directory, pointing at the broker instance on the same machine:

```
cd acme-occupancy/dashboard
python3 dashboard.py --broker localhost:1883
```

6. Open a web browser on your computer to view the dashboard at `http://<public-ip-of-your-VM>:8050`. You should see an empty dashboard, since there aren't any room sensors deployed yet.



ACME Room Occupancy Monitor

Install a Sensor

7. On your local computer with Docker installed, clone ACME occupancy system source code again.

```
git clone https://github.com/xaptum/acme-occupancy.git
```

8. Build the sensor Docker image in the `sensor` directory:

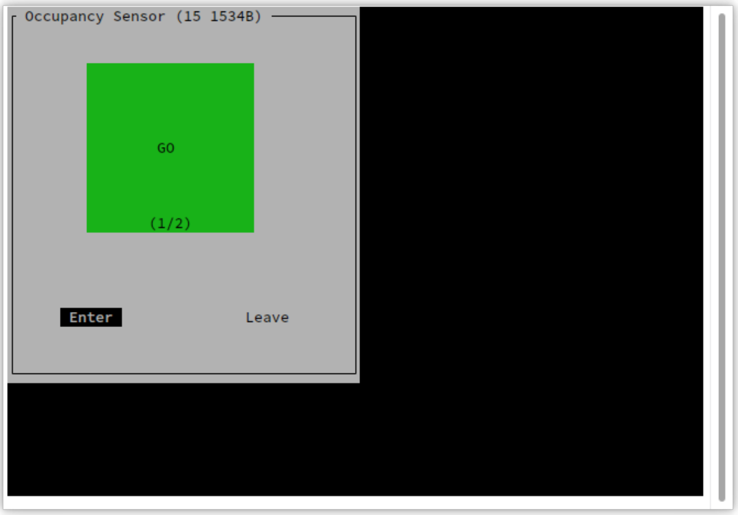
```
cd acme-occupancy/sensor
docker build -t sensor .
```

9. Create a sensor by running the Docker image, pointing it at the broker instance on your cloud VM:

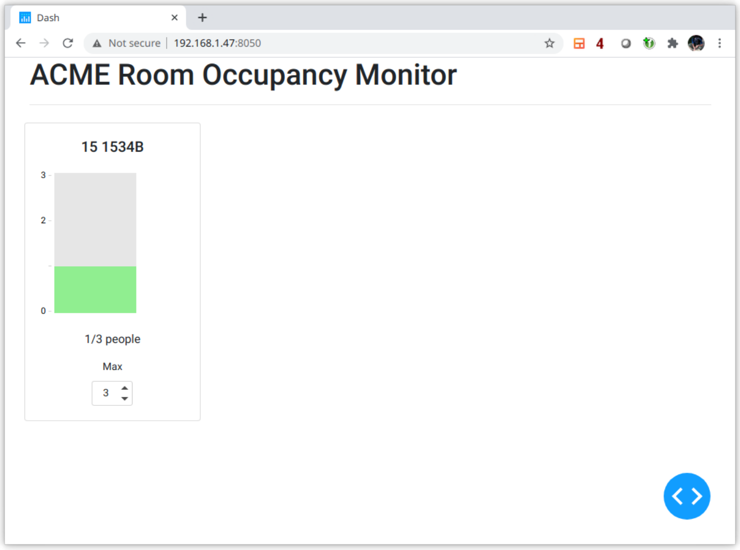
```
docker run -it sensor --broker <public-IP-of-your-VM> --floor 15
--room 1534B
```

The following Occupancy Sensor UI below will be displayed.

