InfraSI documentation

A documentation about an IT infrastructure.

Presentation

Authors

- Antoine de Barbarin
- Nicolas Moyon
- Sabrina Eloundou

In this project, we will describe and explain an IT solution for a small business setting up a private local network connected to internet, in which there are 5 devices:

> - a firewall,
> - a web server,
> - a backup server,
> - a linux client,
> - a Windows client.

The **web server** hosts a web documentation of the business's network solution, and it should be reachable from inside or outside the private local network.

The **backup server** needs to periodically save the web server's files so that it can be rolled back anytime to a previous pristine/stable/working version.

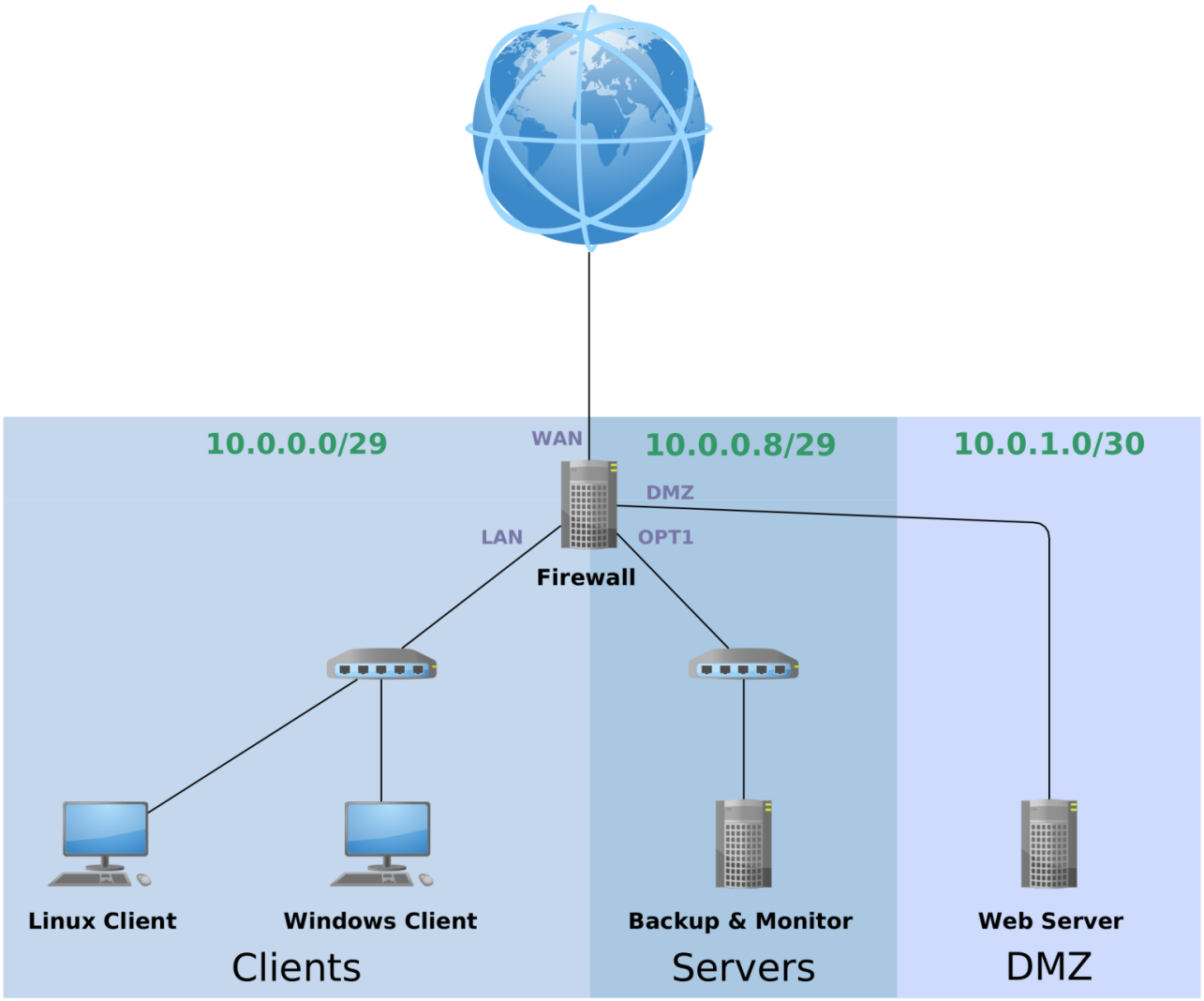
The **clients** are standard desktop workstations intended for general employee use within the corporate office environment. Both fulfill the same function but utilize distinct operating systems: Windows and Ubuntu.