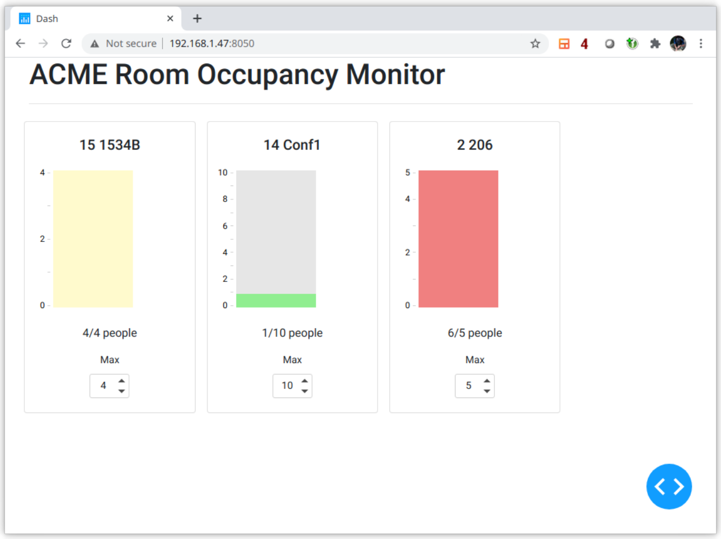
Use the Enter and Leave buttons (use the arrow keys to select and enter to press) to simulate people entering and leaving the room. If more people enter the room than are allowed, the green “Go” sign will change to a red “Wait” sign.



10. Refresh the dashboard to see the new room appear. Use the selector at the bottom to set the occupancy limit for the room.



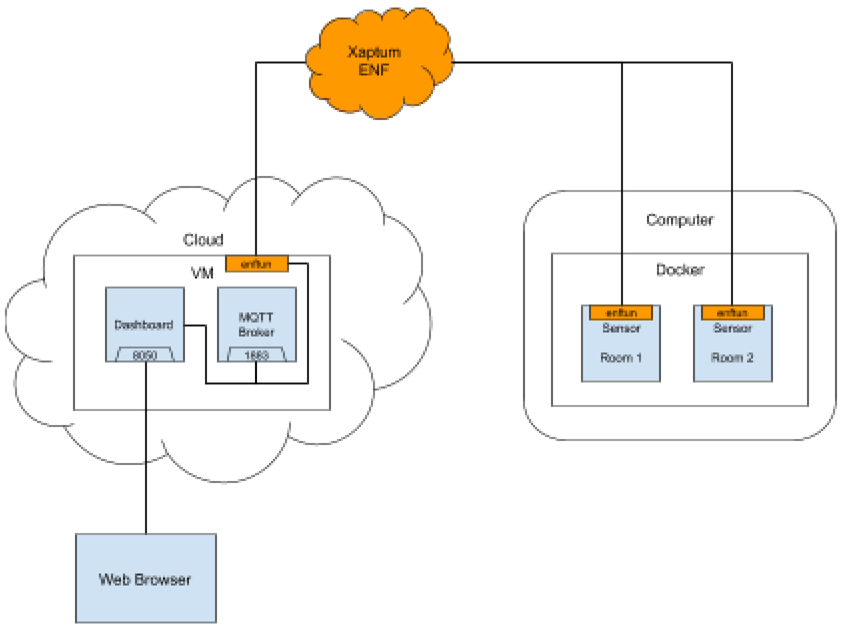
11. Play around with the system but adding new sensors, configuring limits, and simulating the movement of people. Observe that the dashboard updates in realtime.



Using the ENF to Secure the Occupancy Monitoring

The ACME monitoring system as deployed above functions well, but has some serious security flaws. The MQTT broker (port 1883) is publicly exposed (might as well put out a welcome mat for hackers) and the sensor is connected directly to the local network, upsetting the IT department.

Let's integrate the ENF into the ACME system to see how it fixes these issues. The dashboard will still be served over the public Internet, but the MQTT communication will happen securely through the ENF overlay. The sensors will be fully isolated from the local network and Internet.



Before proceeding, you are encouraged to read the *Getting Started* tutorial at https://docs.xaptum.com/getting_started to familiarize yourself with the basic usage of the ENF.

Connect Broker to the ENF

12. Install the `enftun` tunnel client on your cloud VM:

```
# Install the Xaptum API GPG signing key
sudo apt-get install dirmngr
sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys
c615bfaa7fe1b4ca

# Add the Xaptum APT repo
echo "deb http://dl.bintray.com/xaptum/deb stretch main" | sudo tee
/etc/apt/sources.list.d/xaptum.list
sudo apt-get update

# Install the software
sudo apt-get install enftun
```

13. Configure `enftun` by copying the example server configuration:

```
sudo cp /usr/share/doc/enftun/example/server.conf
/etc/enftun/enf0.conf
```

14. Create a directory for the access credentials:

```
sudo mkdir -p /etc/enftun/enf0
```

15. Provision an ENF IPv6 address and credentials for the server using the enftun-keygen utility. Specify an IPv6 address from your ENF account ::/64 network, visible on your https://demo.xaptum.io dashboard.

```
sudo enftun-keygen -c /etc/enftun/enf0.conf -u <ENF_USERNAME> -a
<ENF_IPv6_ADDR>
```

Provide your ENF account password when prompted.

This command creates a key pair and certificate in `/etc/enftun/enf0/`, registers a new endpoint with specified IPv6 address and certificate.

16. Enable and start `enftun`:

```
sudo systemctl enable enftun@enf0
sudo systemctl start enftun@enf0
```

Your VM should now have a virtual enf0 network interface with the specified IPv6 address.

Reconfigure the Firewalls

Now that the MQTT traffic will be securely routed over the ENF overlay, it is no longer necessary to open port 1883 on the cloud firewall. However, it does need to be allowed within the overlay.

17. Delete the firewall rules allowing inbound TCP traffic on port 1883 from your cloud (ACLs, security groups, etc.).

18. Configure the ENF firewall using the `enfcli` to allow the TCP port 1883 traffic between your server and any sensor IPs in the ::/64. The ENF firewall is stateless and needs to be configured for both ends (the server and the sensor), so four rules are required in total.

```
enfcli --host demo.xaptum.io --user <ENF_USERNAME>

> firewall add-firewall-rule --priority=200 --action=ACCEPT --
protocol=TCP --direction=INGRESS --dest-ip=<ENF_IPV6_ADDR>
--dest-port=1883 --source-ip=<<ENF_NETWORK>>::/64

> firewall add-firewall-rule --priority=200 --action=ACCEPT --
protocol=TCP --direction=EGRESS --dest-ip=<ENF_IPV6_ADDR>
--dest-port=1883 --source-ip=<<ENF_NETWORK>>::/64

> firewall add-firewall-rule --priority=200 --action=ACCEPT --
protocol=TCP --direction=INGRESS --source-ip=<ENF_IPV6_ADDR>
--source-port=1883 --dest-ip=<<ENF_NETWORK>>::/64

> firewall add-firewall-rule --priority=200 --action=ACCEPT --
protocol=TCP --direction=EGRESS --source-ip=<ENF_IPV6_ADDR>
--source-port=1883 --dest-ip=<<ENF_NETWORK>>::/64
```

Connect Sensors to the ENF

Connecting the sensors (Docker containers) to the ENF works the same way as the server, simply install `enftun`. To save you some typing, the `sensor-enf` directory contains a new Dockerfile that adds `enftun` to the *sensor* image.

19. Build the *sensor-enf* Docker image in the `sensor-enf` directory:

```
cd acme-occupancy/sensor-enf
docker build -t sensor-enf .
```

20. Provision an address and credentials using the enftun-keygen utility bundled inside of the Docker image. Unlike the server, the specific IPv6 doesn't need to be memorable, so we'll let the ENF automatically generate one. (The `docker-run-enftun-keygen.sh` script is a wrapper around `docker run` with the correct arguments specified).

    ```
    ./docker-run-enftun-keygen -u <ENF_USERNAME>
    ```

    Provide your ENF account password when prompted.

21. The Docker daemon does not enable IPv6 support by default, so add the following options to the Docker daemon configt. On Linux, the config is in `/etc/docker/daemon.json`. On MacOS, change it via Docker `Preferences->Daemon->Advanced`. Restart the daemon for the changes to take effect.