IP addresses

devices \ networks	10.0.0.0 /29	10.0.0.8 /29	10.0.1.0 /30	Other
Firewall	**LAN** 10.0.0.1	**OPT1** 10.0.0.9	**DMZ** 10.0.1.1	**WAN** DHCP (from ISP)
Web Server	---	---	10.0.1.2	---

devices \ networks	10.0.0.0 /29	10.0.0.8 /29	10.0.1.0 /30	Other
Backup Server	---	10.0.0.10	---	---
Linux Workstation	DHCP (10.0.0.2)	---	---	---
Windows Workstation	DHCP (10.0.0.3)	---	---	---
{style="both"}				

Overview



There are few things in the graphic just shown:

- the firewall is the only device directly connected to the exterior, to internet
- the servers and client are on three different subnets inside the business's LAN
  - firewall's LAN interface for clients

    - firewall's OPT1 interface for backup server (and any other internal server that may be needed in the future)
    - firewall's DMZ interface for web server (extendable only when changing the subnet mask from /30 to a lower one)
- the three different sub-LANs are differentiated through the firewall's DHCP and the three physical LAN interfaces (LAN, OPT1 and DMZ on pfSense)
- each sub-LAN (except the DMZ) has a switch of its own, and it can be extended if necessary (up to 6 devices per subnet with the current subnet mask for LAN and OPT1 interfaces)
- the web server is a sensitive point in the current architecture, because of the access from outside the business's private network, that's why it is put in a DMZ, to isolate it from the rest of the network. Also, it will have no access to the other subnets and scheduled access to internet (for updating purposes). Others, however, may access it through SSH, in very defined rules present in the firewall: from the Linux workstation and the backup server, the former for administration purposes, and the latter for backup purposes, as its name suggest it.

Firewall

The firewall is a simple computer with `pfSense` installed on it.

`pfSense` is a `FreeBSD` based software that is often used to power firewalls. This means that there are some hardware requirements involved:

> ⚠ Warning!
>
> - the device must be powered by an **AMD64 CPU** (because ARM CPUs are not widely supported) and its network interfaces mustn't use **Realtek chipsets** (Intel chipset are recommended because of compatibility issues).
> - we need at least three network interfaces (**WAN**, **LAN** and **OPT1**).
>
> `In our case, the hardware doesn't matter because it is installed on a VM.`

Description

The firewall has several uses in this architecture:

- **Packets filtering**: restrict access outwards and inwards using rules applying on specific ports and protocols.
- **DHCP (Dynamic Host Configuration Protocol)**: network management protocol used on Internet Protocol (IP) networks for automatically assigning IP addresses and other communication parameters to devices connected to the network using a client–server architecture.
- **DNS (Domain Name System)**: hierarchical and distributed name service that provides a naming system for computers, services, and other resources on the Internet or other Internet Protocol (IP) networks.
- **Router**: acts as a bridge between different networks (here, between the sub-LANs and the WAN).
- **NAT (Network Address Translation)**: method of mapping an IP address space into another by modifying network address information in the IP header of packets while they are in transit across a traffic routing device (here, rules for HTTP and HTTPS requests for the web server).
- **VPN (Virtual Private Network)**: creates a secure connection on a public network.

- **Monitoring and reports**: provides tools to monitor the traffic on the network and to generate detailed reports.
- **Bandwidth management**: controls and limits the bandwidth.

Network interfaces

LAN (Local Area Network) & OPT1

The LAN (and OPT1) interface is connected to our internal private networks. It is responsible for the communication between devices on those private networks.

- **IP Addresses**: LAN 10.0.0.1, OPT1 10.0.0.9
- **Description**: Gives access to internet to all devices connected to that interface and applies security rules to protect the local networks.

WAN (Wide Area Network)

The WAN interface is connected to internet. It receives a public IP address from the ISP (Internet Service Provider). **On our VM, we configured this interface in `NAT`, so that it receives an IP address from the host, without having problems with the network configurations at YNOV**.

- **IP Address**: Attributed by DHCP (VM's host)
- **Description**: Manages the inward and outward traffic, applying the security rules to protect from outside threats.

DMZ (Demilitarized Zone)

The DMZ interface is used to host services reachable from internet, isolating it from the local network (LAN) for security purposes.

- **IP Address**: 10.0.1.1
- **Description**: Hosts our webserver which documentation is publicly reachable and applies security rules to limit access to specific services.