    ```
    "ipv6" : true
    "fixed-cidr-v6" : "fd00:d0c::/64"
    ```

22. Create a sensor by running the Docker image, pointing it at the ENF IPv6 of your cloud VM:

    ```
    ./docker-run-enftun --broker <ENF_IPV6_ADDR> --floor 15 --room 1534B
    ```

    Once again, the Occupancy Sensor UI below will be displayed.

23. Refresh the dashboard and once again play around with the system, adding sensors, setting occupancy limits, and simulating the movement of occupants. The sensors and MQTT traffic are nowly fully isolated in the ENF overlay network, protected from external threats.

Exercise Left to the Reader

Having to point the sensors directly at the IPv6 address of the broker isn't very nice. It's a pain to remember a specific address and what happens if you want to migrate the broker to a different machine? It'd be nice to use a domain name. The ENF provides a built-in private DNS service for just this situation.

The following steps are left as exercise for the reader. First completing the "Setup Private DNS" section of the *Getting Started* tutorial is highly recommended:

https://docs.xaptum.com/getting_started/#set-up-private-dns

24. Create a private DNS zone and server using the ENF cli.
25. Add a record for the broker's ENF IPv6 address.
26. Restart the sensor Docker container, pointing it at the DNS name for the broker.

Survey

Thanks for completing the ENF walkthrough! We greatly appreciate your responses to the following survey questions to help us improve the product.

Please fill out the survey at https://forms.gle/vVHTrAWY9FoGbwHb8.

A copy of the questions is included below for your convenience.

1. Rate the difficulty of dealing with the following challenges for edge computing and IoT.

	Easy				Difficult
Security	1	2	3	4	5
Scalability	1	2	3	4	5
Deployment	1	2	3	4	5
Maintenance	1	2	3	4	5

2. In your opinion, how well do existing IoT platforms handle the following challenges for edge computing and IoT?

	Poorly				Very Well
Security	1	2	3	4	5
Scalability	1	2	3	4	5
Deployment	1	2	3	4	5
Maintenance	1	2	3	4	5

3. Please describe the biggest problem you have faced with an IoT deployment and how you solved it.

4. In your opinion, how well does the ENF handle the following challenges for edge computing and IoT?

	Poorly				Very Well
Security	1	2	3	4	5
Scalability	1	2	3	4	5
Deployment	1	2	3	4	5
Maintenance	1	2	3	4	5

5. Rate the ease of use of the ENF relative to other IoT platforms and services.

	Much Worse		Equally		Much Better
Security	1	2	3	4	5
Scalability	1	2	3	4	5
Deployment	1	2	3	4	5
Maintenance	1	2	3	4	5

6. (Optional) Tell us about the aspect of the ENF that is most interesting (good or bad) to you personally.

7. How was your experience using the ENF to securely deploy and manage room capacity monitoring devices in the mock exercise?

☐ Very easy
☐ Somewhat easy
☐ Neither easy nor difficult
☐ Somewhat difficult
☐ Very difficult

8. Approximately how many hours did you take to complete the exercise?

< 1	1-2	3+

9. How did your experience using the ENF compare with your first use of a cloud platform (for example, AWS, Azure, etc.)? If you've never used a cloud platform, skip this question.

☐ Much easier
☐ Somewhat easier
☐ About the same
☐ Somewhat harder
☐ Much harder

10. What industry do you work in?

11. What is your role or position?

12. Roughly how many years of experience in this or related industries do you have?

1-5	6-10	11-19	20+

13. How familiar are you with edge computing or IoT *in general*?

1 (not at all)	2	3	4	5 (extremely familiar)

14. How familiar are you with edge computing or IoT *security challenges and solutions* in general?

1 (not at all)	2	3	4	5 (extremely familiar)

15. About how many edge computing or IoT deployments have you helped develop or operate?

None	1-2	3-5	6-9	10+