Filtering Rules

LAN

Rules (Drag to Change Order)

	States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
☐	✓ 0/6.29 MiB	IPv4 TCP	Linux_ Workstation_IP	*	LAN address	443 (HTTPS)	*	none		Allow access to pfSense's webconfigurator.	
☐	✓ 1/309 KiB	IPv4 TCP/ UDP	LAN subnets	*	LAN subnets	Usual_ Ports	*	none		Allow internal traffic.	
☐	✓ 0/18 KiB	IPv4 TCP/ UDP	Linux_ Workstation_IP	*	WebServer_ DMZ	22 (SSH)	*	none		Allow SSH connection from Admin client (Linux Workstation) to Webserver DMZ.	
☐	✓ 0/1.00 MiB	IPv4 TCP	LAN subnets	*	WebServer_ DMZ	WebPorts	*	none		Allow HTTP(S) access to Webserver from LAN.	
☐	✓ 5/60.96 MiB	IPv4 *	LAN subnets	*	! 10.0.0.0/4	*	*	none		Permit all outwards traffic from LAN to public network.	
☐	✓ 0/1.22 MiB	IPv4 TCP/ UDP	LAN subnets	*	Jetbrains	443 (HTTPS)	*	none		Static files requests for documentation website to Jetbrains server.	

OPT1

Rules (Drag to Change Order)

	States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
☐	✓ 0/22 KiB	IPv4 TCP	OPT1 subnets	*	OPT1 address	53 (DNS)	*	none		Allow DNS requests from OPT1 subnets to OPT1 address.	
☐	✓ 0/8 KiB	IPv4 TCP	OPT1 subnets	*	WebServer_ DMZ	WebPorts	*	none		Allow HTTP(S) access to Webserver from OPT1.	
☐	✓ 0/1.07 MiB	IPv4 TCP/ UDP	Backup_ Server_IP	*	WebServer_ DMZ	22 (SSH)	*	none		Allow SSH connection from Backup Server OPT1 to Webserver DMZ.	
☐	✓ 0/0 B	IPv4 TCP/ UDP	OPT1 subnets	*	! 10.0.0.0/4	*	*	none		Allow traffic from OPT1 subnets to public network (ENABLE ONLY WHEN UPDATING BACKUP SERVER).	

WAN

Rules (Drag to Change Order)

	States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
☐	✓ 0/6.29 MiB	IPv4 TCP	Linux_ Workstation_IP	*	LAN address	443 (HTTPS)	*	none		Allow access to pfSense's webconfigurator.	
☐	✓ 1/309 KiB	IPv4 TCP/ UDP	LAN subnets	*	LAN subnets	Usual_ Ports	*	none		Allow internal traffic.	
☐	✓ 0/18 KiB	IPv4 TCP/ UDP	Linux_ Workstation_IP	*	WebServer_ DMZ	22 (SSH)	*	none		Allow SSH connection from Admin client (Linux Workstation) to Webserver DMZ.	
☐	✓ 0/1.00 MiB	IPv4 TCP	LAN subnets	*	WebServer_ DMZ	WebPorts	*	none		Allow HTTP(S) access to Webserver from LAN.	
☐	✓ 5/60.96 MiB	IPv4 *	LAN subnets	*	! 10.0.0.0/4	*	*	none		Permit all outwards traffic from LAN to public network.	
☐	✓ 0/1.22 MiB	IPv4 TCP/ UDP	LAN subnets	*	Jetbrains	443 (HTTPS)	*	none		Static files requests for documentation website to Jetbrains server.	

DMZ

Rules (Drag to Change Order)

	States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
☐ ✓	0/0 B	IPv4 TCP	Backup_Server_IP	*	WebServer_DMZ	22 (SSH)	*	none		Allow access to Webserver from Backup Server over SSH.	
☐ ✓	0/0 B	IPv4 TCP	Linux_Workstation_IP	*	WebServer_DMZ	22 (SSH)	*	none		Allow access to Webserver from Linux Client (administrator) over SSH.	
☐ ✓	0/0 B	IPv4 TCP	*	*	WebServer_DMZ	WebPorts	*	none		Allow access to the Webserver's website over HTTP(S).	
☐ ✗	0/0 B	IPv4 *	DMZ subnets	*	OPT1 subnets	*	*	none		Block traffic from DMZ to OPT1 subnets.	
☐ ✗	0/756 B	IPv4 *	DMZ subnets	*	LAN subnets	*	*	none		Block traffic from DMZ to LAN subnets.	
☐ ✓	0/46 KiB	IPv4 TCP	WebServer_DMZ	*	DMZ address	53 (DNS)	*	none		Allow DNS requests from Webserver to DMZ address.	
☐ ✓	0/504 B	IPv4 ICMP any	WebServer_DMZ	*	DMZ address	*	*	none		Allow ping from Webserver to DMZ address.	
☐ ✓	0/0 B	IPv4 TCP/UDP	WebServer_DMZ	*	! 10.0.0.0/4	Usual_Ports	*	none		Permit outward traffic from Webserver to public network (ENABLE ONLY WHEN UPDATING WEBSERVER).	
☐ ✓	0/0 B	IPv4 ICMP any	WebServer_DMZ	*	! 10.0.0.0/4	*	*	none		Allow ping from Webserver to public network (FOR TESTING PURPOSES ONLY).	

Conclusion

The **pfSense** firewall plays a crucial role in the security of our network infrastructure controlling all traffic between its interfaces (WAN, LAN, OPT1, DMZ) applying security rules that we chose and just presented.

This documentation provides a complete overview of the configurations of pfSense and its filtering rules.

Web Server

Overview

The web server is a Linux server that hosts this documentation and is accessible from outside or inside the enterprise's private local network.

This website is static and generated with Jetbrains Writerside, a plugin used to make documentation. It has next been manually migrated to **Bookstack**.

Configuration

The static HTML, CSS and JS files are served using `caddy` as a server.

- as a server, `caddy` listens to a specific port and answers requests sending the corresponding ressources (web pages in our case);
- it also generates certificates (auto-signed or using Let's Encrypt, depending on the domain name specified in the configuration).

Server

The server listens to port `80` and `443` and serves files present in `/var/www/infraSI/`, where the static files for this documentation are located. All `HTTP` requests are redirected to `HTTPS` for security purposes.

```
# Unless the file starts with a global options block, the first
# uncommented line is always the address of your site.
#
# To use your own domain name (with automatic HTTPS), first make
# sure your domain's A/AAAA DNS records are properly pointed to
# this machine's public IP, then replace ":80" below with your
# domain name.

localhost, webserver.local, 10.0.1.2 {
        # Set this path to your site's directory.
        root * /var/www/infrasi

        # Enable the static file server.
        file_server

        # Another common task is to set up a reverse proxy:
        # reverse_proxy localhost:8080

        # Or serve a PHP site through php-fpm:
        # php_fastcgi localhost:9000
}

# Refer to the Caddy docs for more information:
# https://caddyserver.com/docs/caddyfile
```

Backup server

This guide walks you through configuring BorgBackup for automated backups on your Ubuntu server.

SSH Key pair

- The Backup Server uses an SSH Key pair to perform its backups and restoration jobs.

---

1. Overview

BorgBackup is a program that performs and manages backups, and is usable between the local device and a remote one.

In our case, the **Backup Server** backs up files from a remote device over SSH and manages it locally. That use-case is not common, because normally the contrary is done, and is supported natively. To do what we wanted here, we needed to use `sshfs` to *mount* the remote directory locally through **SSH**, and then perform the backup or restoration, before *unmounting* the remote directory.

The backups are automatically done every day at midnight, and then removed periodically (leaving only a few).

The restoration feature is manual, and a **Bash script** is to use in this case. The user needs to type the date in format `YYYY-MM-DD` to restore that specific snapshot into the **Webserver**.

---

2. Backup script

: `~/backup.sh`:

```
#!/bin/bash


# Configuration
```

```
REMOTE_USER="webserver"
REMOTE_HOST="10.0.1.2"
REMOTE_PATH="/var/www/infrasi"
MOUNT_POINT="/mnt/backup_source"
REPOSITORY="/home/backup-server/backups"
ARCHIVE_NAME="$(hostname)-$(date +%Y-%m-%d)"
BACKUP_DEST="$REPOSITORY::$ARCHIVE_NAME"

# Debugging output
echo "Backup Source: $REMOTE_USER@$REMOTE_HOST:$REMOTE_PATH"
echo "Backup Destination: $BACKUP_DEST"

# Create a mount point directory if it does not exist
if [ ! -d "$MOUNT_POINT" ]; then
    echo "Creating mount point directory..."
    mkdir -p "$MOUNT_POINT"
fi

# Mount the remote directory using sshfs
echo "Mounting the remote directory..."
if ! mountpoint -q "$MOUNT_POINT"; then
    sshfs "$REMOTE_USER@$REMOTE_HOST:$REMOTE_PATH" "$MOUNT_POINT"
    if [ $? -ne 0 ]; then
        echo "Failed to mount the remote directory. Aborting backup."
        exit 1  # Exit with a non-zero status to indicate failure
    fi
fi

# Check destination path
echo "Checking destination path..."
if [ ! -d "$REPOSITORY" ]; then
    echo "Destination directory does not exist. Creating it now."
    mkdir -p "$REPOSITORY"
    if [ $? -ne 0 ]; then
        echo "Failed to create destination directory. Aborting backup."
        exit 1  # Exit with a non-zero status to indicate failure
    fi
fi

# Initialize the Borg repository if it does not exist
```

```
if ! borg info $REPOSITORY &>/dev/null; then
    echo "Initializing the Borg repository..."
    borg init --encryption=repokey $REPOSITORY
    if [ $? -ne 0 ]; then
        echo "Borg repository initialization failed. Check the logs for details."
        exit 1  # Exit with a non-zero status to indicate failure
    fi
fi

# Run BorgBackup with options
echo "Starting Borg Backup..."
borg create -v --stats "$BACKUP_DEST" "$MOUNT_POINT"

# Check the status of the backup command
if [ $? -ne 0 ]; then
    echo "Borg Backup failed. Check the above logs for details."
    exit 1  # Exit with a non-zero status to indicate failure
else
    echo "Borg Backup completed successfully."
    # Verify the backup
    echo "Verifying the backup..."
    borg check "$BACKUP_DEST"
    if [ $? -ne 0 ]; then
        echo "Backup verification failed. Check the logs for details."
        exit 1  # Exit with a non-zero status to indicate verification failure
    else
        echo "Backup verification completed successfully."
    fi
fi

# Unmount the remote directory
echo "Unmounting the remote directory..."
fusermount -u "$MOUNT_POINT"

# Check if unmounting was successful
if [ $? -ne 0 ]; then
    echo "Failed to unmount the remote directory. Manual cleanup might be required."
    exit 1  # Exit with a non-zero status to indicate failure
fi
```

```
echo "Script completed successfully."
```

The automation of the backup is done this way:

```
crontab -e


0 0 * * * /home/user/backup_scripts/backup.sh >> /var/log/backup.log 2>&1
```

---

3. Restoration script

: `~/restore.sh` :

```
#!/bin/bash

# Configuration
REMOTE_USER="webserver"
REMOTE_HOST="10.0.1.2"
REMOTE_PATH="/var/www/infrasi"
MOUNT_POINT="/mnt/backup_source"
REPOSITORY="/home/backup-server/backups"
ARCHIVE_NAME="$(hostname)-$(date +%Y-%m-%d)"
BACKUP_DEST="$REPOSITORY::$ARCHIVE_NAME"

# Retreiving the date of the snapshot to restore:
echo "Type the date of the snapshot you want to restore: [YYYY-MM-DD]"
read -p ">> " date

# Mount the remote directory using sshfs
echo "Mounting the remote directory..."
if ! mountpoint -q "$MOUNT_POINT"; then
    sshfs "$REMOTE_USER@$REMOTE_HOST:$REMOTE_PATH" "$MOUNT_POINT"
    if [ $? -ne 0 ]; then
        echo "Failed to mount the remote directory. Aborting restoration."
        exit 1  # Exit with a non-zero status to indicate failure
    fi
fi

# Restore the requested version into the webserver
```

```
borg extract $REPOSITORY::$(hostname)-$date $MOUNT_POINT

# Unmount the remote directory
echo "Unmounting the remote directory..."
fusermount -u "$MOUNT_POINT"

# Check if unmounting was successful
if [ $? -ne 0 ]; then
    echo "Failed to unmount the remote directory. Manual cleanup might be required."
    exit 1  # Exit with a non-zero status to indicate failure
fi

echo "Script completed successfully."
```

Clients

The clients are, as indicated in the presentation, two: : - Linux Workstation : - Windows Workstation.

Linux Workstation

The Linux Workstation is a VM running Ubuntu 22.04 with a desktop environment (GNOME) and `ssh` installed (to access to the Backup Server and Webserver as an administrator).

It has a single network interface set at `VMnet1`, to match the firewall's **LAN interface**.

There are almost no configurations to do here, just to enable the network interface and set it to `automatic`, so that the firewall gives it an IP address on the **LAN network** (10.0.0.0 /29) with the **DHCP** service.

Windows Workstation

The Windows Workstation is a VM running Windows 11.

As the Linux Workstation, it has a single network interface set at `VMnet1` to match the firewall's **LAN interface**.

Same as before, there is only the network to configure and set to `automatic` for it to receive an IP address from the firewall.